UNIVERSIDAD NACIONAL DE COLOMBIA

Dinámica del volteo de bloques en taludes rocosos

Juan Pablo Romero Bermúdez

Universidad Nacional de Colombia
Facultad de Ingeniería, Programa de Maestría en Ingeniería - Geotecnia
Bogotá, Colombia
2019

Dinámica del volteo de bloques en taludes rocosos

Juan Pablo Romero Bermúdez

Trabajo de grado presentado como requisito parcial para optar al título de:
Magíster en Ingeniería - Geotecnia

Director(a):
Ingeniero Civil, MSc. en Geotecnia, Felíx Hernández Rodríguez

Línea de Investigación:
Modelación y análisis en geotecnia
Grupo de Investigación en Geotecnia - GIGUN

Universidad Nacional de Colombia
Facultad de ingeniería, Programa de Maestría en Ingeniería - Geotecnia
Bogotá, Colombia
2019

The dynamic description of a many-particle system is technically unrealizable, theoretically unsuitable, and practically useless

A.N. Matveev

Resumen

En este trabajo se presenta el análisis dinámico de un proceso de volteo de bloques en un talud rocoso. Para esto, se creó el programa *Toppling_UN*, que es capaz de reproducir el movimiento de un sistema de bloques. El programa se formuló con base en el Método de los Elemento Discretos (DEM). Este método es un procedimiento numérico que permite estudiar el movimiento de un sistema de partículas, en este caso bloques rígidos, a partir del cálculo de la interacción entre bloques. Las fuerzas de choque entre bloques se simulan por medio de unidades reológicas, mientras el movimiento se calcula a través de un conjunto de ecuaciones que surgen de la segunda Ley de Newton.

El programa se aplica en un talud rocoso que es susceptible a volteo. Como resultado se obtiene la fuerza, la velocidad y el desplazamiento de cada uno de los bloques que conforman el talud. Con esta información se analiza el comportamiento dinámico del talud para diferentes valores en el ángulo de fricción de los bloques. Al final se puede observar que, aunque talud sea establece, es posible que se presenten una serie de movimiento locales por la inestabilidad individual de los bloques.

Palabras clave: Volteo de rocas, Método de los Elementos Discretos (DEM), Bloque Rígido.

Abstract

This document presents the dynamic analysis of a toppling's process in a rock slope. For this, the program *Toppling_UN* was created, which is capable of reproducing the movement of a block system. The program was formulated based on the Discrete Element Method (DEM). This method is a numerical procedure that allows to study the movement of a system of particles, in this case rigid blocks, from the calculation of the interaction between blocks. The shock forces between blocks are simulated by rheological units, while the movement is calculated through a set of equations that arise from Newton's second Law.

The program is applied in a rocky slope susceptible to toppling. As a result, is obtained the forces, the velocity and the displacement of each blocks in the slope. With this information, the dynamic behavior of the slope for different values in the friction angle of the blocks is analyzed. It can be seen that, although a slope is established, it is possible that a series of local movements may occur due to the individual instability of the blocks.

Keywords: Toppling, Discrete Elements Method (DEM), Rigid Block

Contenido

Lista de Figuras

Lista de Tablas

Introducción

En este trabajo se presenta el análisis dinámico de un proceso de volteo de bloques mediante el método de los elementos discretos (DEM). Para realizar este análisis, se creó el programa *Toppling_UN*, el cual es capaz de reproducir el movimiento del conjunto de bloques que conforman un talud en un macizo rocoso. El programa fue escrito en lenguaje C++ y se desarrolló con la metodología DEM, de manera que es posible estudiar el movimiento de los bloques desde una condición de equilibrio estático y durante el proceso de falla. Este tipo de estudios es útil para evaluar la vulnerabilidad y, posteriormente, el riesgo de los elementos expuestos a la posible falla de un talud.

En Colombia, los análisis de riesgo se describen en la *Guía metodológica para estudios de amenaza, vulnerabilidad y riesgo por movimientos en masa* del Servicio Geológico Colombiano[1]. Esta Guía realiza el cálculo de la amenaza con base en un análisis de equilibrio límite, que no contempla el movimiento de la masa inestable. Asimismo, la guía propone el calculo de vulnerabilidad mediante la formulación de escenarios de exposición y fragilidad de los elementos expuestos, lo que implica conocer las características del movimiento de la masa inestable. Finalmente, el calculo de riesgo incluye la amenaza, la vulnerabilidad y los costos asociados a los daños ocasionados por el evento. Este tipo de evaluaciones permanecen en el campo de lo cualitativo -o semicuantitativo- y pueden ser perfeccionadas si se logra anticipar la velocidad y la aceleración con que la masa movilizada puede impactar a los elementos considerados. De ahí la importancia de un análisis como el que aquí se propone.

El principal objetivo de este trabajo es presentar una alternativa para compatibilizar el cálculo de las variables de movimiento con el análisis de equilibrio límite, lo que permitirá tener un cálculo cuantitativo de vulnerabilidad. Además, este trabajo pretende continuar con una línea de trabajo incentivada por varios estudiantes del programa de Maestría en Geotecnia de la Universidad Nacional de Colombia, que busca promover el desarrollo y uso de herramientas numéricas propias, y de acceso libre, en la solución de problemas dinámicos de ingeniería geotécnica. Trabajos como los realizados por Claudia Basto [2], Raúl moreno [3] y David Aponte [4], orientados a la mecánica de suelos; Andrés Eraso [5] y Alejandro Becerra [6], orientados a la mecánica de rocas; son muestra del esfuerzo interno por promover estos temas.

El Método de los Elementos Discretos es un procedimiento numérico diseñado para analizar

el comportamiento de sistemas discontinuos. Fue propuesto por Peter Cundall en 1971 [7] como una extensión de la técnica denominada "dinámica molecular", la cual buscaba simular la interacción de muchas partículas de tamaño atómico y molecular [8]. La principal ventaja del método DEM frente a los métodos basados en el continuo, como los elementos finitos, es que permite modelar grandes desplazamientos por medio de las leyes de Newton. Con el paso del tiempo, el DEM ha tenido gran acogida en el campo de la geotecnia, principalmente en la mecánica de rocas y medios granulares. Dentro de las aplicaciones más usuales del método se destacan los análisis de deslizamientos [9][10], la simulación de ensayos de laboratorio [11], los estudios de fracturamiento [12][13], entre otros [14][15].

Para la aplicación del método, se seleccionó un proceso de volteo de rocas, también llamado *toppling* por su nombre en inglés. El volteo de rocas es un tipo de movimiento que consiste en la rotación y traslación simultanea de un sistema de bloques, bajo la acción de fuerzas externas, como la gravedad, y por empujes de los bloques adjuntos [16]. Cada uno de los bloques del sistema tiene la posibilidad de rotar, deslizar y chocar con sus vecinos. En la práctica convencional, el proceso de *toppling* se estudia por medio de equilibrio límite siguiendo la metodología desarrollada por Goodman[17]. Lo que aquí se propone pasa del análisis estático y está orientado a un equilibrio dinámico que se materializa en el programa *Toppling_UN*.

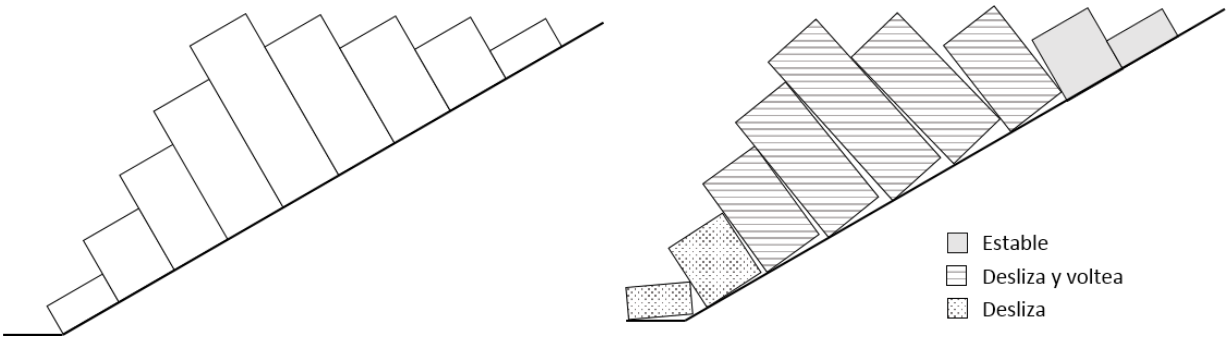
El documento se encuentra dividido en 4 capítulos orientados a la compresión del programa Toppling_UN y a sus aplicaciones. En el Capítulo 1 se describe el procedimiento de análisis de un proceso de *toppling*, con base en la metodología de Goodman, y se presenta una introducción a la solución dinámica por medio del Método de los Elementos Discretos. En el Capítulo 2 se presentan los dos grandes componentes en la formulación del programa: el modelo constitutivo de transmisión de fuerzas entre los bloques y la solución numérica a la dinámica de un bloque rígido. En el Capítulo 3 se muestra la aplicación del programa, para ello se planteó un talud rocoso en que el cual el mecanismo de inestabilidad es volteo. A este talud se le analiza la estabilidad desde el punto de vista dinámico y, además, se estudia el influencia que tiene la fricción sobre su movimiento. Finalmente, en el Capítulo 4 se presentan las conclusiones del trabajo, la recomendaciones en cuanto al uso del programa y los campos de acción para futuros trabajos que se desarrollen.

1. El problema del volteo de bloques

El volteo de bloques rocosos, o *toppling*, es un proceso de remoción en masa que se caracteriza por la rotación y el deslizamiento de uno o más bloques de un sistema. La rotación de estos bloques desencadena un movimiento en serie, en el cual cada bloque del sistema tiene la posibilidad de rotar, deslizar y colisionar con los bloques vecinos. En esta condición, la dinámica del sistema puede desencadenar movimientos transitorios de bloques individuales sin que se desarrolle una falla general. Desde luego, ante un balance crítico entre fuerzas actuantes y resistentes, se debe analizar el movimiento generalizado hasta alcanzar una nueva condición de estabilidad. En este sentido, nuestro propósito es estudiar el movimiento de un talud rocoso susceptible de fallar por el mecanismo de volteo, hasta encontrar una condición de equilibrio.

La Figura **1-1** presenta un talud susceptible de volteo. En la parte (a), el talud se encuentra en una condición inicial sin que se haya iniciado el movimiento de los bloques. En este escenario, los bloques centrales son los más esbeltos y en consecuencia son los bloques que tienen la mayor facilidad de rotar. Por otro lado, en la parte (b) se presenta el mismo talud luego de haber alcanzado una condición de equilibrio. Es decir, una vez ha finalizado el movimiento del sistema de bloques. Se observa que durante el movimiento algunos bloques permanecieron estables, otros han deslizado, mientras los demás han rotado y deslizado en forma simultánea. Cabe mencionar que la falla de un talud depende de cualquier aumento en las cargas del sistema o de la disminución en las características de resistencia de los bloques. Esto quiere decir que aunque el talud haya encontrado el equilibrio, sigue siendo susceptible de nuevos procesos de inestabilidad.

Generalmente, el análisis de volteo se realiza por medio de equilibrio límite siguiendo el procedimiento propuesto por Goodman [17]. En este procedimiento se evalúa la condición de equilibrio de cada bloque, desde el bloque inestable más alto, hasta el bloque inferior del sistema. Si en el bloque inferior las fuerzas actuantes son mayores que las fuerzas resistentes se asume que todo el sistema está en falla. Este análisis permite establecer la condición de equilibrio estático del talud, aunque no da ninguna información acerca de las características del movimiento. Por esta razón, es necesario complementar el método de Goodman con otros métodos que permitan estudiar la dinámica del volteo de bloques. En mecánica de rocas, debido a la naturaleza discontinua de los macizos rocosos, y considerando que los análisis dinámicos requieren un gran número de cálculos, es conveniente utilizar el Método de los



(a) Condición de equilibrio aparente (b) Condición de equilibrio real

Figura 1-1.: Esquema de un talud conformado por bloques

Elementos Discretos (DEM). Este método utiliza la segunda Ley de Newton para calcular las variables dinámicas -posición, velocidad, aceleración y fuerza-, en pequeños y progresivos intervalos de tiempo. De esta manera se consigue un análisis dinámico de volteo.

El objetivo de este capítulo es presentar los fundamentos para realizar el análisis dinámico de un proceso de volteo. Con este propósito se dividió el capítulo en dos partes. En la primera parte se describe el procedimiento propuesto por Goodman, donde se identifican las condiciones necesarias para que se presente el volteo de bloques. En la segunda parte, se explican los fundamentos del Método de los Elementos Discretos, por medio de un par de ejemplos que son: el rebote elástico de una partícula sobre una superficie y el volteo de un bloque sobre una superficie inclinada. Los fundamentos de este capítulo son la base que se emplea en la construcción del programa *Toppling_UN*.

1.1. Solución por equilibrio límite estático

La solución estática al problema de volteo fue abordada en principio por Asbhy [18] en 1971, quién estableció un criterio para definir deslizamiento y volteo de un bloque sobre una superficie inclinada. Años después, en 1976, Goodman [17] propuso un procedimiento para definir el factor de seguridad de un sistema de bloques que conforman un talud sometido a *toppling*. A continuación se desarrolla la formulación que emplearon ambos investigadores.

1.1.1. Estabilidad de un bloque bajo la acción de la gravedad

El problema que aquí se aborda corresponde a un bloque sobre una superficie inclinada como se muestra en la Figura **1-2**. En este ejercicio es fácil deducir que el bloque voltea si la fuerza gravitacional se proyecta por fuera del punto de giro 'O'.

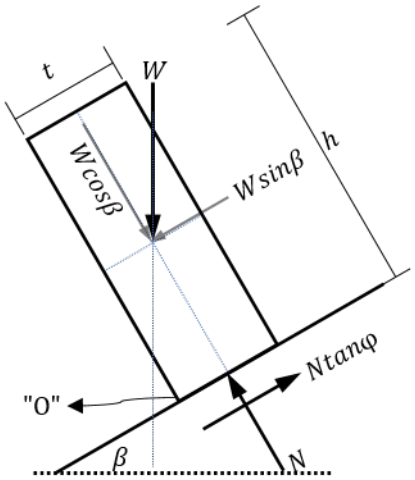


Figura 1-2.: Diagrama de fuerzas de un bloque bajo la acción de la gravedad

En el equilibrio de momentos alrededor del punto 'O' se obtiene que hay volteo si:

$$\begin{aligned} Wh\sin\beta &> Wt\cos\beta \\ \tan\beta &> \frac{t}{h} \end{aligned} \tag{1-1}$$

Con el equilibrio de fuerzas tangenciales se obtiene que hay deslizamiento si:

$$\begin{aligned} W\sin\beta &> W\cos\beta\tan\phi \\ \tan\beta &> \tan\phi \\ \beta &> \phi \end{aligned} \tag{1-2}$$

De las Ecuaciones 1-1 y 1-2 se concluye que un bloque puede deslizar o voltear dependiendo de su geometría, de la inclinación de la superficie y de su ángulo de fricción. Las condiciones de volteo y deslizamiento generan cuatro posibilidades que se resumen en la Figura **1-3** y se describen a continuación:

- Zona 1. $\beta < \phi$ y $\frac{t}{h} > \tan\beta$ Es el escenario de estabilidad, el bloque no desliza ni voltea.
- Zona 2. $\beta > \phi$ y $\frac{t}{h} > \tan\beta$ Se presenta una condición de deslizamiento sin volteo.
- Zona 3. $\beta < \phi$ y $\frac{t}{h} < \tan\beta$ El bloque no desliza pero sí experimenta volteo.
- Zona 4. $\beta > \phi$ y $\frac{t}{h} < \tan\beta$ Ocurre, de forma simultanea, un proceso de deslizamiento y volteo.

Así se analiza la estabilidad de un bloque, lo siguiente es verificar la estabilidad de un sistema de bloques.

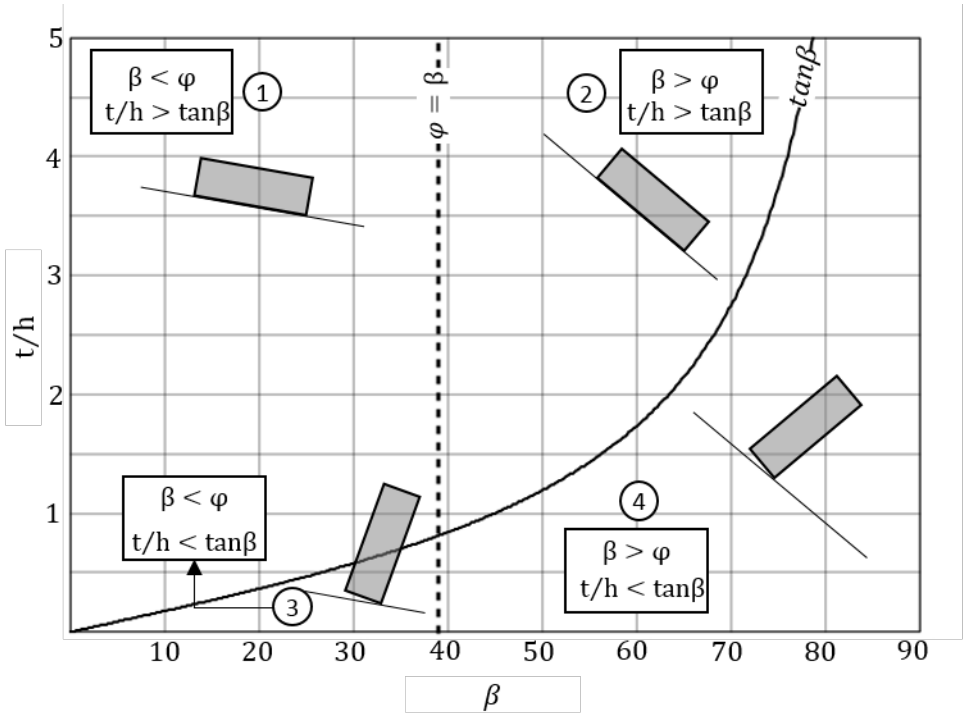


Figura 1-3.: Posibilidades de deslizamiento y volteo de un bloque
adaptado del libro *Rock slope stability* [19]

1.1.2. Análisis de estabilidad de un sistema de bloques

Para analizar este problema, Goodman propone un procedimiento que consiste en un proceso progresivo de transmisión de fuerzas desde el bloque inestable más alto, hasta el bloque en la pata del talud. El método se fundamenta en el cálculo de dos fuerzas: Una por deslizamiento y otra por volteo. La idea principal es transmitir la mayor fuerza que resulte entre las dos. En la Figura **1-4** se muestra el diagrama de fuerzas de un bloque que pertenece a un sistema de bloques. El equilibrio de fuerzas y de momentos se realiza en el bloque n_i. Llamamos bloque n_{i+1} al bloque que se ubica sobre n_i y n_{i-1} al bloque que se ubica por debajo de n_i. Como el análisis se realiza en orden descendente, la incógnita que se debe evaluar es la fuerza P_i, la cual se puede calcular de dos maneras:

$$P_i = P_{i+1} + \frac{W\cos\beta(\tan\beta - \tan\phi)}{1-\tan^2\phi} \tag{1-3}$$

$$P_i = \frac{1}{l}\left(\frac{W}{2}(h\sin\beta - t\cos\beta) + P_{i+1}(m - t\tan\phi)\right) \tag{1-4}$$

La Ecuación 1-3 corresponde a la fuerza P_i calculada por deslizamiento, se obtiene al hacer equilibrio de fuerzas en la dirección paralela y perpendicular a la superficie inclinada. Por otra parte, la Ecuación 1-4 corresponde a la fuerza P_i calculada por volteo, se obtiene al realizar

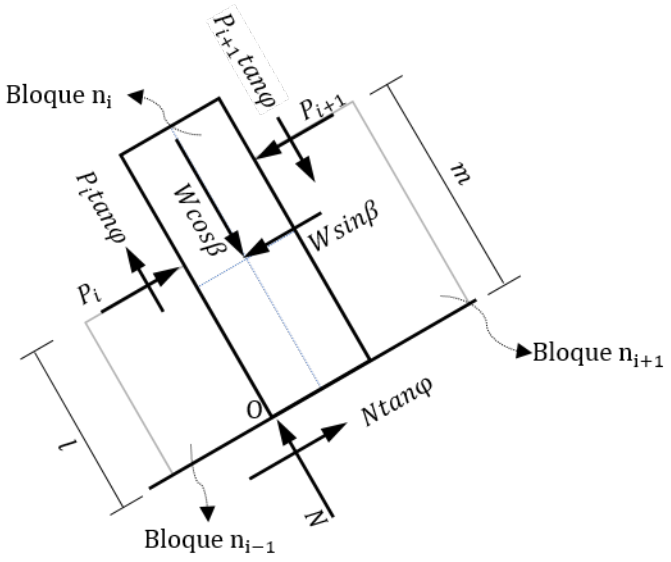


Figura 1-4.: Diagrama de fuerzas de un bloque en un sistema de bloques

equilibrio de momentos en el punto de giro 'O'. A continuación se presenta el procedimiento que se debe seguir para solucionar un problema de *toppling* en condición estática:

1. Se define la geometría del problema.
2. Se selecciona un ángulo de fricción que debe ser mayor a la inclinación de la base para evitar el deslizamiento ($\phi > \beta$).
3. Iniciando en el bloque más alto, se verifica la posibilidad de volteo de cada bloque mediante la Ecuación 1-1.
4. Se denomina n_i al bloque más alto que voltea.
5. En el bloque n_i se usan las Ecuaciones 1-3 y 1-4 para calcular la fuerza P_i por deslizamiento y por volteo. La mayor fuerza P_i que surja de las dos ecuaciones es la fuerza que se transmite al siguiente bloque n_{i-1}.
6. Se avanza al siguiente bloque. Ahora el bloque n_{i-1} es n_i. Se repite el paso 5. Este paso se debe realizar hasta que n_i sea el bloque en la pata del talud.
7. Si el bloque de la pata es estable contra el deslizamiento y volteo, es decir $P_i < 0$, se considera que talud es estable. Si por el contrario, este bloque desliza o rota, el talud se considera inestable.

Para hallar el factor del seguridad se debe encontrar el ángulo de fricción crítico. Esto se consigue al repetir el procedimiento desde el paso 2, con diferentes valores de ϕ, hasta que

la fuerza resultante P_i sea aproximadamente igual a 0 en el bloque de la pata. El valor de ϕ que consiga el $P_i = 0$ en la pata se denomina ángulo de fricción crítico ϕ_c. De esta manera el factor de seguridad para el talud se calcula como:

$$FS = \frac{tan\phi}{tan\phi_c} \tag{1-5}$$

donde ϕ es el ángulo de fricción en el contacto de los bloques.

1.1.2.1. Ejemplo de estabilidad de un sistema de bloques

Como ejemplo de aplicación del método de Goodman, se resuelve el siguiente ejercicio tomado del libro *Rock slope stability* [19]. Considérese el talud que se presenta en la Figura **1-5**, este talud tiene un total de 11 bloques, con una altura H de 9m y un peso unitario γ de $2600kg/m^3$. El talud está inclinado un ángulo α de 64,31° y está limitado por un plano en la base con inclinación β de 30°. Los bloques están escalonados con un ángulo de 3°, que equivale a un escalón de 0.08m. El ángulo de fricción disponible es de 30°.

El problema se resuelve utilizando el algoritmo de computador que se presenta en el Anexo A.1. El algoritmo sigue el procedimiento de Goodman. El primer paso es calcular de las características geométricas del talud, que se presentan en la tabla **1-1**. De allí se puede identificar

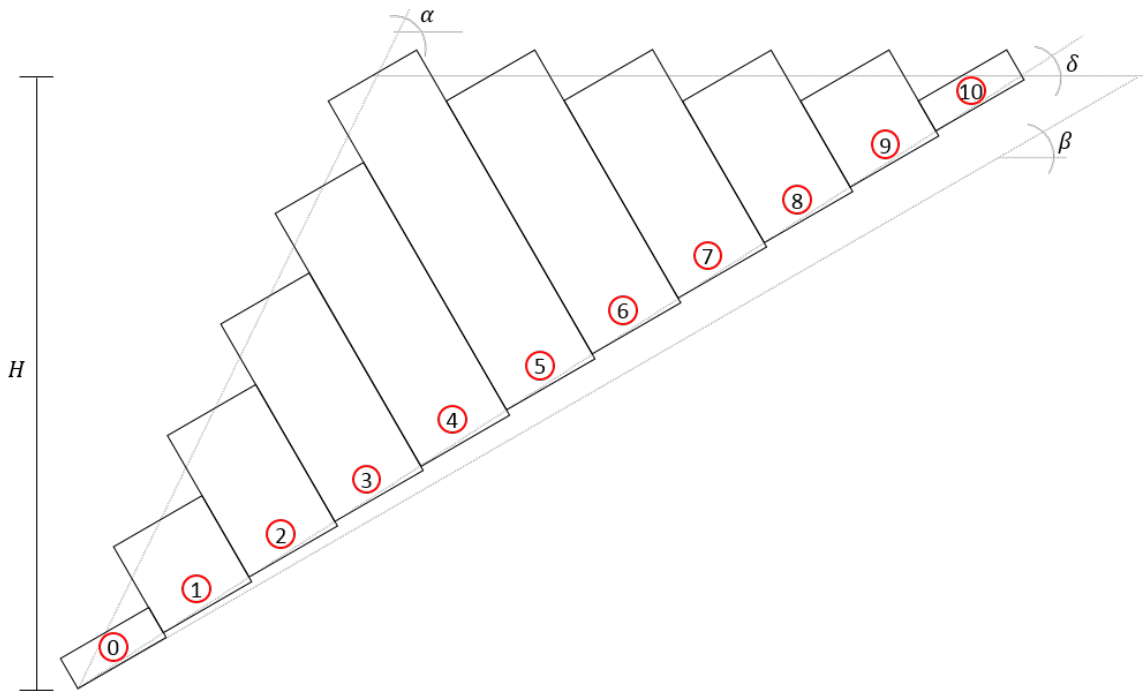


Figura 1-5.: Ejemplo de *toppling* en un talud rocoso

que el primer bloque que tiene la posibilidad cinemática de voltear es el 7. Luego, iniciando desde el bloque 7, se calculan las fuerzas P_i hasta llegar al bloque 0, en donde se determinar si el talud es estable o no. Si el bloque 0 resulta estar en movimiento, por desplazamiento o volteo, todo el talud se considera en falla. En caso el opuesto, si el bloque 0 es estable, el talud está en una condición de estabilidad. El factor de seguridad del talud se obtiene al encontrar la condición de ϕ critico -ϕ_c- y compararlo con el ϕ disponible según la Ecuación 1-5. Con las condiciones de este problema, el ϕ_c es igual 39,75°, con lo cual el factor de seguridad es de 0,74. En la Figura **1-6** se presenta el resultado del análisis estático. De allí se puede concluir

Tabla 1-1.: Geometría del talud de referencia

Bloque	Altura h(m)	Ancho t(m)	Brazo m(m)	Brazo l(m)	t/h	$t/h > \tan\beta$
10	0.51	1.50	0.51	0.25	2.94	No
9	1.46	1.50	1.46	0.43	1.03	No
8	2.40	1.50	2.40	1.38	0.63	No
7	3.35	1.50	3.35	2.32	0.45	Sí
6	4.29	1.50	4.29	3.27	0.35	Sí
5	5.24	1.50	4.37	4.21	0.29	Sí
4	4.29	1.50	3.43	4.29	0.35	Sí
3	3.35	1.50	2.48	3.35	0.45	Sí
2	2.40	1.50	1.54	2.40	0.63	Sí
1	1.46	1.50	0.59	1.46	1.03	Sí
0	0.51	1.50	0.51	0.51	2.94	Sí

que el talud es inestable en su condición de ϕ disponible. Los bloques superiores, en donde la relación de esbeltez es baja, se encuentran en una condición estable. Ningun bloque desliza en su condición individual ya que se parte de una condición en la cual $\phi > \beta$. En los bloques 5, 6, 7 y 8 predomina el movimiento de rotación ya que son los de mayor altura y, por tanto, su centro de masa se proyecta fuera de su punto de giro. Esto genera el choque y transmisión de fuerzas hacia los bloques inferiores. Los bloques 1, 2, 3, y 4 son susceptibles a volteo y deslizamiento. El bloque 0 solo desplaza ya que su relación de altura - ancho no permite el volteo.

Esta es la manera convencional como se analiza la estabilidad por *toppling*. Todo desde un punto de vista estático, dejando un sinfín de interrogante respecto a las características del movimiento de los bloques. En la siguiente sección se presenta una breve introducción a la solución de un problema dinámico por medio del Método de los Elementos Discretos.

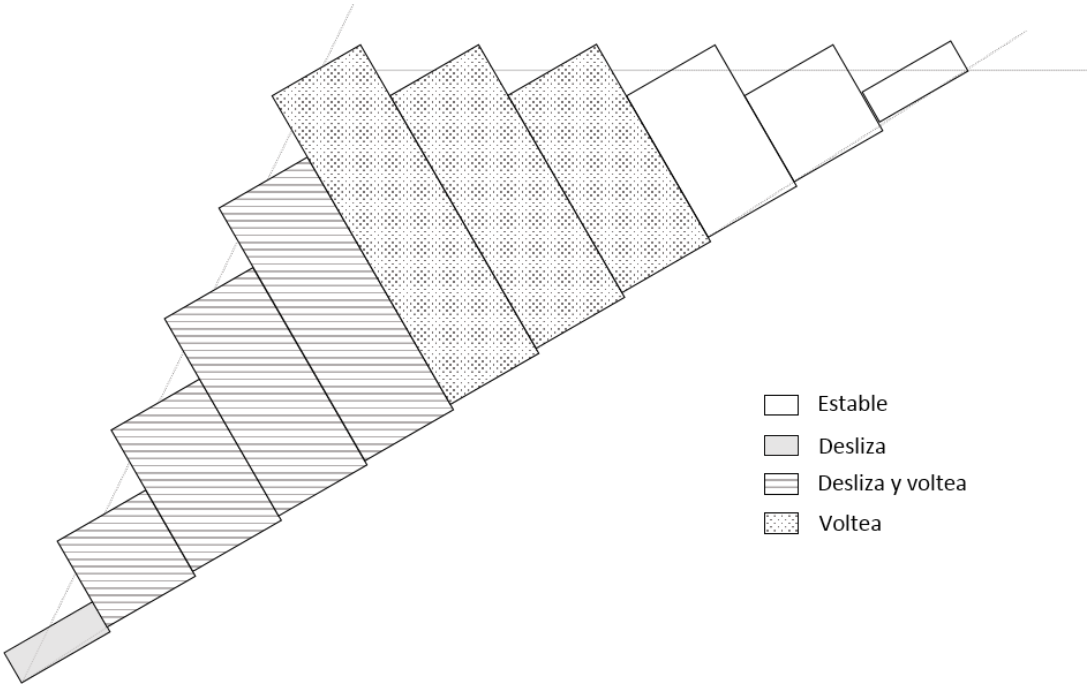


Figura 1-6.: Análisis de toppling en talud conformado por bloques

1.2. Introducción a la solución dinámica

El Método de los Elementos Discretos (DEM), es un procedimiento de análisis numérico que permite seguir la trayectoría, la velocidad, la aceleración y la fuerzas de cada uno de los bloques que conforman un sistema. El método fue propuesto por Peter Cundall en 1971 y presentado bajo el título *A computer model for simulating progressive, large-scale movements in blocky rock systems* [7]. Surge como una extensión a la técnica denominada "Dinámica Molecular" que fue diseñada con el objetivo de evaluar el comportamiento de muchas partículas de tamaño atómico y molecular, sometidas a las fuerzas del campo eléctrico y magnético [8]. Con el paso de los años, el método ha estado en constante desarrollo y ha permitido el surgimiento de algunas variaciones dentro de las cuales destaca el Método de Análisis de Deformaciones Discontinuas (DDA), que fue presentado por primera vez por Goodman y Shi en el año 1984 y extendido en 1988 por Shi en su trabajo doctoral *Discontinuous deformation analysis—a new numerical model for the statics and dynamics of block systems* [20].

Las dos versiones del método, DEM y DDA, que también son llamadas DEM explícito y DEM implícito, utilizan la segunda Ley de Newton para predecir la posición futura de los elementos de un sistema. La posición final de los elementos se obtiene luego de un proceso de análisis que avanza en el tiempo en muy pequeños intervalos desde una condición inicial. La principal diferencia entre las dos versiones del método es que el DEM emplea el concepto de

Momento Angular para marchar la solución en el tiempo, mientras el DDA utiliza la formulación de trabajo y energía, con el objetivo de minimizar la energía potencial del sistema de bloques antes de avanzar al siguiente intervalo de tiempo. Más allá de esto, ambos versiones del método son de uso actual en la geotecnia y se han validado con bastante aceptación en múltiples problemas de la mecánica de rocas [21][22].

No hay manera de determinar cuál versión es mejor ya que ambas presentan ventajas y desventajas. Quizás, el criterio de selección más común está relacionado con la mayor crítica al DEM: esta versión del método permite la superposición de bloques durante un corto periodo de tiempo; es decir, en varios instante de tiempo es posible que dos bloques se solapen. Por supuesto, la superposición de bloques no sucede en la realidad. Para solucionar esta situación, el DDA realiza un proceso de cálculo iterativo adicional, donde verifica que no haya superposición y luego avanza al siguiente intervalo tiempo. En contra parte, los ciclos adicionales del DDA requieren un mayor uso de recursos computacionales y esto se refleja en el tiempo de ejecución de los programas. Para el desarrollo del programa Topppling_UN se usa la técnica del elemento discreto explícito, DEM.

1.2.1. Fundamentos del método DEM

El método DEM calcula las características dinámicas de un elemento, en progresivos y continuos intervalos de tiempo. Este cálculo se realiza en muy pequeños pasos de tiempo denominados Δt, desde un tiempo inicial t_0 hasta un tiempo final t_f como se muestra en la Figura **1-7**. Durante los intervalos Δt se asume que las variables dinámicas son constantes. Esto significa que en el intervalo que va de t_0 a t_1, un elemento tendrá una posición fija en r_0, una velocidad v_0 y una aceleración a_0.

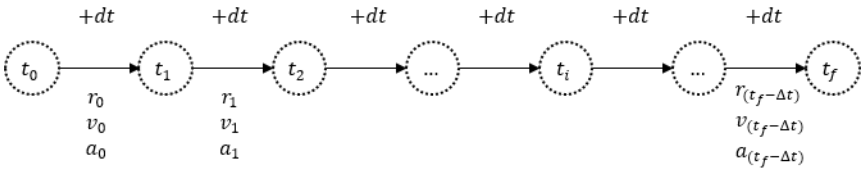


Figura 1-7.: Procedimiento progresivo DEM

Con base en esto, un programa DEM está conformado por tres componentes principales: un conjunto de elementos discretos, bien sea partículas para suelos o bloques para macizos rocosos; un modelo de contacto para identificar los posibles choques entre elementos y para calcular las fuerzas que se generan; y unas ecuaciones de movimiento de la mecánica clásica.

Estos componentes se encuentran implícitos en el proceso cíclico de tres pasos que se presenta en la Figura **1-8**. El ciclo se realiza en cada instante de tiempo t y para cada uno de los elementos que conforman un sistema. El ciclo comienza en una condición inicial en el cual se conoce la ubicación y orientación de todos elementos. En ese momento se verifican los posibles choques entre elementos y se calcula la Fuerza Neta por medio del equilibrio de fuerzas externas y fuerzas internas. De forma simultanea, se calcula el Torque Neto a través del equilibrio de momentos. Luego de esto, se calcula la velocidad lineal y angular para un instante posterior Δt por medio de las ecuaciones de la dinámica. Con la nueva velocidad es sencillo obtener la posición en el tiempo $t + \Delta t$. El ciclo se repite secuencialmente en incrementos de tiempo Δt, a partir de una condición inicial definida por las condiciones del problema, hasta la condición de equilibrio final del sistema.

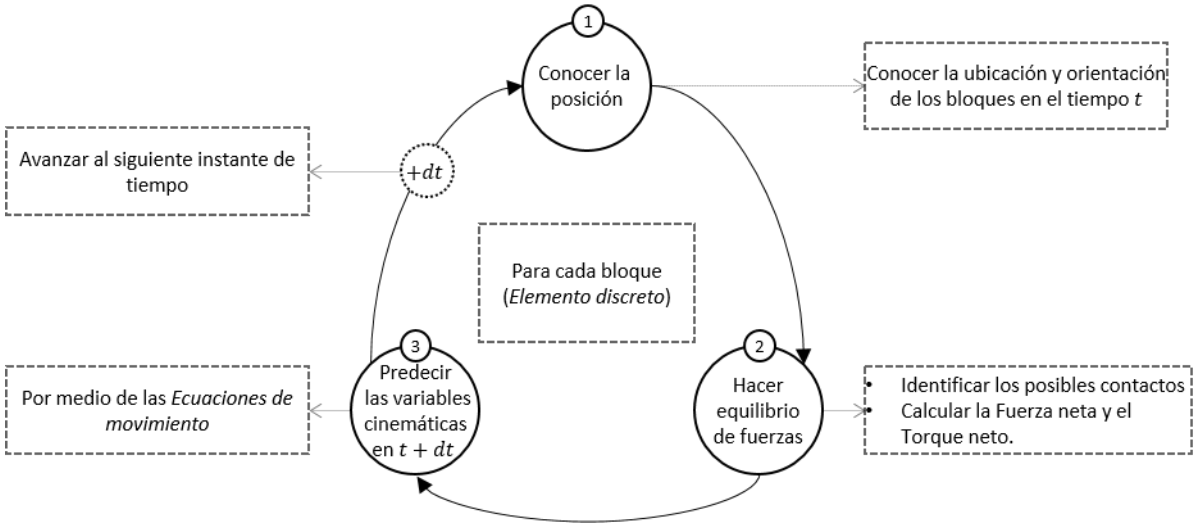


Figura 1-8.: Proceso iterativo de cálculo en cada instante de tiempo

Para comprender estos conceptos se propone el siguiente ejemplo de choque entre dos cuerpos. Considérese una partícula suspendida en el aire sin experimentar ninguna velocidad ni fuerza diferente a su peso. Bajo la partícula se encuentra ubicada una superficie inclinada como se aprecia en la Figura **1-9**. El objetivo del problema es predecir la posición de la partícula en todo instante t desde un tiempo inicial t_0 hasta un tiempo final t_f. Antes de realizar la solución por el procedimiento DEM se tienen que definir las ecuaciones de fuerza y de movimiento.

1.2.1.1. Ecuaciones de Fuerza

Al resolver este problema conviene analizar tres escenarios: antes de la colisión -entre t_0 y t_1-, durante la colisión -entre t_1 y t_2- y después de la colisión -entre t_2 y t_f-. Antes y después

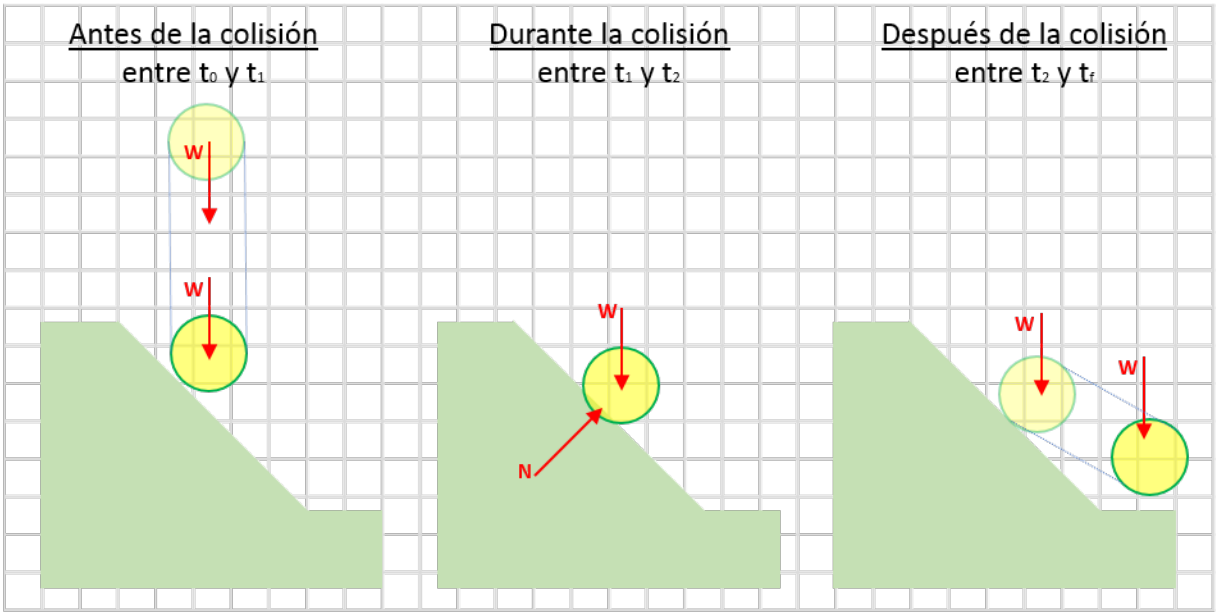


Figura 1-9.: Ejemplo de choque entre dos cuerpos

de la colisión se sabe que no hay choque. Por tanto, la Fuerza neta es:

$$\vec{F} = \vec{W} \tag{1-6}$$

Durante la colisión la partícula está en contacto con la superficie, lo que origina una fuerza de repulsión N que permanece desde el tiempo t_1 hasta el tiempo t_2. En este intervalo la Fuerza neta es:

$$\vec{F} = \vec{W} + \vec{N} \tag{1-7}$$

1.2.1.2. Ecuaciones de Movimiento

Como la partícula es redonda, únicamente es necesario calcular su traslación, la rotación no es relevante en este caso. Por tanto, la aceleración de la partícula se calcula de acuerdo a la segunda Ley Newton como:

$$\vec{a}_t = \frac{\vec{F}}{m} \tag{1-8}$$

donde $\vec{F}$ es la fuerza neta de la partícula en el tiempo t, m es la masa de la partícula y $\vec{a}_t$ es la aceleración de la partícula en el tiempo t. La Fuerza $\vec{F}$ es constante entre los pasos de tiempo Δt.

La aceleración es la variación de la velocidad en el tiempo, $a = dv/dt$, luego es posible reescribir la Ecuación 1-8 como:

$$\frac{dv}{dt} = \frac{\vec{F}}{m} \tag{1-9}$$

al separar variables e integrar en el intervalo $[t, t + \Delta t]$ se obtiene una relación entre la velocidad en el tiempo t y el tiempo posterior $t + \Delta t$ así:

$$\begin{aligned} \int_{\vec{v}_t}^{\vec{v}_{t+\Delta t}} dv &= \int_t^{t+\Delta t} \frac{\vec{F}}{m} dt \\ \vec{v}_{t+\Delta t} &= \vec{v}_t + \Delta t \frac{\vec{F}}{m} \end{aligned} \tag{1-10}$$

De forma similar, la velocidad es la razón de cambio de la posición en el tiempo, $v = dr/dt$, entonces la Ecuación 1-10 se reescribe como:

$$\frac{dr}{dt} = \vec{v}_t + \Delta t \frac{\vec{F}}{m} \tag{1-11}$$

y al integrar por partes es posible establecer una relación entre la posición de la partícula en un tiempo t y $t + \Delta t$ así:

$$\begin{aligned} \int_{\vec{r}_t}^{\vec{r}_{t+\Delta t}} dr &= \int_t^{t+\Delta t} (\vec{v}_t + \Delta t \frac{\vec{F}}{m}) dt \\ \vec{r}_{t+dt} &= \vec{r}_t + \vec{v}_t \Delta t + \frac{1}{2} \frac{\vec{F}}{m} \Delta t^2 \end{aligned} \tag{1-12}$$

La Ecuación 1-12 es la que permite predecir la posición de la partícula en cualquier instante t.

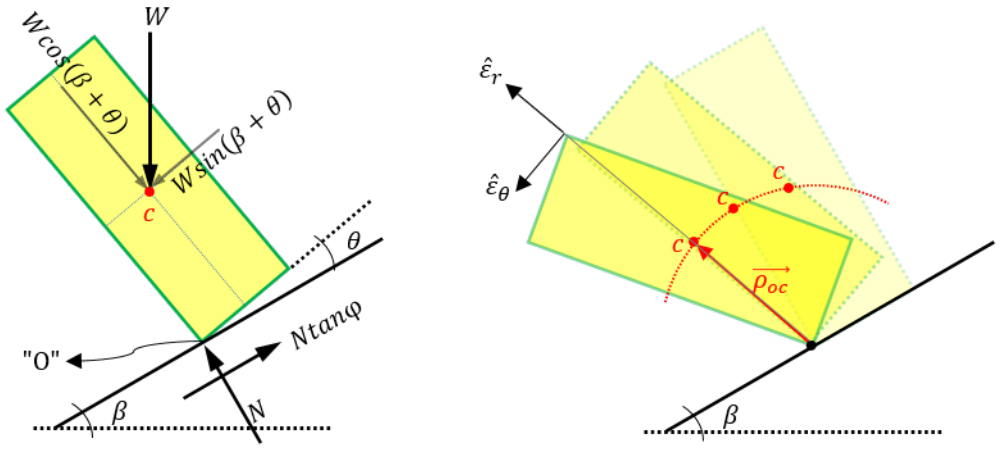
1.2.1.3. Solución por el Método DEM

La solución del problema por medio del ciclo DEM, de la Figura **1-8**, inicia con conocer la posición en t_0. En ese tiempo se calcula la Fuerza Neta por la Ecuación 1-6 -no hay choques-. Luego se calcula la velocidad y la posición en el siguiente intervalo de tiempo por medio de las Ecuaciones 1-10 y 1-12. Conocida la nueva posición se incrementa el tiempo de análisis en Δt y se repite el ciclo para el nuevo tiempo. Cuando el tiempo de análisis se encuentra en el intervalo $[t_1, t_2]$ la Fuerza Neta se calcula por la Ecuación 1-7 -hay Choques-.

Así se analiza el problema del ejemplo. En el caso de volteo de bloques el procedimiento es similar, sólo se deben añadir las ecuaciones que describen la rotación. A continuación se trata el problema de un bloque que voltea sin deslizar.

1.2.2. Solución dinámica al volteo de un bloque

En la Figura **1-10** se muestra un bloque que voltea sobre una superficie inclinada, alrededor de un punto fijo o. El bloque se comporta como un sólido rígido, al cual llamamos $\mathcal{S}$. El objetivo del problema es definir el ángulo de giro del bloque durante todo el tiempo que dure el proceso de volteo, considerando que el punto o no se desplaza. La rotación del bloque se mide en sentido antihorario por medio de un ángulo de giro θ. El bloque sigue una trayectoria circular y procedemos a analizar lo que sucede en el centro de masa del bloque -punto c-, por medio del vector $\vec{\rho}_{oc}$ que va de o a c, como se muestra en la parte (b) de la Figura **1-10**.



(a) Diagrama de cuerpo libre

(b) trayectoria del centro de masa y vector de posición

Figura 1-10.: Un bloque rotando sobre una superficie

Al igual que en el ejemplo anterior, se tienen que definir las ecuaciones de fuerza y de movimiento antes de aplicar el procedimiento DEM. Esto se realiza a partir del análisis dinámico de un bloque rígido, por medio de la formulación de Momento Angular.

1.2.2.1. Ecuaciones de Fuerza

De la dinámica de bloque rígido, se sabe que la fuerza total externa que actúa sobre un bloque $\mathcal{S}$ es la suma vectorial de todas las fuerzas f_i sobre él; esto es:

$$\vec{F} = \sum_{i=1}^{n} \vec{f}_i \tag{1-13}$$

Así mismo, el momento de la fuerza resultante respecto al punto "o" se calcula como la suma vectorial del momento producido por cada una de las fuerzas f:

$$\vec{M}_o = \sum_{i=1}^{n} \vec{m}_{oi} = \sum_{i=1}^{n} \vec{\rho}_{oi} \times \vec{f}_i \tag{1-14}$$

En este caso se considera que únicamente actúa la fuerza gravitacional en el centro de masa del bloque. La fuerza Neta y el Momento Neto que en el bloque son:

$$\vec{F} = m\vec{g} = \vec{W} \tag{1-15}$$

$$\vec{M}_o = \vec{\rho}_{oc} \times \vec{W} \tag{1-16}$$

1.2.2.2. Ecuaciones de Movimiento

La formulación de momento angular establece que el momento de una fuerza f_i, respecto a un punto o, es la derivada en el tiempo del momento de su cantidad de movimiento $\vec{p}_i$ respecto al mismo punto o. Esto quiere decir que:

$$\begin{aligned} \vec{m}_{oi} &= \frac{d}{dt}(\vec{\rho}_{oi} \times \vec{p}_i) \\ &= \frac{d}{dt}(\vec{\rho}_{oi}) \times \vec{p}_i + \vec{\rho}_{oi} \times \frac{d}{dt}(\vec{p}_i) \end{aligned} \tag{1-17}$$

La cantidad de movimiento $\vec{p}_i$ es el producto de la masa del punto i por su velocidad $\vec{v}_i$, con lo cual la Ecuación 1-17 se reescribe como:

$$\begin{aligned} \vec{m}_{oi} &= \frac{d}{dt}(\vec{\rho}_{oi}) \times (m\vec{v}_i) + \vec{\rho}_{oi} \times \frac{d}{dt}(m\vec{v}_i) \\ &= \vec{v}_i \times (m\vec{v}_i) + \vec{\rho}_{oi} \times (m\vec{a}_i) \\ &= \vec{\rho}_{oi} \times (m\vec{a}_i) \end{aligned} \tag{1-18}$$

donde $\vec{a}_i$ es la aceleración del punto i y se puede calcular como el producto cruz entre la aceleración angular del bloque $\vec{\alpha}_i$ y el vector $\vec{\rho}_{oi}$:

$$\vec{a}_i = \vec{\alpha}_i \times \vec{\rho}_{oi} \tag{1-19}$$

Remplazando la Ecuación 1-19 en 1-18 se tiene:

$$\vec{m}_{oi} = m(\vec{\rho}_{oi} \times \vec{\alpha}_i \times \vec{\rho}_{oi}) \tag{1-20}$$

Esta ecuación relaciona el momento de fuerzas de un punto i, que pertenece al bloque $\mathcal{S}$ y que gira alrededor de o, con su velocidad angular $\vec{\alpha}_i$. Como el bloque $\mathcal{S}$ es un sólido rígido todos los puntos en él giran con la misma velocidad angular $\vec{\alpha}$. El momento de fuerzas de $\mathcal{S}$ se obtiene al integrar la Ecuación 1-20, similar a lo que se hizo en la Ecuación 1-14. Esta

integral se facilita al utilizar el sistema de coordenadas cilíndricas $(\hat{\varepsilon_r}, \hat{\varepsilon_\theta}, \hat{\varepsilon_z})$, que se observa en la parte (b) de la Figura **1-10**:

$$\begin{aligned}\vec{M}_o &= \int_{\mathcal{S}} (\vec{\rho}_{oi} \times \vec{\alpha}_i \times \vec{\rho}_{oi}) dm \\ &= \int_{\mathcal{S}} (\rho\hat{\varepsilon_r} \times \alpha\hat{\varepsilon_z} \times \rho\hat{\varepsilon_r}) dm \\ &= \vec{\alpha} \int_{\mathcal{S}} \rho^2 dm \end{aligned} \tag{1-21}$$

donde ρ es la distancia que hay de o a cada punto i. La integral $\int_{\mathcal{S}} r^2 dm$ se conoce como momento de inercia I_o del sólido:

$$I_o = \int_{\mathcal{S}} r^2 dm \tag{1-22}$$

En un bloque rectangular es sencillo calcular el valor de I_o. Para ello se realiza la integral 1-22 en coordenadas cartesianas $(\hat{x}, \hat{y}, \hat{z})$, considerando que $dm = \gamma dV$, donde γ es la densidad de masa del bloque y dV es el diferencial de volumen ($dV = dxdydz$). La posición ρ de cada punto en $\mathcal{S}$ varía en el plano $\hat{x}\hat{y}$ y, en consecuencia, la integral se convierte en una integral triple con límites definidos por las dimensiones del bloque: L_x, L_y y L_z. El procedimiento de integración se presenta en el Anexo A.2 y da como resultado la siguiente ecuación:

$$I_o = \frac{1}{12} m(L_x^2 + L_y^2) + m|\vec{\rho}_{oc}|^2 \tag{1-23}$$

Donde m es la masa del bloque $\mathcal{S}$.

Remplazando la Ecuación 1-23 en 1-21 se tiene:

$$\vec{M}_o = \vec{\alpha} I_o \tag{1-24}$$

Esta ecuación permite establecer la relación del movimiento de un bloque con las fuerzas que actúan en él. Para obtener el cambio en el giro, en cada paso de tiempo Δt, se despeja la aceleración angular $\vec{\alpha}$ de 1-24 y se realiza una doble integral en el intervalo $[t, t + \Delta t]$. Despejando $\vec{\alpha}$ tenemos:

$$\vec{\alpha} = \frac{\vec{M}_o}{I_o} = \frac{d\vec{\omega}}{dt} = \frac{d^2\vec{\theta}}{dt^2} \tag{1-25}$$

Al separar variables para la velocidad angular $\vec{\omega}$ e integrando entre $[t, t + \Delta t]$ tenemos:

$$\begin{aligned}\int_{\vec{\omega}_t}^{\vec{\omega}_{t+\Delta t}} d\omega &= \int_t^{t+\Delta t} \frac{\vec{M}_o}{I_o} dt \\ \vec{\omega}_{t+\Delta t} - \vec{\omega}_t &= \Delta t \frac{\vec{M}_o}{I_o} \\ \vec{\omega}_{t+\Delta t} &= \vec{\omega}_t + \Delta t \frac{\vec{M}_o}{I_o} \end{aligned} \tag{1-26}$$

Del mismo modo, como la velocidad angular es $\vec{\omega} = d\vec{\theta}/dt$, la Ecuación 1-26 se reescribe como:

$$\frac{d\vec{\theta}}{dt} = \vec{\omega}_t + \Delta t \frac{\vec{M}_o}{I_o} \tag{1-27}$$

y al integrar por partes es posible establecer una relación entre el giro del bloque en el tiempo t y $t + \Delta t$ así:

$$\begin{aligned}
\int_{\vec{\theta}_t}^{\vec{\theta}_{t+\Delta t}} d\theta &= \int_{t_i}^{t_i+\Delta t} (\vec{\omega}_t + \Delta t \frac{\vec{M}_o}{I_o}) dt \\
\vec{\theta}_{t+dt} - \vec{\theta}_t &= \vec{\omega}_t \Delta t + \frac{1}{2} \frac{\vec{M}_o}{I_o} \Delta t^2 \\
\vec{\theta}_{t+dt} &= \vec{\theta}_t + \vec{\omega}_t \Delta t + \frac{1}{2} \frac{\vec{M}_o}{I_o} \Delta t^2
\end{aligned} \tag{1-28}$$

La Ecuación 1-28 permite calcular el incremento en el giro del bloque en pequeños pasos de tiempo Δt.

1.2.2.3. Solución por el Método DEM

La solución del problema por medio del ciclo DEM, inicia al conocer el giro θ en el tiempo inicial del problema t_0. En este tiempo se calcula el momento de fuerzas por medio de la la Ecuación 1-16. Luego se calcula la velocidad angular y el giro en el siguiente instante de tiempo por medio de las Ecuaciones 1-26 y 1-28. Una vez se conozca el nuevo ángulo de giro, se incrementa el tiempo de análisis en Δt y se repite el ciclo para el nuevo tiempo. El problema de un bloque rotando tiene una solución analítica sencilla y, quizás, no vale la pena resolverlo en pequeños incrementos de tiempo por el método DEM. Sin embargo, cuando el problema analizado incluye es un sistema de bloques, cuyos elementos tienen la posibilidad de girar, trasladarse y colisionar entre sí; el DEM se convierte en una herramienta indispensable.

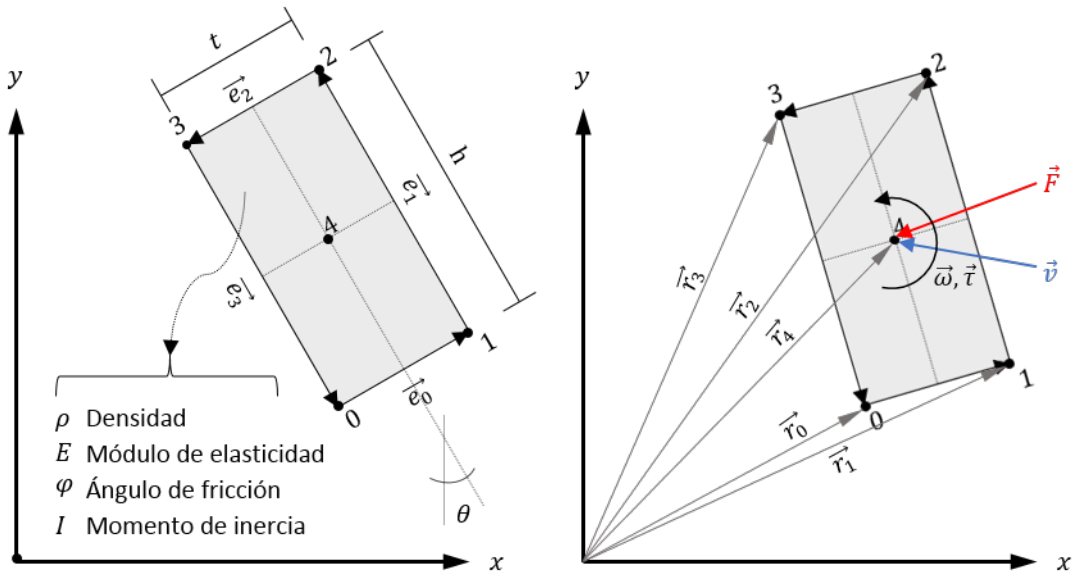
En el volteo de rocas, el número de posibilidades y formas de colisión, hace imposible analizar analíticamente lo que ocurre durante el movimiento de los bloques de roca. Por este motivo, el proceso dinámico de *toppling*, se analiza por medio del método DEM. Este proceso cíclico, que marcha en el tiempo, y utiliza herramientas computacionales para su desarrollo. En la siguiente sección se presenta un preámbulo al programa *Toppling_UN*, que se creó en este Trabajo Final, allí se dan las bases que se usan en su programación. Luego, en el Capítulo 2, se describe la mecánica de un sistema de bloques que tiene la posibilidad de rotar, deslizar y colisionar entre sí.

1.2.3. Preámbulo al programa "Toppling_UN"

El programa *Toppling_UN* se diseñó con base en el paradigma de la Programación Orientada a Objetos (POO). La programación POO tiene su origen en la programación estructurada, que se basa en rutinas y subrutinas creadas con ciclos y bifultaciones [23]. Lo que la hace diferente, es que la programación POO adiciona el concepto de *Clase* y *Objeto*, con lo cual se simplifica el manejo de la información del elemento discreto en un programa DEM . Las clases son un tipo de dato creado por el programador con el objetivo de agrupar un conjunto de elementos con las misma características. Los elementos de una clase se denominan objetos y tiene una identidad que los diferencia de los demás -ningún bloque de un sistema se comporta igual que otro-. En el programa *Toppling_UN* se diseñó una Clase llamada "Block" y los objetos de esta Clase son los bloques del talud rocoso.

Cada bloque tiene 4 vértices y 4 bordes que se numeran desde 0 en sentido contrario al giro de las manecillas de un reloj como se muestra en la parte (a) Figura **1-11**. Adicionalmente, se incluyó un vértice virtual que corresponde al centro de masa. En el programa los bordes se diseñaron como vectores denominados $\vec{e_i}$. Cada bloque cuenta con unos atributos que reflejan las propiedades de la roca: densidad γ, modulo de elasticidad k, ángulo de fricción en las paredes ϕ y momento de inercia I que depende de la masa y la geometría. En la parte (b) de la misma Figura, se observa que la posición de cada vértice se define por medio de unos vectores $\vec{r_i}$, referidos al marco de referencia inercial XY. Además, en el centro de masa de cada bloque siempre están actuando 4 vectores que representan la velocidad lineal $\vec{v}$, la velocidad angular $\vec{\omega}$, la fuerza neta $\vec{F}$ y el momento de la fuerza $\vec{\tau}$ que actúa sobre el bloque en un determinado instante.

Además de estos atributos, el bloque también cuenta con unos procedimientos incorporados que se listan en la Tabla **1-2**. Los procedimientos son funciones que se definen para desarrollar un determinado problema. En POO estos procedimientos se conocen como métodos. El código que contiene la clase "Block" se presenta en el Anexo B.2. Allí, además de esta clase, se encuentra una segunda clase denominada "Link" que cumple la función de modelar la interacción entre los bloques. En el siguiente capítulo se describe en detalle la mecánica de los bloques; esto incluye el mecanismo con el cual se identifican las colisiones, el modelo reológico que se utiliza para simular las fuerzas de choques y la solución numérica a las ecuaciones de movimiento para un bloque rígido.



(a) (b)

Figura 1-11.: Atributos del elemento discreto

Tabla 1-2.: Métodos de la Clase Block

Nombre	Tipo	Descripción
Inice	Método	Carga las condiciones iniciales del bloque
Dibujese	Método	Imprime en un archivo de texto plano ".gnu" las coordenadas de los vértices del bloque
Mover_r	Método	Calcula el movimiento de traslación en el centro de masa
Mover_θ	Método	Calcula el movimiento de rotación en el centro de masa
Agregar_F	Método	Añade la resultante de fuerzas del tiempo t_i al balance de fuerza
Agregar_τ	Método	Añade la resultante de momento del tiempo t_i al balance de momentos
Actualizar_V	Método	Actualiza la nueva velocidad lineal en el tiempo $t_i + \Delta t$
Actualizar_ω	Método	Actualiza la nueva velocidad angular en el tiempo $t_i + \Delta t$

2. Mecánica del volteo de bloques en Toppling_UN

En este capítulo se presentará el método utilizado para solucionar el problema de movimiento de un sistema de bloques, sometido a fuerzas externas y fuerzas internas. A estas últimas las denominamos fuerzas de choque y están presentes durante el tiempo que dura la colisión entre bloques. La mecánica de bloques en un programa tipo DEM depende de la manera como se identifiquen los choques, del modelo de fuerzas que se empleé y de la formulación dinámica que se seleccione para describir el movimiento. En el método DEM, se considera que una pareja de bloques choca cuando existe superposición entre ellos, tal como se presenta en la Figura **2-1**.

La superposición se mide por medio de una longitud de penetración, denominada 'd', que en ningún momento es superior a una micra. La longitud 'd' se mide desde el vértice del bloque i hasta el punto donde se inició el choque sobre el borde del bloque j, como se muestra en la misma figura. La pareja de bloques, 'i' y 'j' -donde i ejerce como bloque "actuante" y j como "receptor" del impacto-, conserva la condición de contacto durante un corto periodo de tiempo, que puede abarcar varios microsegundos, hasta que una fuerza repulsiva finaliza la superposición. Las fuerzas de repulsión son las mismas a las que se han denominado como fuerzas de choque.

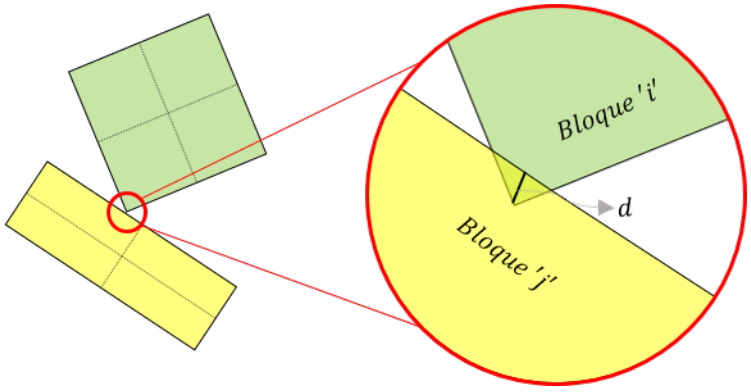


Figura 2-1.: Contacto entre dos bloques

Las fuerzas de choque están presentes únicamente durante el periodo que dura la colisión y se estudian a través de un modelo de fuerzas normal y otro tangencial. Estos modelos se simulan por medio de unidades reológicas. Además de las fuerzas de choque, en los bloques actúa la fuerza debida a su peso y, en consecuencia, la Fuerza Neta que actúa en cada bloque es la suma vectorial entre la Fuerza gravitacional y la Fuerza de Choque. En el programa *Toppling_UN*, la componente normal de las fuerzas de choque se modela con la unidad reológica de Kelvin-Voig [24, Cap 5], la cual simula un comportamiento viscoelático usando la combinación de un resorte y un amortiguador conectados en forma paralela. Por otra parte, la componente tangencial de las fuerzas de choque se modela por medio de la unidad reológica Elástico Plástico Perfecto, que consisten en un resorte conectado a un elemento plástico que limita la deformación del resorte. El peso se calcula simplemente como el producto de la gravedad y la masa del bloque.

En cuanto a las ecuaciones que describen el movimiento, en el programa *Toppling_UN* se recurre a la dinámica de sólido rígido, específicamente a la formulación de momento angular. Este trabajo se desarrolla en 2 dimensiones, con lo cual cada bloque tiene 3 grados de libertad: traslación en X, traslación en Y y rotación en el plano XY. Donde X y Y son los ejes cartesianos de un sistema de referencia inercial. La principal ventaja de trabajar con la suposición de cuerpo rígido es que el movimiento de un bloque se puede estudiar a partir de la superposición de los movimientos de traslación y rotación.

En este capítulo se desarrollan las ecuaciones que describen el movimiento de cada bloque, se discute el procedimiento para identificar choques y se presenta la modelación de las fuerzas de choque.

2.1. Ecuaciones de movimiento de los bloques

En la mecánica clásica existen 3 maneras de aplicar la segunda Ley de Newton para resolver problemas dinámicos: 1)Por la formulación de impulso y cantidad de movimiento; 2)Por el teorema de trabajo y energía; y 3)por la formulación de momento angular. En el programa *Toppling_UN* se utilizó la formulación de momento angular. En esta sección se analiza la condición en la cual un bloque, que pertenece a un sistema de bloques, experimenta desplazamientos por traslación y por rotación en forma simultanea. La sección de divide en 3 partes. En la primera parte se desarrolla la formulación cinemática de un bloque rígido; luego, en la segunda parte, se aplica la formulación de Momento Angular para relacionar las fuerzas que actúan en el bloque con sus características de movimiento. Por ultimo, en la tercera parte, se presenta la solución numérica de las ecuaciones.

2.1.1. Cinemática de un sólido rígido

Una de las principales características del análisis de un bloque rígido, es que cada punto que pertenece a un bloque conserva su posición relativa respecto a los demás puntos del bloque. Considérese un bloque rígido $\mathcal{S}$, ubicado en un sistema de referencia inercial $(\hat{x}, \hat{y})$, como el que se presenta en la Figura **2-2**(a). En el tiempo t el bloque $\mathcal{S}$ se encuentra rotando alrededor del punto 'o' con una velocidad angular ω; en el mismo tiempo, el punto o se desplaza con una velocidad $\vec{v}$. Nótese que el punto 'o' pertenece a $\mathcal{S}$ y debido a que el bloque no se deforma, cualquier punto 'i' sobre $\mathcal{S}$ conservará su posición relativa $\vec{\rho}_{oi}$ en todo instante de tiempo. La posición de 'o' es $\vec{r_o}$ y la posición del punto 'i' es $\vec{r_i}$. De esta manera se puede decir que:

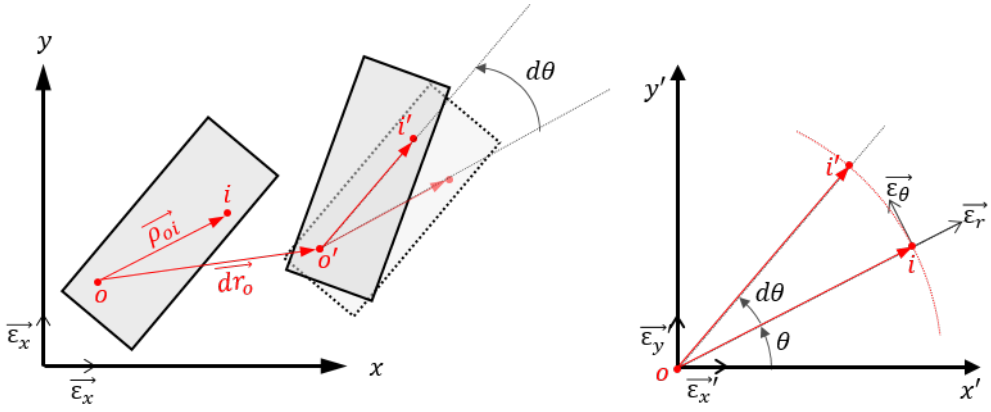
$$\vec{r_i} = \vec{r_o} + \vec{\rho}_{oi} \tag{2-1}$$

Un instante de tiempo después, en el tiempo $t + dt$, la pareja de puntos 'i' y 'o' se desplazan a una nueva posición $\vec{r_i}$' y $\vec{r_o}$'. Considérese un segundo sistema de referencia $(\hat{x}', \hat{y}')$ que se moviliza en conjunto con el punto o, como se muestra en la Figura **2-2**(b). En este sistema coordenado se forma un ángulo θ entre $\hat{x}$ y $\vec{\rho}_{oi}$. Ahora, el vector $\vec{\rho}_{oi}$ se calcula como:

$$\vec{\rho}_{oi} = \rho\cos\theta\hat{x} + \rho\sin\theta\hat{y} \tag{2-2}$$

En consecuencia, la posición del punto i e i' será:

$$\vec{r_i} = (x_o + \rho\cos\theta)\hat{x} + (y_o + \rho\sin\theta)\hat{y} \tag{2-3}$$
$$\vec{r_{i'}} = (x_{o'} + \rho\cos(\theta + d\theta))\hat{x} + (y_{o'} + \rho\sin(\theta + d\theta))\hat{y} \tag{2-4}$$



(a) Desplazamiento de los puntos i y o (b) Rotación del punto i

Figura 2-2.: Movimiento general de un sólido rígido

El desplazamiento que experimenta el punto i, durante en el intervalo $[t, t+dt]$, es la diferencia entre las Ecuaciones 2-4 y 2-3.

$$\begin{aligned} \vec{dr}_i &= \vec{r}_{i'} - \vec{r}_i \\ &= (x_{o'} - x_o + \rho\cos(\theta + d\theta) - \rho\cos\theta)\hat{x} + (y_{o'} - y_o + \rho\sin(\theta + d\theta) - \rho\sin\theta)\hat{y} \end{aligned} \tag{2-5}$$

De forma similar, el desplazamiento del punto o es

$$\begin{aligned} \vec{dr}_o &= \vec{r}_{o'} - \vec{r}_o \\ &= (x_{o'} - x_o)\hat{x} + (y_{o'} - y_o)\hat{y} \end{aligned} \tag{2-6}$$

Remplazando la Ecuación 2-6 en 2-5 tenemos:

$$\vec{dr}_i = ((dr_o)_x + \rho\cos(\theta + d\theta) - \rho\cos\theta)\hat{x} + ((dr_o)_y + \rho\sin(\theta + d\theta) - \rho\sin\theta)\hat{y} \tag{2-7}$$

El seno y el coseno de la suma de dos ángulo es:

$$\sin(a + b) = \sin(a)\cos(b) + \cos(a)\sin(b) \tag{2-8}$$
$$\cos(a + b) = \cos(a)\cos(b) - \sin(a)\sin(b) \tag{2-9}$$

entonces la Ecuación 2-7 se queda como:

$$\begin{aligned} \vec{dr}_i = ((dr_o)_x + \rho(\cos\theta\cos d\theta - \sin\theta\sin d\theta - \cos\theta))\hat{x} \\ + ((dr_o)_y + \rho(\sin\theta\cos d\theta + \cos\theta\sin d\theta - \sin\theta))\hat{y} \end{aligned} \tag{2-10}$$

En un programa tipo DEM, el diferencial de giro $d\theta$ es muy pequeño ya que se calcula para intervalos de tiempo inferiores a 10^{-5} segundos[1]. Por esta razón es válido asumir que $\cos d\theta = 1$ y $\sin d\theta = d\theta$, incurriendo en un error de redondeo que es despreciable[2]. Con esta simplificación la expresión 2-10 se reduce a:

$$\vec{dr}_i = ((dr_o)_x - \rho d\theta\sin\theta)\hat{x} + ((dr_o)_y + \rho d\theta\cos\theta)\hat{y} \tag{2-11}$$

En el sistema de referencia $\mathcal{A}'$, Figura **2-2**(b), existe un vector unitario $\hat{\theta}$ que se puede expresar como:

$$\hat{\theta} = -\sin\theta\hat{x} + \cos\theta\hat{y} \tag{2-12}$$

[1] Así que definió en la sección 3.1.1: calibración del paso de tiempo Δt

[2] Por ejemplo, cuando $d\theta$ es 0,01°, $\cos d\theta$ = 0,999999985 y asumir que $\cos d\theta = 1$ produce un error de redondeo de $1{,}52 \times 10^{-8}$; por otra parte asumir que $\sin d\theta = d\theta$ produce un error de $8{,}86 \times 10^{-13}$. En el programa $d\theta$ nunca es mayor 0,01°.

Al remplazar la Ecuación 2-12 en 2-11 y como el ángulo $d\theta$ se puede representar con un vector $\vec{d\theta}$ perpendicular a $\hat{x}$ y $\hat{y}$, tenemos:

$$\begin{aligned} \vec{dr}_i &= (dr_o)_x\hat{x} + (dr_o)_y\hat{y} + \rho d\theta\hat{\theta} \\ &= \vec{dr}_o + \vec{d\theta} \times \vec{\rho}_{oi} \end{aligned} \tag{2-13}$$

La ecuación 2-13 define el desplazamiento de cualquier punto i contenido en el bloque $\mathcal{S}$. Con base en esta ecuación se calcula la velocidad con la que se desplaza un punto i, para ello se deriva en el tiempo la posición $\vec{r_i}$ de la Ecuación 2-1 .

$$\begin{aligned} \vec{v_i} &= \frac{d\vec{r_i}}{dt} = \frac{d\vec{r_o}}{dt} + \frac{d}{dt}\vec{\rho}_{oi} \\ &= \vec{v_o} + \frac{d}{dt}\vec{\rho}_{oi} \end{aligned} \tag{2-14}$$

Para resolver la derivada $d\vec{\rho}_{oi}/dt$ se emplea el sistema de referencia $\mathcal{A}'$, con lo cual tenemos:

$$\frac{d}{dt}\vec{\rho}_{oi} = \frac{d}{dt}\rho\cos\theta\hat{x} + \frac{d}{dt}\rho\sin\theta\hat{y} \tag{2-15}$$

Como ρ es la distancia entre los puntos i y o, y dado que ambos puntos pertenecen al bloque $\mathcal{S}$, ρ no cambia en el tiempo. Derivando y definiendo la velocidad angular con la que rota i respecto a o como $\omega = d\theta/dt$ tenemos:

$$\begin{aligned} \frac{d}{dt}\vec{\rho}_{oi} &= -\rho\sin\theta\frac{d\theta}{dt}\hat{x} + \rho\cos\theta\frac{d\theta}{dt}\hat{y} \\ &= \rho\omega(-\sin\theta\hat{x} + \cos\theta\hat{y}) \end{aligned} \tag{2-16}$$

Recordando que en la Ecuación 2-12 se definió un vector $\hat{\theta}$ y dado que la velocidad angular ω se puede representar con un vector $\vec{\omega_o}$ que va en dirección $\hat{z} = \hat{x} \times \hat{y}$ tenemos que

$$\begin{aligned} \frac{d}{dt}\vec{\rho}_{oi} &= \rho\omega\hat{\theta} \\ &= \vec{\omega_o} \times \vec{\rho}_{oi} \end{aligned} \tag{2-17}$$

Remplazando la Ecuación 2-17 en 2-14 tenemos

$$\vec{v_i} = \vec{v_o} + \vec{\omega_o} \times \vec{\rho}_{oi} \tag{2-18}$$

La ecuación 2-18 define la velocidad de un punto i para cualquier instante de tiempo. Con base en esta ecuación se calcula la aceleración, para ello se deriva $\vec{v_i}$ de la ecuación 2-18.

$$\begin{aligned} \vec{a_i} &= \frac{d\vec{v_i}}{dt} = \frac{d\vec{v_o}}{dt} + \frac{d}{dt}(\vec{\omega_o} \times \vec{\rho}_{oi}) \\ &= \vec{a_o} + \frac{d}{dt}\vec{\omega_o} \times \vec{\rho}_{oi} + \vec{\omega_o} \times \frac{d}{dt}\vec{\rho}_{oi} \end{aligned} \tag{2-19}$$

Definiendo la aceleración angular como el vector $\vec{\alpha_o} = d\omega_o/dt$ y remplazando la Ecuación 2-17 en 2-19

$$\vec{a_i} = \vec{a_o} + \vec{\alpha_o} \times \vec{\rho}_{oi} + \vec{\omega_o} \times (\vec{\omega_o} \times \vec{\rho}_{oi}) \tag{2-20}$$

Las ecuaciones 2-13, 2-18 y 2-20 son las que se utilizan para analizar el movimiento del sólido $\mathcal{S}$. Sólo falta definir una ecuación que relacione las Fuerzas con el movimiento. Para ello se recurre a la definición de Momento Angular.

2.1.2. Dinámica de un sólido rígido

Considérese un sólido rígido $\mathcal{S}$, sometido a un conjunto de n fuerzas f_i que actúan sobre cada punto i de $\mathcal{S}$, como se ilustra en la figura **2-3**. La fuerza f_i es la resultante de todas las fuerzas que actúan sobre el punto i. Cada punto i del sólido ocupa un volumen infinitesimal denotado como dV y tiene una masa diferencial dm proporcional a su densidad, $dm = \gamma dV$. La fuerza total que que actua sobre el bloque $\mathcal{S}$ es la suma vectorial de todas las fuerzas sobre el.

$$\vec{F} = \sum_{i=1}^{n} \vec{f_i} \tag{2-21}$$

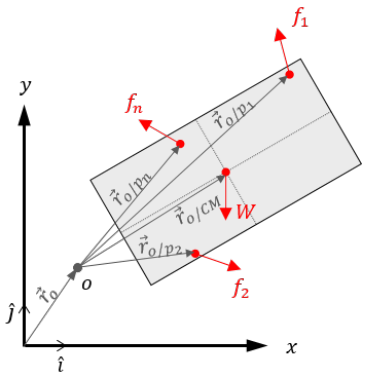


Figura 2-3.: Fuerzas actuantes sobre un sólido rígido

La aceleración $\vec{a_i}$ de i se obtiene directamente de la segunda ley de Newton.

$$\vec{f_i} = m_i \frac{d^2}{dt^2}(\vec{r_i}) = m_i \vec{a_i} \tag{2-22}$$

Por tanto, al remplazar 2-22 en 2-21 tenemos:

$$\vec{F} = \int_{\mathcal{S}} \frac{d^2}{dt^2}(\vec{r}) dm \tag{2-23}$$

Como la masa es invariante en el tiempo la ecuación se reescribe como:

$$\vec{F} = \frac{d^2}{dt^2}\int_{\mathcal{S}}(\vec{r})dm \tag{2-24}$$

El centro de masa de un bloque rígido se calcula como:

$$\vec{r_c} = \frac{\int_{\mathcal{S}} r dm}{\int_{\mathcal{S}} dm} = \frac{\int_{\mathcal{S}} r dm}{m} \tag{2-25}$$

Donde m es la masa del bloque $\mathcal{S}$. Despejando $\int_{\mathcal{S}} r dm$ de 2-25 y remplazando en 2-24 tenemos:

$$\vec{F} = \frac{d^2}{dt^2} M\vec{r_c} \tag{2-26}$$

Derivando se obtiene la relación entre la fuerza total sobre $\mathcal{S}$ en función de la aceleración del centro de masa.

$$\vec{F} = M\vec{a_c} \tag{2-27}$$

Por otra parte, dado que todos los punto de $\mathcal{S}$ giran respecto a un punto o, cada una de las fuerzas f_i generan un torque respecto a o que se calcula como:

$$\vec{\tau_{oi}} \;=\; \vec{\rho_{oi}} \times \vec{f_i} = (\vec{r_i} - \vec{r_o}) \times \vec{f_i} \tag{2-28}$$

El momento de fuerza resultante para el bloque $\mathcal{S}$ se calcula como la suma vectorial del torque producido por cada una de la fuerza f_i

$$\vec{M_o} = \sum_{i=1}^{n} \vec{\tau_{oi}} \tag{2-29}$$

La fuerza f_i es igual al producto de a_i por m_i, entonces al remplazar 2-28 en 2-29 y realizando la suma en forma de integral en todo el sólido $\mathcal{S}$ tenemos:

$$\vec{M_o} = \int_{\mathcal{S}}((\vec{r_i} - \vec{r_o}) \times \vec{a_i})dm \tag{2-30}$$

Recordando que en la ecuación 2-20 se definió que : $\vec{a_i} = \vec{a_o} + \vec{\alpha_o} \times \vec{\rho_{oi}} + \vec{\omega_o} \times (\vec{\omega_o} \times \vec{\rho_{oi}})$. Tenemos que 2-30 se puede reescribir como:

$$\vec{M_o} = \int_{\mathcal{S}}(\vec{r_i} - \vec{r_o}) \times \vec{a_o} dm + \int_{\mathcal{S}}(\vec{r_i} - \vec{r_o}) \times (\vec{\alpha_o} \times (\vec{r_i} - \vec{r_o}))dm + \int_{\mathcal{S}}(\vec{r_i} - \vec{r_o}) \times (\vec{\omega_o} \times (\vec{\omega_o} \times (\vec{r_i} - \vec{r_o}))dm \tag{2-31}$$

Esta Ecuación es muy importante porque nos relaciona el momento en el sólido con su aceleración. A continuación se va a desarrollar cada una de las tres integrales.

Reescribiendo la tercera integral de 2-31 en términos de sus vectores directores se tiene que

$$\int_{\mathcal{S}} (\vec{r_i} - \vec{r_o}) \times (\vec{\omega_o} \times (\vec{\omega_o} \times (\vec{r_i} - \vec{r_o}))dm = \int_{\mathcal{S}} |\vec{r_i} - \vec{r_o}|\hat{r} \times (|\vec{\omega_o}|\hat{z} \times (|\vec{\omega_o}|\hat{z} \times |\vec{r_i} - \vec{r_o}|\hat{r}))dm \quad (2\text{-}32)$$

sin embargo, nótese que

$$\hat{r} \times (\hat{z} \times (\hat{z} \times \hat{r})) = \hat{r} \times (\hat{z} \times \hat{\theta}) = \hat{r} \times (-\hat{r}) = 0 \quad (2\text{-}33)$$

por lo tanto la Ecuación 2-32 queda como:

$$\int_{\mathcal{S}} (\vec{r_i} - \vec{r_o}) \times (\vec{\omega_o} \times (\vec{\omega_o} \times (\vec{r_i} - \vec{r_o}))dm = 0 \quad (2\text{-}34)$$

y remplazando la Ecuación 2-34 en 2-31 tenemos:

$$\vec{M_o} = \int_{\mathcal{S}} (\vec{r_i} - \vec{r_o}) \times \vec{a_o} dm + \int_{\mathcal{S}} (\vec{r_i} - \vec{r_o}) \times (\vec{\alpha_o} \times (\vec{r_i} - \vec{r_o}))dm \quad (2\text{-}35)$$

Reescribiendo la segunda integral de la Ecuación 2-35 en términos de sus vectores directores, se tiene:

$$\int_{\mathcal{S}} (\vec{r_i} - \vec{r_o}) \times (\vec{\alpha_o} \times (\vec{r_i} - \vec{r_o}))dm = \int_{\mathcal{S}} (|\vec{r} - \vec{r_o}|\hat{r} \times (\alpha_o \hat{z} \times |\vec{r} - \vec{r_o}|\hat{r}))dm \quad (2\text{-}36)$$

pero el producto $\hat{r} \times (\hat{z} \times \hat{r})$ da como resultado $\hat{z}$, con lo cual la ecuación 2-36 queda como:

$$\begin{aligned} \int_{\mathcal{S}} (\vec{r_i} - \vec{r_o}) \times (\vec{\alpha_o} \times (\vec{r_i} - \vec{r_o}))dm &= \int_{\mathcal{S}} (|\vec{r} - \vec{r_o}|^2 \alpha_o \hat{z})dm \\ &= \vec{\alpha_o} \int_{\mathcal{S}} |\vec{r} - \vec{r_o}|^2 dm \end{aligned} \quad (2\text{-}37)$$

donde la integral $\int_{\mathcal{S}} |\vec{r} - \vec{r_o}|^2 dm$ se conoce como la inercia, I_o, del sólido girando alrededor del punto o. Esta expresión se presentó con anterioridad en la Ecuación 1-22. Allí se explicó el procedimiento de calculo y se concluyo que I_o es:

$$I_o = \int_{\mathcal{S}} |\vec{r} - \vec{r_o}|^2 dm \quad (2\text{-}38)$$

El procedimiento de integración se presenta en el Anexo A.2. Con base en esto, remplazando la Ecuación 2-38 en 2-37 y luego remplazando el resultado en 2-35 se tiene:

$$\vec{M_o} = \int_{\mathcal{S}} (\vec{r_i} - \vec{r_o}) \times \vec{a_o} dm + \vec{\alpha_o} I_o \quad (2\text{-}39)$$

En la Ecuación 2-39 el valor de $(\vec{r_i} - \vec{r_o})$ y de $\vec{a_o}$ no cambia ya que los puntos i y o conservan su distancia y la aceleración de o es única para ese punto; es decir, $\vec{a_o}$ no depende de dm. Con todo esto, la integral de la Ecuación 2-39 se puede reescribir como:

$$\int_{\mathcal{S}} (\vec{r_i} - \vec{r_o}) \times \vec{a_o} dm = (\vec{r_i} - \vec{r_o}) \times \vec{a_o} \int_{\mathcal{S}} dm = m(\vec{r_i} - \vec{r_o}) \times \vec{a_o} \tag{2-40}$$

Remplazando 2-40 en 2-39 se obtiene la ecuación que relaciona el torque con la aceleración angular que experimenta $\mathcal{S}$. La ecuación 2-31 se ha simplificado a:

$$\vec{M_o} = m(\vec{r_i} - \vec{r_o}) \times \vec{a_o} + \vec{\alpha_o} I_o \tag{2-41}$$

Cuando el bloque gira alrededor de su centro de masa, un punto denotado como c, el punto o pasa a ser c. Además, en el centro de masa se puede estudiar el movimiento del bloque, así que se puede seleccionar el punto i en c. Con esto, el valor de $(\vec{r_i} - \vec{r_o})$ es 0 y la Ecuación 2-41 se reduce a:

$$\vec{M_c} = \vec{\alpha_c} I_c \tag{2-42}$$

En conclusión, un conjunto de n fuerzas que actúan sobre un sólido rígido, generan un desplazamiento debido a translación y rotación. Este movimiento está gobernado por las ecuaciones 2-27 y 2-42. Excepto cuando el bloque gira en un punto arbitrario que no sea el centro de masa. En ese caso se debe usar la Ecuación 2-27 y la 2-41.

En el programa *Topplin_UN* la rotación se calcula en el centro de masa. En consecuencia, y de acuerdo con las Ecuaciones 2-27 y 2-42, la aceleración lineal y la aceleración angular de un bloque son:

$$\vec{a_c} = \frac{\vec{F}}{m} \tag{2-43}$$

$$\vec{\alpha_c} = \frac{\vec{M_c}}{I_c} \tag{2-44}$$

La aplicación de estas ecuaciones en el programa *Toppling_UN* se realiza en forma numérica, por esta razón en la siguiente sección se explican algunos métodos comunes que se emplean para ello.

2.1.3. Solución numérica del sistema de ecuaciones

Las ecuaciones de movimiento se implementan por medio de un algoritmo que permite establecer la posición y la orientación de un bloque en el tiempo $t_{i+1} = t_i + \Delta t$. En el algoritmo también se calculan las variables de velocidad, aceleración y fuerza en el mismo tiempo. El

procedimiento por el cual se calculan las variables cinemáticas en un programa de computador, se conoce como algoritmo de integración [11]. Los algoritmos más populares se fundamentan en el uso de la Serie de Taylor. En la actualidad existe una amplia variedad de algoritmos de integración, aquí se consideran 4 con el propósito de seleccionar el que mejor represente las características de un proceso de *toppling*.

La Serie de Taylor es una serie de potencias que se emplea para calcular el valor de una función $f(t)$ a partir de un punto conocido t_i, donde la función es infinitamente diferenciable y se conocen todas las derivadas en t_i [25]. La Serie de Taylor es de la forma

$$f(t) = \frac{f(t_i)}{0!} + \frac{f'(t_i)}{1!}(t-t_i)^1 + \frac{f''(t_i)}{2!}(t-t_i)^2 + \ldots = \sum_{n=0}^{\infty} \frac{f^n(t_i)}{n!}(t-t_i)^n \tag{2-45}$$

donde la distancia que existe entre los puntos t y t_i se puede considerar como el paso de tiempo Δt. La ecuación 2-45 se puede reescribir como

$$f(t_i+\Delta t) = \frac{f(t_i)}{0!} + \frac{f'(t_i)}{1!}(\Delta t)^1 + \frac{f''(t_i)}{2!}(\Delta t)^2 + \ldots = \sum_{n=0}^{\infty} \frac{f^n(t_i)}{n!}(\Delta t)^n \tag{2-46}$$

Esta ecuación es del tipo progresivo, ya que calcula $f(t)$ en un punto $t_i + \Delta t$ por delante de t_i. Si por el contrario se quisiera calcular la función en un punto anterior a t_i basta con reescribir la ecuación así:

$$f(t_i-\Delta t) = \frac{f(t_i)}{0!} - \frac{f'(t_i)}{1!}(\Delta t)^1 + \frac{f''(t_i)}{2!}(\Delta t)^2 + \ldots = \sum_{n=0}^{\infty} \frac{f^n(t_i)}{n!}(-\Delta t)^n \tag{2-47}$$

El objetivo de emplear series de Taylor es relacionar una función con su primera y segunda derivada. Equivalente a la relación entre posición, velocidad y aceleración. Para esto es necesario limitar la serie a un número finito de 'n' términos, ocasionando un error que se conoce como *error por truncamiento local* y cuyo orden de magnitud es $O[(\Delta t^{q-1})]$ donde q es el número de puntos, es decir: los valores conocidos que se emplean para obtener la n-ésima derivada [25].

Existen varias maneras de implementar las series de Taylor para resolver en forma numérica un problema dinámico. A continuación de describen 4 procedimientos: algoritmo de Verlet, algoritmo de Verlet modificado (Diferencias Finitas), algoritmo de velocidad de Verlet y algoritmo de leapfrog. Al final de esta sección se comparan los 4 procedimientos y se selecciona el que resulta más adecuado. Por comodidad en la nomenclatura se desarrollan las expresiones en términos de $f(t_i)$ y cuando sea necesario se asocia el valor de f a la posición r. En consecuencia f' y f'' serán la velocidad v y la aceleración a respectivamente. Esta suposición incluye, por supuesto, los efectos del movimiento de traslación y rotación.

2.1.3.1. Algoritmo de Verlet

El algoritmo de integración de Verlet surge del siguiente procedimiento. Sumando las ecuaciones 2-46 y 2-47 se tiene

$$f(t_i + \Delta t) + f(t_i - \Delta t) = 2f(t_i) + f''(t_i)(\Delta t)^2 + \frac{1}{12}f^4(t_i)(\Delta t)^4 + \dots \tag{2-48}$$

Ya que el valor de Δt debe ser muy pequeño para que la aproximación sea precisa, Δt^4 es aún más pequeño y se puede truncar la serie hasta la derivada de segundo orden $O[(\Delta t)^2]$. Despejando el valor $f(t + \Delta t)$ de la Ecuación 2-48 se tiene:

$$f(t_i + \Delta t) = 2f(t_i) - f(t_i - \Delta t) + f''(t_i)(\Delta t)^2 + O[(\Delta t)^2] \tag{2-49}$$

Nótese que $O[(\Delta t)^2]$ es el error por truncamiento de segundo orden, y no cuarto orden como se estaría tentado a pensar. Esto ocurre porque la ecuación requiere de tres puntos conocidos para que exista solución: $f(t_i)$, $f(t_i - \Delta t)$ y $f''(t_i)(\Delta t)$. El orden del error se calcula como número de puntos utilizados menos 1 [25].

La ecuación 2-49 se puede emplear para calcular la posición en el tiempo $t + \Delta t$, con base en sus dos posiciones anteriores, $f(t)$ y $f(t - \Delta t)$ y con la aceleración en el tiempo t, $f''(t)$. Lo que que quiere decir que la Ecuación 2-49 es igual a escribir:

$$r_{(t_i+\Delta t)} = 2r_{t_i} - r_{(t_i-\Delta t)} + a_{t_i}(\Delta t)^2 \tag{2-50}$$

En el algoritmo de Verlet la velocidad no influye en la trayectoria. Sin embargo, si se requiere conocer la velocidad basta con restar las ecuaciones 2-46 y 2-47.

$$f(t_i + \Delta t) - f(t_i - \Delta t) = 2f'(t_i)(\Delta t) + \frac{1}{3}f'''(t_i)(\Delta t)^3 + \dots \tag{2-51}$$

Al despejar la variable de interés $f(t_i + \Delta t)$ e ignorando los términos de la serie desde la derivada de tercer orden se tiene

$$f'(t_i) = \frac{1}{2(\Delta t)}(f(t_i + \Delta t) - f(t_i - \Delta t)) + O[(\Delta t)^1] \tag{2-52}$$

lo que es igual a

$$v_{t_i} = \frac{1}{2(\Delta t)}(r_{(t_i+\Delta t)} - r_{(t_i-\Delta t)}) \tag{2-53}$$

Las ecuaciones 2-50 y 2-53 son el fundamento del algoritmo de Verlet; con ellas se puede calcular la posición y la velocidad de un cuerpo en cualquier tiempo $t_i + \Delta t$. Sin embargo para la condición inicial, en la cual $t_i = 0$, no existe una posición previa ya que no es posible un tiempo $t_i = -\Delta t$. Este es un problema de ecuaciones diferenciales con valor inicial, para solventarlo se considera una posición virtual $f(t_i - \Delta t)$ y una velocidad virtual $f'(t_i - \Delta t)$.

Luego por medio de la ecuación 2-46, serie de Taylor progresiva, y ubicado en el tiempo virtual $t_i = -\Delta t$ se calcula la posición en $t_i = 0$. así:

$$f(0) = f(-\Delta t) + f'(-\Delta t)(\Delta t) + O[(\Delta t)^1] \tag{2-54}$$

lo que es igual a

$$r_0 = r_{(-\Delta t)} + v_{(-\Delta t)}(\Delta t) \tag{2-55}$$

El procedimiento que se debe utilizar para el algoritmo de Verlet se presenta en la figura **2-4** y se describe a continuación.

1. El análisis inicia en el tiempo virtual $t_i = -\Delta t$. Allí se conoce la posición y la velocidad.
2. Se realiza un preproceso por medio de la ecuación 2-55 donde se calcula la posición en $t_i = 0$. Se avanza una cantidad Δt, a un nuevo tiempo de análisis $t_i = 0$.
3. Se verifican los posibles contactos en el tiempo t_i y se hace el calculo de la fuerza neta y torque.
4. Se calcula la aceleración en t_i por medio de las ecuaciones de movimiento 2-43 y 2-44.
5. Se calcula la posición en $t_i + \Delta t$ por medio de la ecuación 2-50. Se avanza el tiempo de análisis una cantidad Δt y se repite el proceso desde el paso 3.
6. De manera opcional se va calculando la velocidad en cada tiempo t_i por medio de la ecuación 2-53.

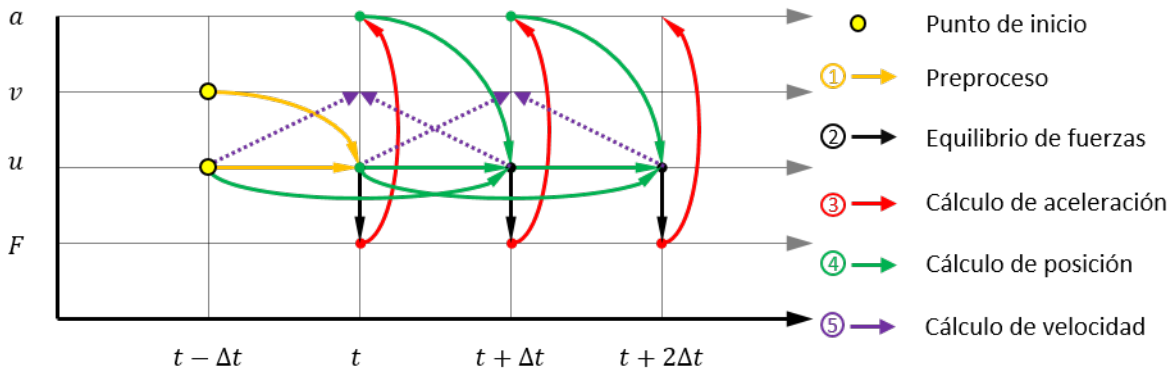


Figura 2-4.: Algoritmo de Verlet

2.1.3.2. Algoritmo de Verlet modificado (Diferencias Finitas)

Una de la principales críticas al algoritmo de Verlet es que no considera la velocidad en el cálculo de posición. Sin embargo, partiendo del algoritmo de Verlet es sencillo deducir una expresión que relacione las tres variables cinemáticas. Para ello se despeja el término $f(t_i - \Delta t)$ de la ecuación 2-52 y se remplaza en 2-49. Luego, ordenando la expresión, se obtiene que:

$$f(t_i + \Delta t) = f(t_i) + f'(t_i)(\Delta t) + \frac{1}{2}f''(t_i)(\Delta t)^2 + O[(\Delta t)^2] \tag{2-56}$$

lo que es igual a

$$r_{(t_i+\Delta t)} = r_{t_i} + v_{t_i}(\Delta t) + \frac{1}{2}a_{t_i}(\Delta t)^2 \tag{2-57}$$

Esta nueva ecuación calcula la posición en el tiempo $t = t_i + \Delta t$, a partir de la posición, la velocidad y la aceleración del tiempo t_i. No requiere el preproceso que se utilizaba en Verlet. Esto reduce los pasos del algoritmo de integración y lo hace iniciar de forma directa en un tiempo $t_i = 0$ sin recurrir a cálculos adicionales. La velocidad se obtiene por medio de la ecuación 2-46, serie de Taylor progresiva así:

$$f(t_i + \Delta t) = f(t_i) + f'(t_i)(\Delta t) + O[(\Delta t)^1] \tag{2-58}$$

lo que es igual a

$$v_{(t_i+\Delta t)} = f_{t_i} + a_{t_i}(\Delta t) \tag{2-59}$$

La figura **2-5** muestra el procedimiento de este algoritmo de integración y los pasos se describen a continuación:

1. El análisis inicia en el tiempo $t_i = 0$. Allí se conoce la posición y la velocidad.

2. En el tiempo t_i se hace el calculo de la Fuerza Neta y se calcula la aceleración de las ecuaciones 2-43 y 2-44.

3. Se calcula la velocidad en $t_i + \Delta t$ por medio de la ecuación 2-59.

4. Se calcula la posición en $t_i + \Delta t$ por medio de la ecuación 2-57. Se avanza el tiempo de análisis una cantidad Δt y se repite el proceso desde el paso 2.

Las ecuaciones que utiliza este algoritmo coinciden con el método numérico de las Diferencias Finitas.

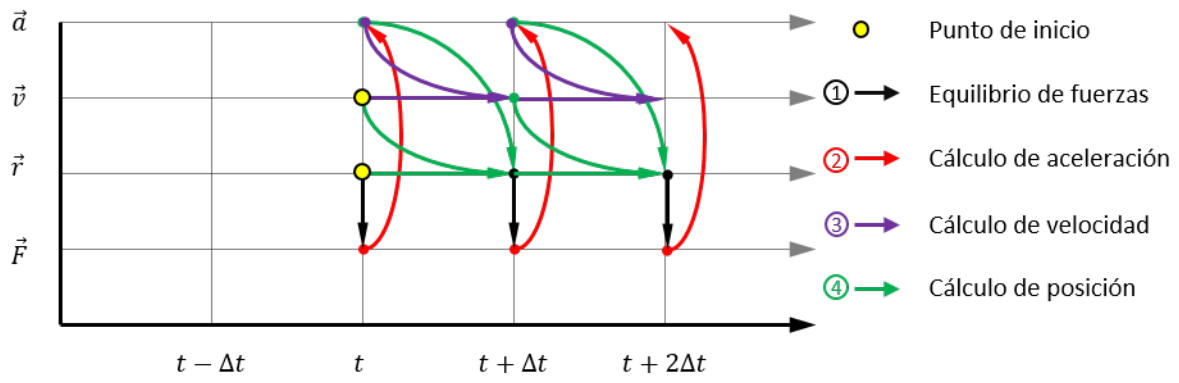


Figura 2-5.: Algoritmo de Verlet modificado (Diferencia Finitas)

2.1.3.3. Algoritmo de velocidad de Verlet

El algoritmo de velocidad de Verlet es una extensión al algoritmo de Verlet [26]. La principal diferencia con su predecesor es que refina el cálculo de la velocidad añadiendo un paso intermedio para su calculo en el instante $t = t_i + \frac{\Delta t}{2}$. Con esto el calculo de velocidad se realiza en dos etapas: La primera por medio de una serie de Taylor progresiva y la segunda por medio de una regresiva. En ambos casos de expande la serie de Taylor con una distancia entre puntos de $\frac{\Delta t}{2}$. Las ecuaciones que usa son las siguientes:

$$f(t_i + \frac{\Delta t}{2}) = f(t_i) + f'(t_i)\frac{\Delta t}{2} + O[(\Delta t)^1] \tag{2-60}$$

$$f(t_i + \Delta t) = f(t_i + \frac{\Delta t}{2}) + f'(t_i + \Delta t)\frac{\Delta t}{2} + O[(\Delta t)^1] \tag{2-61}$$

Donde, para este caso, f corresponde a la velocidad y, por consiguiente, f' a la aceleración así:

$$v_{(t_i + \frac{\Delta t}{2})} = v_{t_i} + a_{t_i}\frac{\Delta t}{2} \tag{2-62}$$

$$v_{(t_i + \Delta t)} = v_{(t_i + \frac{\Delta t}{2})} + a_{(t_i + \Delta t)}\frac{\Delta t}{2} \tag{2-63}$$

El procedimiento de cálculo se presenta en la figura **2-6** y se describe a continuación.

1. El análisis inicia en el tiempo $t_i = 0$. Allí se conoce la posición y la velocidad. En este tiempo se calcula la fuerza neta y el torque. También la aceleración por las ecuaciones 2-43 y 2-44.

2. Se calcula la velocidad en $t_i + \frac{\Delta t}{2}$ por medio de la ecuación 2-62.

3. Se calcula la posición en $t_i + \Delta t$ por medio de la ecuación 2-57.

4. En el tiempo $t_i + \Delta t$ se hace un nuevo cálculo de la fuerza neta y se calcula la aceleración por medio de las ecuaciones 2-43 y 2-44.

5. Se calcula la velocidad en $t_i + \Delta t$ por medio de la ecuación 2-63. Se avanza el tiempo de análisis una cantidad Δt y se repite el proceso desde el paso 2.

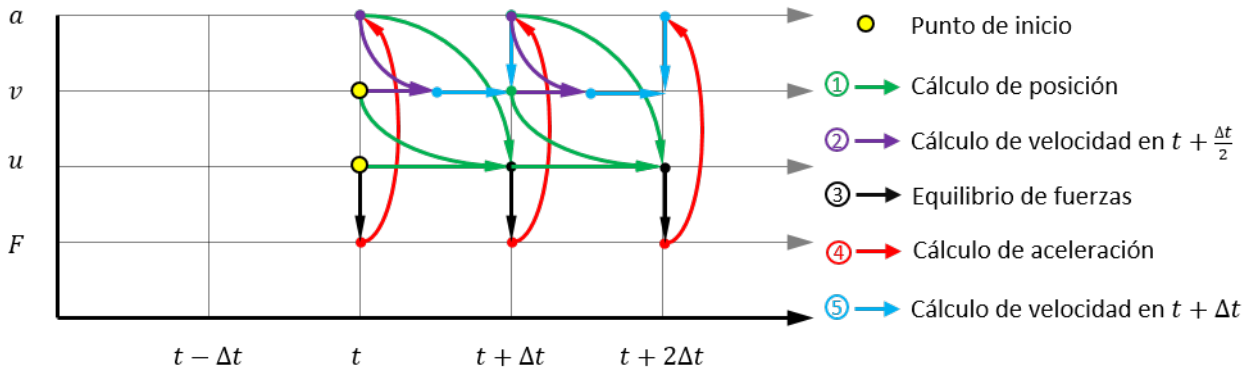


Figura 2-6.: Algoritmo velocidad de Verlet

2.1.3.4. Algoritmo leapfrog

El algoritmo de integración de leapfrog es una propuesta que varía de las anteriores ya que calcula la posición y la velocidad en instantes de tiempo desfasados una cantidad $\Delta t/2$. Las ecuaciones de posición y velocidad que utiliza para el cálculo son de tipo progresivo, con un intervalo de tiempo Δt, y tiene error de primer orden $O[(\Delta t)^1]$, ya que únicamente relacionan la función f con su primera derivada f'. La posición en $t_i + \Delta t$ se calcula con la ecuación 2-58, la velocidad con la misma expresión pero calculada en $t = t_i - \frac{\Delta t}{2}$ así:

$$f(t_i + \Delta t) = f(t_i) + f'(t_i)\Delta t + O[(\Delta t)^1] \quad (2\text{-}64)$$

$$f(t_i - \frac{\Delta t}{2} + \Delta t) = f(t_i - \frac{\Delta t}{2}) + f'(t_i)\Delta t + O[(\Delta t)^1] \quad (2\text{-}65)$$

lo que es igual a

$$r_{(t_i+\Delta t)} = r_{t_i} + v_{t_i}\Delta t \quad (2\text{-}66)$$

$$v_{(t_i+\frac{\Delta t}{2})} = v_{(t_i-\frac{\Delta t}{2})} + a_{t_i}\Delta t \quad (2\text{-}67)$$

El procedimiento de este algoritmo de integración se puede observar en la figura **2-7** y se resume a continuación:

1. El análisis inicia en el tiempo $t_i = 0$. Se conoce la posición en t_i y la velocidad en $t_i - \frac{\Delta t}{2}$.
2. En el tiempo t_i se hace el calculo de la Fuerza Neta y se calcula la aceleración por medio de las ecuaciones 2-43 y 2-44.
3. Con base en la aceleración en t_i y la velocidad en $t_i - \frac{\Delta t}{2}$, se calcula la velocidad en $t_i + \frac{\Delta t}{2}$ por medio de la ecuación 2-67.
4. Se calcula la posición en $t_i + \Delta t$ por medio de la ecuación 2-66. Se avanza el tiempo de análisis una cantidad Δt y se repite el proceso desde el paso 2.

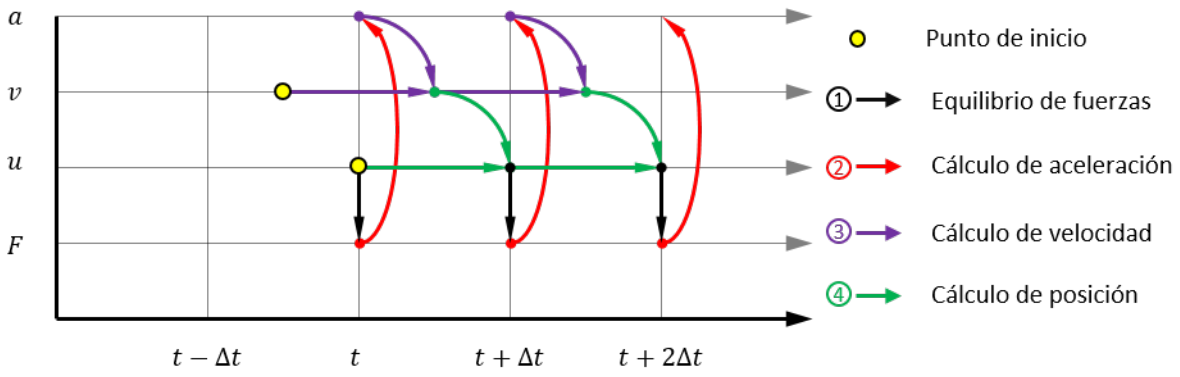


Figura 2-7.: Algoritmo leapfrog

2.1.3.5. Comparación de algoritmos

Estos algoritmos de integración se comparan por medio de un problema de bloque deslizante. Considérese un bloque de masa m, sobre una superficie inclinada un ángulo β y con un ángulo de fricción ϕ. La solución teórica del problema se obtiene a partir del equilibrio de las fuerzas en dirección normal y tangencial a la superficie:

$$N = mg\cos\beta \tag{2-68}$$

$$F_x = mg\cos\beta(\tan\beta - \tan\phi) \tag{2-69}$$

donde x es la dirección en que se desplaza el bloque. La aceleración en dirección x (a_x), se despeja de la Segunda Ley de Newton. La velocidad v_x y el desplazamiento x del bloque, se obtiene de la primera y la segunda derivada de a_x, respectivamente. En la solución teórica se considera que la aceleración del bloque es constante.

$$a_x = g\cos\beta(\tan\beta - \tan\phi) \tag{2-70}$$

$$v_x = v_0 + a_x t \tag{2-71}$$

$$x = x_0 + v_x t + \frac{1}{2}a_x t^2 \tag{2-72}$$

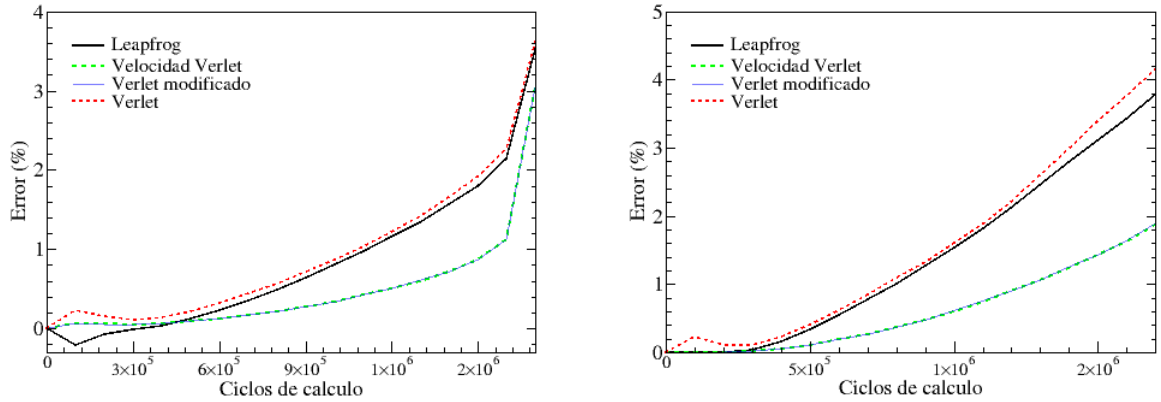
donde v_0 y x_0 son la velocidad y la posición en x del bloque en el tiempo $t = 0$. El bloque tiene una masa de 5000kg, se encuentra sobre una superficie de 30° de inclinación con un ángulo de fricción de 15°. El problema se analiza para un total de 1.7 millones de pasos de tiempo Δt que equivalen aproximadamente a 5.3 segundos. Con estos datos se resuelve el problema por los 4 algoritmos de integración. La comparación de los resultados se hace con base en la solución teórica a través de un error porcentual para el cálculo de velocidad y de posición así:

$$Err_v = \frac{v^{teorica} - v^{algoritmo}}{v^{teorica}} \tag{2-73}$$

$$Err_r = \frac{r^{teorica} - r^{algoritmo}}{r^{teorica}} \tag{2-74}$$

En la Figura **2-8** se presenta el error porcentual que ocasiona cada algoritmo. En la parte (a) se observa que los algoritmos de Leapfrog y Verlet son los que más se alejan de la respuesta teórica en el calculo de la posición. Sin embargo, pareciera que al final los algoritmos convergen a un mismo valor. En la parte (b) de la Figura la situación es similar. De nuevo los algoritmos de Leapfrog y Verlet dan como resultado un mayor error en el cálculo de velocidad. Los algoritmos de Verlet modificado y Velocidad Verlet dan los mejor resultados y son prácticamente idénticos. Como Verlet modificado utiliza un paso menos en su aplicación, en comparación a Velocidad Verlet, este algoritmo es el que se utiliza en el programa *Toppling_UN*.

El cálculo de la Fuerza Neta y el Torque resultante dependen del choque entre los bloques, tema que se discutirá a continuación.



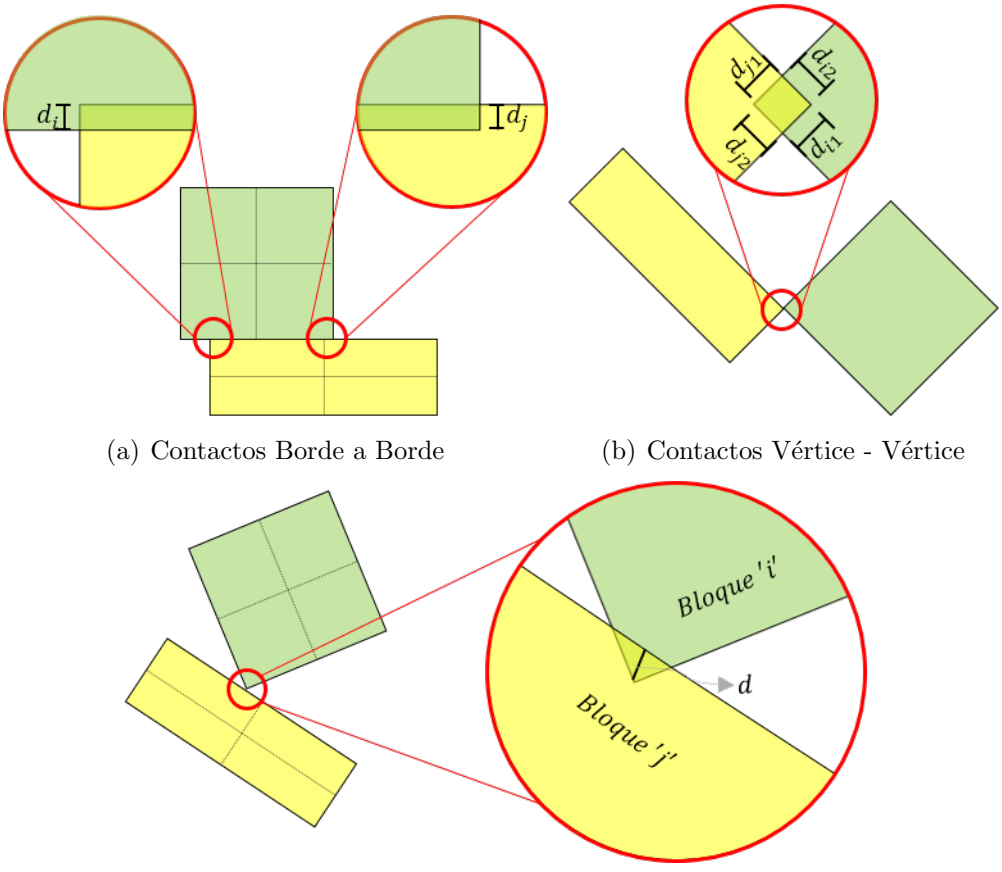
(a) Error para el cálculo de desplazamiento

(b) Error para el cálculo de velocidad

Figura 2-8.: Comparación de algoritmo de integración: un bloque deslizando

2.2. Tipos de contacto y superposición de una pareja de bloques

Cuando se analiza el movimiento de dos bloque i y j sobre un plano, existen tres formas diferentes en que se presentan los contactos: Vértice a Borde (V-B), Borde a Borde (B-B) y Vértice a Vértice (V-V). El bloque i se considera como *activo* mientras el bloque j como *pasivo*. La superposición que de i sobre j se mide por el vector d_{ij}. De esta manera, los contactos B-B y V-V se pueden simplificar a una combinación del contacto V-B. Un contacto B-B se puede descomponer en dos contactos V-B con dos longitudes de superposición, d_i y d_j, como se observa en la parte (a) de la figura **2-9**. Por otro lado, como se presenta en la parte (b) de la figura **2-9**, la descomposición de un contacto V-V se logra evaluando 4 contactos del tipo V-B con longitudes de superposición d_{i1}, d_{i2}, d_{j1} y d_{j2}. Con base en esto, las posibilidades de contacto se reducen a colisiones del tipo vértice-borde.



(a) Contactos Borde a Borde

(b) Contactos Vértice - Vértice

(c) Contactos Vértice a Borde

Figura 2-9.: Simplificación de los contactos V-V y B-B al tipo V-B

Para una pareja de bloques 'i' y 'j', cada uno de los cuatro vértices de i puede impactar con cada uno de los bordes j. Del mismo modo, cada vértice de j puede estar en contacto con los bordes de i. De esta manera se tiene un número total de posibilidades de contacto para una pareja de bloques rectangulares de:

$$N_c = V_i * B_j + V_j * B_i = 4 * 4 + 4 * 4 = 32 \tag{2-75}$$

donde N_c es el número de contactos, V_i y V_j es el número del vértices, y B_j yB_j es el número del bordes de cada bloque. Nótese que un macizo rocoso está compuesto por un número n de bloques, con lo cual el número total de posibles contactos aumenta en razón de las posibles combinaciones entre bloques así:

$$N_{ct} = N_c * \frac{n!}{2!(n-2)!} \tag{2-76}$$

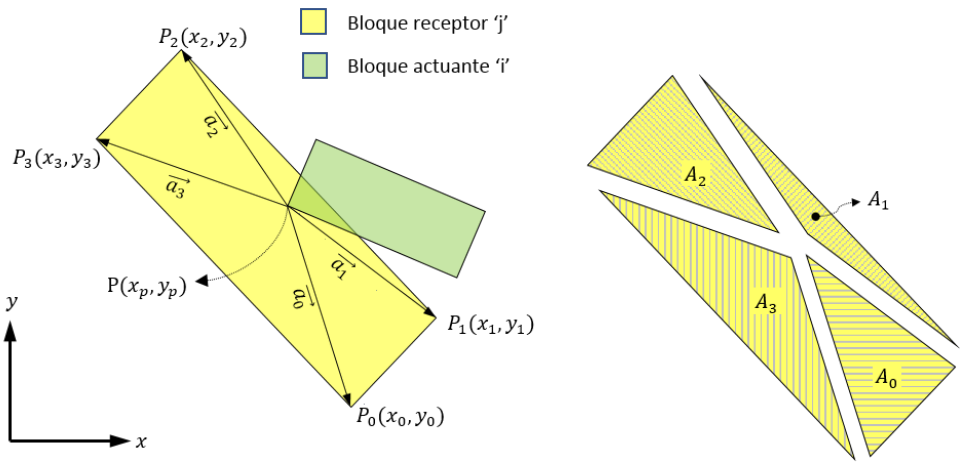
donde N_{ct} es el número total de contactos del sistema de bloques, N_c es el número de contactos entre dos bloques y n es el número de bloques del sistema.

El talud rocoso que se presentó en el capítulo 1.1 tiene un total de once bloques, sin considerar los bloques fijos necesarios para simular el "piso", lo que significa que hay 1760 posibles contactos que deben ser analizados en cada uno de los intervalos de tiempo que dure el proceso de volteo. El mecanismo de buscar todos los choques posibles se conoce como "método de búsqueda directo" [27] y es la manera más práctica de encontrar contactos en un sistema de bloques con pocos elementos. Para sistemas con gran número de elementos, es necesario emplear métodos más desarrollados que disminuyan los ciclos de búsqueda y, consecuentemente, la cantidad de cálculos y memoria computacional empleada durante la simulación.

A continuación se describe el procedimiento empleado para identificar el posible contacto entre dos bloques. Posteriormente, se presenta la metodología empleada para calcular la longitud de superposición; esta última es variable en cada intervalo de tiempo.

2.2.1. Método para identificar el contacto entre dos bloques

Para determinar si hay o no contacto entre dos bloques se empleó un procedimiento geométrico sencillo que consiste en identificar si un punto se encuentra ubicado dentro de un polígono convexo. Para bloques se busca saber si el vértice de un bloque 'i', representado por un punto, se encuentra dentro de un bloque 'j', representado por un polígono rectangular. En la Figura **2-10** (a) se ilustran los elementos que de deben conocer para aplicar el procedimiento. Del bloque actuante, se requiere la posición del vértice que posiblemente esté en contacto, a esta posición la llamaremos $\vec{P}$, con componentes $(x_p, y_p, 0)$. Del bloque receptor es necesario



(a) Puntos de referencia (b) Subolígonos del bloque

Figura 2-10.: División de un bloque en polígonos triangulares

conocer la posición de todos sus vértices; a esas posiciones las llamamos $\vec{P}_0$, $\vec{P}_1$, $\vec{P}_2$ y $\vec{P}_3$, con sus respectivas componentes cartesianas $(x, y, 0)$.

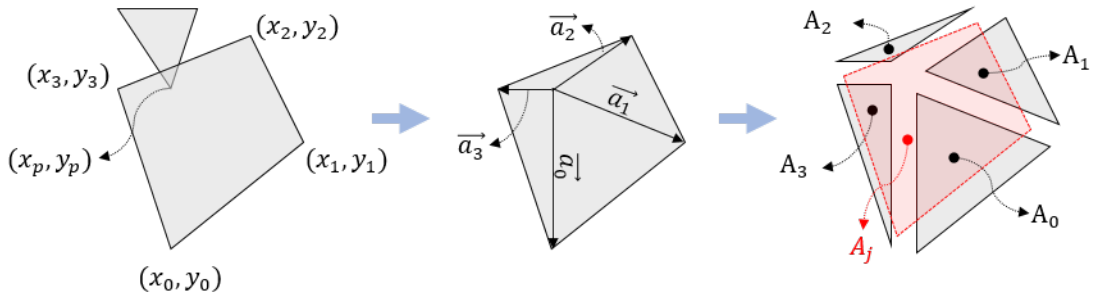
El procedimiento parte de una hipótesis elemental: Cualquier polígono se puede dividir en un conjunto de triángulos que abarquen su misma área. Esto se ilustra en la figura **2-10** (b). Para construir los triángulos, únicamente es necesario trazar vectores desde el punto P hacia cada uno de los vértices del bloque receptor. El punto P es el punto común de la triangulación, es el ápice de todos los triángulos que dividen al bloque. Los vectores que van del punto P a cada vértice se denomina $\vec{a_i}$ y se calculan como:

$$\vec{a_i} = \vec{P_j} - \vec{P} \tag{2-77}$$

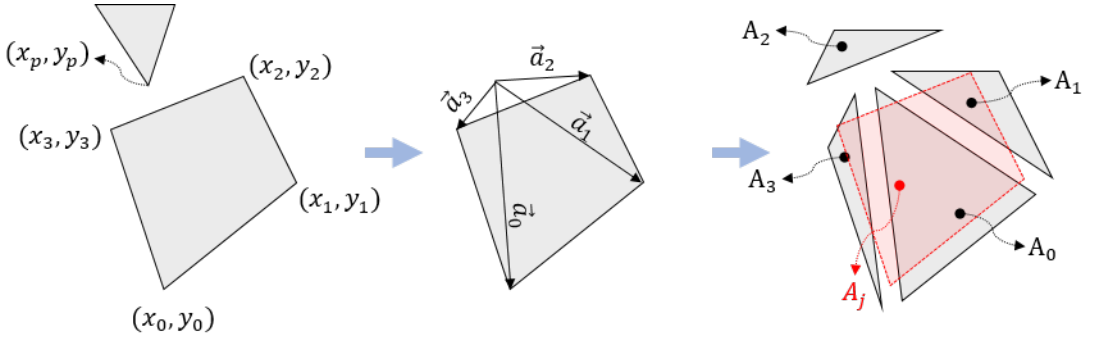
Los vectores $\vec{a_i}$ delimitan los triángulos que conforman el bloque. Cada triángulo se forma a partir de una pareja de vectores que se seleccionan en un orden secuencial y ordenado, siguiendo el sentido de las manecillas del reloj e iniciando desde $\vec{a_0}$

Cuando dos bloques están en contacto se cumple fielmente la hipótesis. Si los bloques no están en contacto, el área de los triángulos resulta mayor a la de bloque como se ejemplifica en la Figura **2-11**. En la parte (a) de la figura, los bloques están en contacto y se observa que las áreas coinciden. Por otro lado, en la parte (b), se presenta un escenario donde los bloques se encuentran cercanos pero sin contacto. Allí se ve que las áreas son diferentes, lo que difiere de la hipótesis

El área de un triangulo se puede obtener por medio del producto cruz, entre sus dos vectores $\vec{a}$. Como el problema es en condición plana, el resultado es un pseudovector paralelo al plano



(a) Dos bloques en contacto



(b) Dos bloques sin contacto

Figura 2-11.: Procedimiento gráfico para la busqueda de contactos

$(\hat{x}\hat{y})$. En este caso el procedimiento se reduce a evaluar el signo del pseudovector que resulta. Si para los cuatro triángulos el producto cruz es positivo significa que hay contacto entre los bloques. En caso contrario no hay contacto.

En resumen, para conocer si dos bloques 'i' y 'j' se encuentran en contacto, se requiere seguir los siguientes pasos:

1. Localizar el vértice actuante del bloque 'i' por medio de un vector $\vec{P}$ con componentes $(x_p, y_p, 0)$.

2. Localizar todos los vértices del bloque receptor 'j' por medio de un conjunto de vectores $\vec{P_j}$ con componentes $(x_j, y_j, 0)$.

3. Construir los vectores $\vec{a_j}$ por medio de la ecuación 2-77.

4. Identificar los triángulos que conforman al polígono j. Cada triangulo está limitado, en forma ordenada y siguiendo el sentido de las manecillas del reloj, por una pareja de vectores $\vec{a_j}$.

5. Calcular el producto cruz entre la pareja de vectores que conforman el primer triangulo. Si el resultado da una componente en Z negativa significa que no hay contacto entre los bloques. Si por el contrario la componente en Z del producto cruz es positiva se prosigue por el siguiente triangulo.

6. Si luego de evaluar todos los triángulos no se encontró un pseudovector con componente negativa en Z significa que los bloques están en contacto.

2.2.2. Método para calcular la superposición de los bloques

Se tiene una pareja de bloques $Bq1$ y $Bq2$ que se encuentran en movimiento. En el tiempo $t = t_0$ los bloques tienen una velocidad $\vec{v_0}^1$ y $\vec{v_0}^2$, respectivamente. Los bloques no están en contacto en el tiempo t_0, sin embargo, debido a la velocidad que poseen, un instante de tiempo después en $t = t_1$ entran en contacto. El choque de los bloques se origina debido a la colisión entre uno de los vértices de $Bq1$ -ubicado en el punto P_i- y el borde $\vec{e}$ de $Bq2$, como se observa en la parte (a) Figura **2-12**. El borde $\vec{e}$ está limitado por los vértices P_0^2 y P_1^2 de $Bq2$.

El vector $\vec{d}$, que determina la superposición de los bloques, varía durante en el intervalo de tiempo en que dura la colisión, como se ve en la parte (b) Figura **2-12**. Allí se muestra la trayectoria que sigue el bloque $Bq2$ desde un tiempo inicial t_0, hasta un tiempo final t_f. En el tiempo t_0 y t_f no hay colisión, durante los instantes intermedios $(t_1, t_2, ..., t_{f-1})$ los bloques sí se encuentran en contacto. La velocidad final de cada bloque es $\vec{v_f}^1$ y $\vec{v_f}^2$, respectivamente. Se observa que en el intervalo $[t_0, t_1]$ el bloque $Bq1$ entró en contacto con el bloque $Bq2$ por

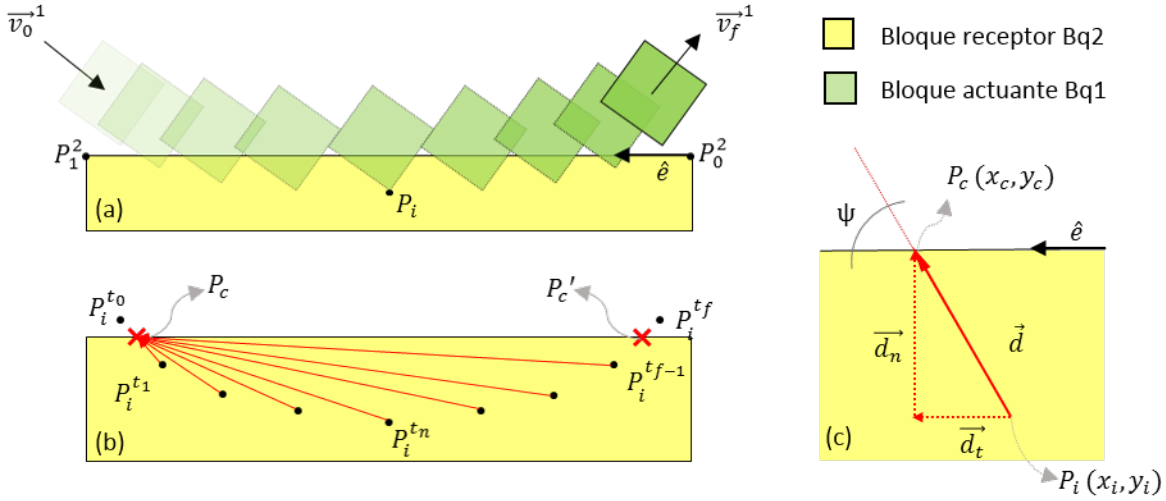


Figura 2-12.: Definición de la longitud de superposición

el punto P_c. Igualmente, en el intervalo $[t_{f-1}, t_f]$ los bloques finalizan la colisión y el ultimo punto donde se presentó el contacto es P'_c.

Con base en un sistema de referencia inercial se tiene que el vector $\vec{d}$ va de la posición del punto P_i a la posición del punto P_c:

$$\vec{d} = \vec{r_i} - \vec{r_c} \tag{2-78}$$

De esta manera, la longitud de superposición d es la magnitud del vector $\vec{d}$.

$$d = |\vec{d}| \tag{2-79}$$

Nótese que el vector $\vec{d}$ no es perpendicular al borde por el cual inició el contacto. Así se muestra en la parte (c) de la figura **2-12**. Entre el borde $\vec{e}$ y el vector $\vec{d}$ se forma un ángulo ψ que se calcula como:

$$\psi = \cos^{-1} \frac{\vec{d} \cdot \hat{e}}{|\vec{d}||\hat{e}|} \tag{2-80}$$

Para calcular el vector $\vec{d}$ se debe conocer con antelación el punto inicial de contacto P_c. Este punto se encuentra ubicado en la intersección de las líneas L_1 y L_2, que se ilustran en la figura **2-13**. La línea L_1 es la recta que pasa sobre los puntos $P_i^{t_0}$ y $P_i^{t_1}$, que son el punto P_i en el tiempo t_0 y t_1 respectivamente. La linea L_2 es la recta que pasa sobre el borde $\vec{e}$.

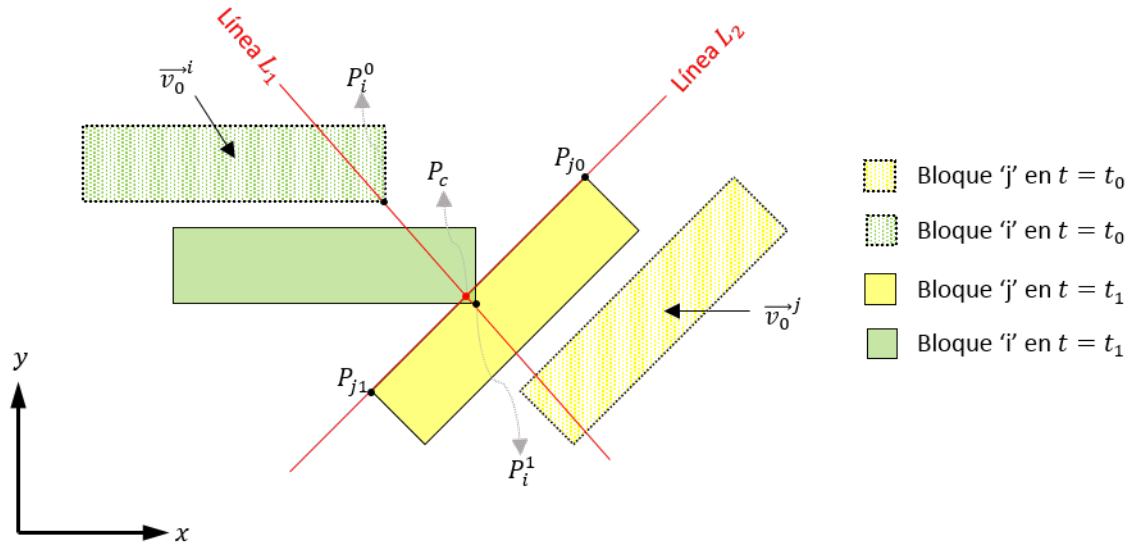


Figura 2-13.: Ubicación del punto inicial de contacto

La línea L_1 se define en forma paramétrica como:

$$L_1 = \begin{pmatrix} x_1 + a_1\lambda_1 \\ y_1 + b_1\lambda_1 \end{pmatrix} \tag{2-81}$$

donde (x_1, y_1) son las coordenadas del punto $P_i^{t_0}$, (a_1, b_1) son las componentes del vector que va de $P_i^{t_0}$ a $P_i^{t_1}$ y λ_1 es el parámetro de L_1. De igual manera, la línea L_2 se define como:

$$L_2 = \begin{pmatrix} x_2 + a_2\lambda_2 \\ y_2 + b_2\lambda_2 \end{pmatrix} \tag{2-82}$$

donde (x_2, y_2) son las coordenadas del punto P_0^2, (a_2, b_2) son las componentes del vector $\vec{e}$.

$$\vec{e} = (a_2, b_2) = |\vec{e}|\hat{e} = \vec{r}_{P_1^2} - \vec{r}_{P_0^2} \tag{2-83}$$

Para conocer las coordenadas del punto P_c basta con hallar el punto de intersección entre la líneas L_1 y L_2. Al igualar las ecuaciones 2-81 y 2-82, se obtiene un sistema de ecuaciones de 2x2 donde las incognitas son λ_1 y λ_2.

$$x_1 + a_1\lambda_1 = x_2 + a_2\lambda_2 \tag{2-84}$$

$$y_1 + b_1\lambda_1 = y_2 + b_2\lambda_2 \tag{2-85}$$

Al despejar λ_1 en las ecuaciones 2-84 y 2-85, luego igualando los resultados y posteriormente despejando λ_2 se obtiene

$$\lambda_2 = \frac{a_1}{a_1 b_2 - a_2 b_1}\left(\frac{b_1}{a_1}(x_2 - x_1) - (y_2 - y_1)\right) \tag{2-86}$$

La Ecuación 2-86 es indeterminada cuando $a_1 = 0$, esto ocurre si la trayectoria del punto P_i es paralela al eje Y. En este caso conviene despejar λ_2 en las ecuaciones 2-84 y 2-85, igualar los resultados y obtener λ_1 de allí:

$$\lambda_1 = \frac{a_2}{a_2 b_1 - a_1 b_2}\left(\frac{b_2}{a_2}(x_1 - x_2) - (y_1 - y_2)\right) \tag{2-87}$$

En cualquier caso, bien sea por la ecuación 2-86 o 2-87, se conoce el valor de λ_1 o λ_2 y por medio de las ecuación 2-81 o 2-82, según corresponda, se obtienen las coordenadas del punto P_c. El punto P_c se mueve en conjunto con el bloque $Bq2$, sin embargo, el parámetro λ_2 permanece constante durante la colisión y debido a esto es posible obtener las coordenadas de P_c en todo instante de tiempo.

Conocida la posición de P_c y P_i se obtiene el vector de superposición $\vec{d}$ por medio de la ecuación 2-78. Las componentes normal y tangencial del vector $\vec{d}$ ($\vec{d_n}$ y $\vec{d_t}$), respecto a $\hat{e}$, se

obtienen a través de una relación trigonométrica simple. Cuando $\psi \leq \pi/2$ tenemos que la magnitud de las componentes de $\vec{d}$ son:

$$\begin{cases} d_n &= d\sin\psi \\ d_t &= d\cos\psi \end{cases} \tag{2-88}$$

Cuando $\psi > \pi/2$ la magnitud de las componentes de $\vec{d}$ son:

$$\begin{cases} d_n &= d\cos(\psi - \pi/2) \\ d_t &= d\sin(\psi - \pi/2) \end{cases} \tag{2-89}$$

El vector director de $\vec{d_n}$ está definido por el resultado del producto cruz entre $\hat{e}$ y $\hat{z}$, siendo $\hat{z}$ el eje perpendicular al sistema de referencia.

$$\hat{d_n} = \hat{e} \times \hat{z} \tag{2-90}$$

Por otra parte, dado que $\vec{d} = \vec{d_n} + \vec{d_t}$, el vector director de $\vec{d_t}$ es

$$\hat{d_t} = \frac{\vec{d} - \vec{d_n}}{|\vec{d} - \vec{d_n}|} \tag{2-91}$$

Con esto quedan definidas las componente normal y tangente del vector de superposición $\vec{d}$, durante la totalidad del tiempo que dura la colisión.

$$\vec{d_n} = d_n\hat{d_n} \tag{2-92}$$

$$\vec{d_t} = d_t\hat{d_t} \tag{2-93}$$

Esta pareja de ecuaciones es de gran importancia para el cálculo de fuerzas, ya que la fuerza normal y la fuerza tangencial, que se produce durante una colisión de bloques, se calcula con base en $\vec{d_n}$ y $\vec{d_n}$ respectivamente. En el siguiente capítulo se presenta el modelo de fuerzas que se empleó para simular la colisión entre bloques.

2.3. Modelo reológico para simular las fuerzas de choque

En el programa *Toppling_UN*, desarrollado en este trabajo, se utilizan dos modelos reológicos para simular la fuerza de choque: el modelo Kelvin-Voigth, para simular la componente normal de la fuerza, $\vec{F_n}$; y el modelo Elástico Plástico Perfecto, para simular la componente tangencial de la fuerza, $\vec{F_t}$. El choque se presenta entre uno de los vértices del bloque i y uno de los bordes del bloque j; como se presenta en la parte (a) de la Figura **2-14**. La Fuerza de Choque $\vec{F}$ es la suma vectorial de $\vec{F_n}$ y $\vec{F_t}$:

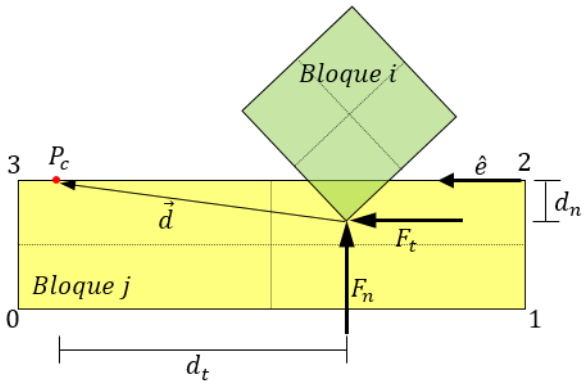
$$\vec{F} = \vec{F_n} + \vec{F_t} \tag{2-94}$$

La Fuerza Normal $\vec{F_n}$ es una fuerza de repulsión de tipo viscoelástico mientras la Fuerza Tangente $\vec{F_t}$ es una fuerza que simula la fricción durante el choque. Nótese que $\vec{F}$ es la fuerza de reacción sobre el bloque i

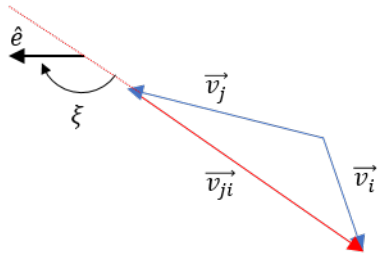
$$\vec{F_i} = \vec{F} \tag{2-95}$$

y, de acuerdo con la tercera Ley de Newton, la fuerza que actúa sobre j es:

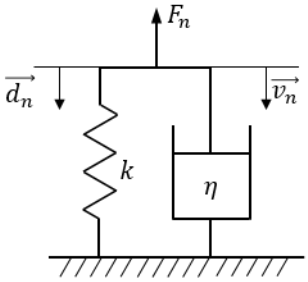
$$\vec{F_j} = -\vec{F} \tag{2-96}$$



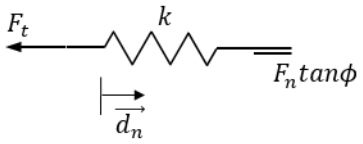
(a) Diagrama de fuerzas visto desde i



(b) Velocidad relativa de los bloques



(c) Modelo de Kelvin-Voigth



(d) Modelo Elástico Plástico Perfecto

Figura 2-14.: Modelo reológico para simular las fuerzas de choque

La fuerza $\vec{F}$ se localiza en la posición del vértice del bloque i. Cada una de sus componentes va en la misma dirección de su correspondiente vector $\vec{d_n}$ y $\vec{d_t}$, los cuales se definieron en el capítulo anterior en las Ecuaciones 2-92 y 2-93. El plano de choque está definido por el vector unitario $\hat{e}$ que va del vértice 2 al vértice 3 del bloque j. Es bueno recordar que los vértices de los bloques se numeran de 0 a 3 en sentido antihorario, como se definió en el capítulo 1.2.3. De esta manera, $\vec{F_n}$ es perpendicular a $\hat{e}$ y $\vec{F_t}$ es vector paralelo a $\hat{e}$. El sentido de los vectores es función de la velocidad relativa de los bloques. Cada uno de lo bloques se mueve con velocidades diferentes, $\vec{v_i}$ y $\vec{v_i}$. La velocidad relativa $\vec{v_{ji}}$ es la diferencia entre $\vec{v_i}$ y $\vec{v_j}$:

$$\vec{v}_{ji} = \vec{v_i} - \vec{v_j} \tag{2-97}$$

como se observa en la parte (b) de la Figura **2-14**.

El ángulo ξ, que va del vector $\vec{v}_{ji}$ al vector $\hat{e}$ se calcula como:

$$\xi = \cos^{-1} \frac{\vec{v}_{ji} \cdot \hat{e}}{|\vec{v}_{ji}||\hat{e}|} \tag{2-98}$$

Conocido $\vec{v}_{ji}$ y ξ, es posible calcular la magnitud $\vec{v_n}$ como

$$v_n = \begin{cases} |\vec{v}_{1-2}| \sin \xi & si \quad \xi \leq \pi/2 \\ |\vec{v}_{1-2}| \cos(\xi - \pi/2) & si \quad \xi > \pi/2 \end{cases} \tag{2-99}$$

y vector director de $\vec{v_n}$ resulta del producto cruz entre $\hat{e}$ y $\hat{z}$, donde $\hat{z}$ es $\hat{x} \times \hat{y}$:

$$\hat{v_n} = \hat{e} \times \hat{z} \tag{2-100}$$

con lo cual $\vec{v_n}$ es:

$$\vec{v_n} \;=\; v_n \hat{v_n} \tag{2-101}$$

En la parte (c) de la Figura **2-14** se muestra el modelo reológico de la fuerza $\vec{F_n}$. El resorte tiene una rigidez k_n y el amortiguador tiene una viscosidad η. La fuerza $\vec{F_n}$ es la suma vectorial entre la fuerzas elástica $\vec{F}_{ne}$ y la fuerza viscosa $\vec{F}_{nv}$:

$$\vec{F_n} \;=\; \vec{F}_{ne} + \vec{F}_{nv} = k_n \vec{d_n} + \eta \vec{v_n} \tag{2-102}$$

De igual manera, en la parte (d) de la Figura **2-14**, se muestra el modelo reológico de la fuerza $\vec{F_t}$. La fuerza F_t tiene un valor máximo de $F_n \tan \phi$, por debajo de este valor la fuerza se calcula por medio del resorte de rigidez k_t.

$$\vec{F_t} \;=\; \begin{cases} k_t \vec{d_t} & si \quad k\vec{d_t} \leqslant F_n \tan \phi \\ \vec{F_n} \tan \phi & si \quad k\vec{d_t} > F_n \tan \phi \end{cases} \tag{2-103}$$

Nótese que la fuerza viscosa $\vec{F}_{nv}$ y la fuerza friccional $\vec{F}_t$ no son fuerzas conservativas. En el caso de $\vec{F}_{nv}$, el amortiguador viscoso cumple la función de restringir la velocidad de la colisión, con lo cual la energía cinética del sistema disminuye con cada choque, lo que significa que se presentan rebotes inelásticos. En el caso de $\vec{F}_t$, el límite plástico simula una disipación de energía por fricción. A continuación se analizan estos procesos y se verifica su aplicación en el programa *Toppling_UN*.

2.3.1. Fuerzas de choque normal

Con el propósito de verificar la correcta implementación de las fuerzas $\vec{F}_n$ en el programa *Toppling_UN*, se estudia la energía mecánica un bloque que cae desde una altura de 5m. El bloque tiene un volumen de $2m^3$ y una masa de 5000kg. En este problema no se consideran pérdidas de energía por fricción. Se estudian dos caso: Cuando el bloque puede girar y cuando el bloque no gira. En la Figura **2-15** se presenta el diagrama de cuerpo libre de las fuerzas que actúan antes, durante y después, del rebote del bloque. En la parte (a), el bloque no gira. Es como si fuera una partícula esférica rebotando sobre una superficie horizontal. En la parte (b), la forma del bloque y la ubicación del vértice en contacto produce un torque τ, que se calcula en el centro de masa. Este torque produce un cambio en la velocidad angular y, en consecuencia, el bloque gira. En el bloque siempre actúa la fuerza gravitacional W. El bloque se comporta como cuerpo rígido, por esto es conveniente separar los procesos de rotación y traslación.

La energía potencial del bloque, depende de la altura de su centro de masa respecto a un nivel de referencia que se encuentra fijo en la superficie horizontal de impacto. También, la energía potencial depende de la compresión del resorte durante la colisión. La energía potencial se calcula como:

$$E_p = mgh + \frac{1}{2}kd_n^2 \tag{2-104}$$

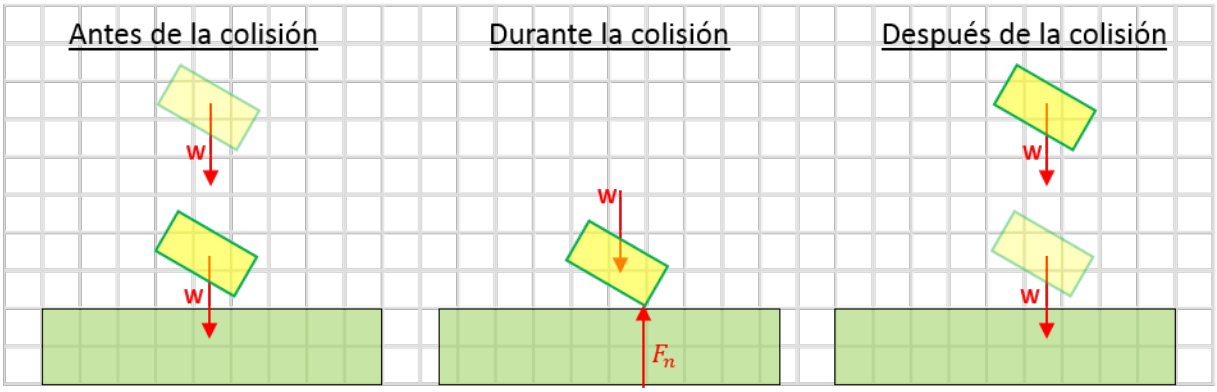
Donde m es la masa del bloque, g es la gravedad, h la elevación del centro de masa respecto de la superficie de choque, k y d_n son la rigidez y la deformación del resorte, respectivamente. Por otra parte, la energía cinética depende de la velocidad del bloque, tanto en el movimiento lineal, como en rotación, así:

$$E_c = \frac{1}{2}m|\vec{v}|^2 + \frac{1}{2}I_c|\vec{\omega}|^2 \tag{2-105}$$

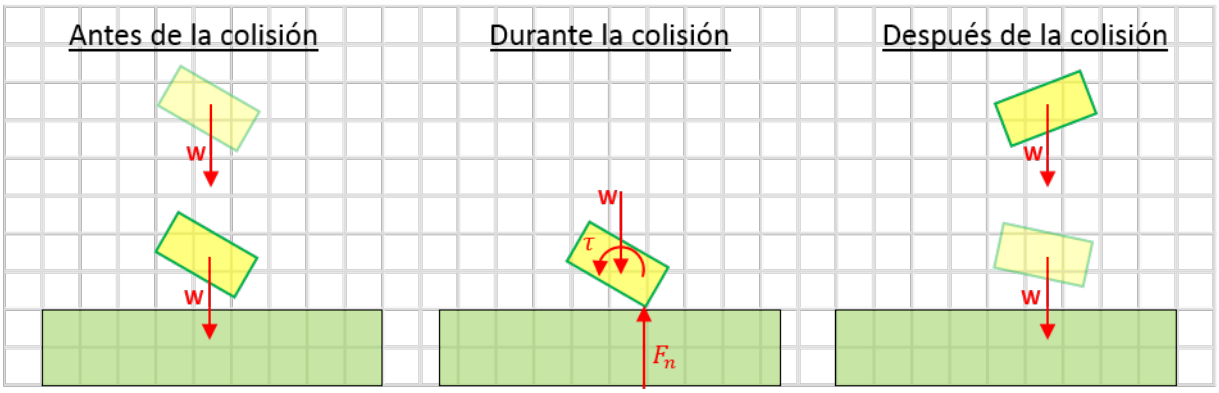
Donde $\vec{v}$ es la velocidad del bloque, $\vec{\omega}$ es la velocidad angular y I_c es el momento de inercia del bloque girando en el centro de masa. La energía mecánica es la suma de E_c y E_p.

$$E_t = E_c + E_p \tag{2-106}$$

En el programa *Toppling_UN* el bloque se mueve de acuerdo con las ecuaciones descritas en la sección 2.1.



(a) Bloque rebotando sin rotación



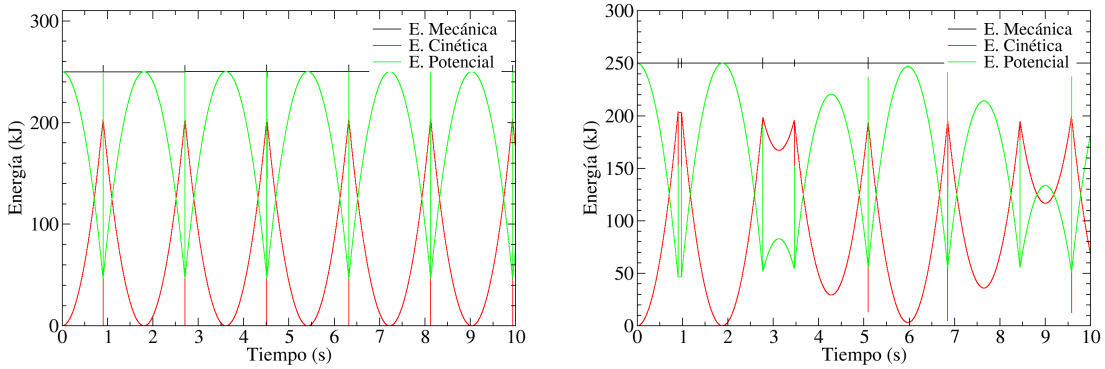
(b) Bloque rebotando con rotación

Figura 2-15.: Valicación de la fuerza de choque normal

2.3.1.1. Fuerzas de choque normal - elástica

Cuando el bloque cae y rebota sin amortiguamiento; es decir, con $\eta = 0$, la energía mecánica se conserva, como se observa en la Figura **2-16**. En la parte (a) de la Figura, no se considera la rotación del bloque. Se pueden ver varios picos en la energía potencial que ocurren cuando se presenta cada rebote. En total se cuentan 6 rebotes en un intervalo de 10 segundos. En la parte (b) de la Figura, el bloque tiene la posibilidad de rotar. Ahora, con cada rebote el bloque cambia su velocidad angular y por tanto su orientación. Los cambios irregulares que se observan, en comparación al caso sin rotación, surgen porque el cálculo de la energía cinética ahora incluye la velocidad angular del bloque, según la Ecuación 2-105. En este escenario se presentan 8 rebotes en 10 segundos.

La energía mecánica del bloque en su condición inicial es de 250kJ. Como es un proceso conservativo, esta energía siempre permanece constante, con muy pequeñas variaciones producidas por las oscilaciones del cálculo numérico. Durante el intervalo de 2,5s y 4,0s, se



(a) Bloque rebotando sin rotación

(b) Bloque rebotando con rotación

Figura 2-16.: Variación de la energía con fuerzas elástica ($k = 20GPa$)

presentan dos picos en la energía cinética que corresponden a dos rebotes del bloque. En ese intervalo de tiempo la velocidad angular predomina frente a la velocidad lineal, lo que se ve reflejado en el cálculo de la energía cinética. La misma situación ocurre entre el segundo 8,5 y el segundo 10.

Para ilustrar este proceso, en la Figura **2-17** se realiza un seguimiento al recorrido del bloque entre $t = 2{,}6s$ y $t = 3{,}8s$, cuando este tiene la posibilidad de rotar. Nótese que el bloque choca con el "piso" aproximadamente en $t = 2{,}75s$, ahí ocurre un cambio en la orientación del bloque, lo que coincide con uno de los picos de energía cinética. En los siguientes instantes, el bloque mantiene una altura cercana a 1.5m y su energía potencial permanece entre 60kJ y 80kJ. El bloque se encuentra rotando con una velocidad angular constante hasta que ocurre el siguiente choque en $t = 3{,}45s$. Luego de este choque disminuye la velocidad angular, aumenta velocidad lineal y, en consecuencia la elevación del bloque aumenta.

La fuerza elástica mantiene rebotando el bloque de forma infinita, lo que no sucede en la realidad. Por esto, se añade la fuerza de amortiguamiento.

2.3.1.2. Fuerzas de choque normal - amortiguada

Con objetivo de finalizar el rebote se incluye una fuerza de amortiguamiento. Esta fuerza actúa en paralelo con la fuerza elástica, modelando el comportamiento viscoelástico que se describió al principio de esta sección en la Figura **2-14**. Se consideraron 3 opciones de amortiguamiento con relación a la rigidez del resorte[3]: $\eta/k = 5x10^{-5}$, $\eta/k = 25x10^{-5}$ y

[3]La relación η/k se calcula con base en la unidades de computador, no en las unidades físicas. El análisis de unidades se presenta en el título 3.1.3.

(a) $t = 2{,}62s$ (b) $t = 2{,}69s$ (c) $t = 2{,}75s$ (d) $t = 2{,}81s$ (e) $t = 2{,}88s$

(f) $t = 2{,}94s$ (g) $t = 3{,}00s$ (h) $t = 3{,}07s$ (i) $t = 3{,}13s$ (j) $t = 3{,}19s$

(k) $t = 3{,}26s$ (l) $t = 3{,}32s$ (m) $t = 3{,}38s$ (n) $t = 3{,}45s$ (ñ) $t = 3{,}51s$

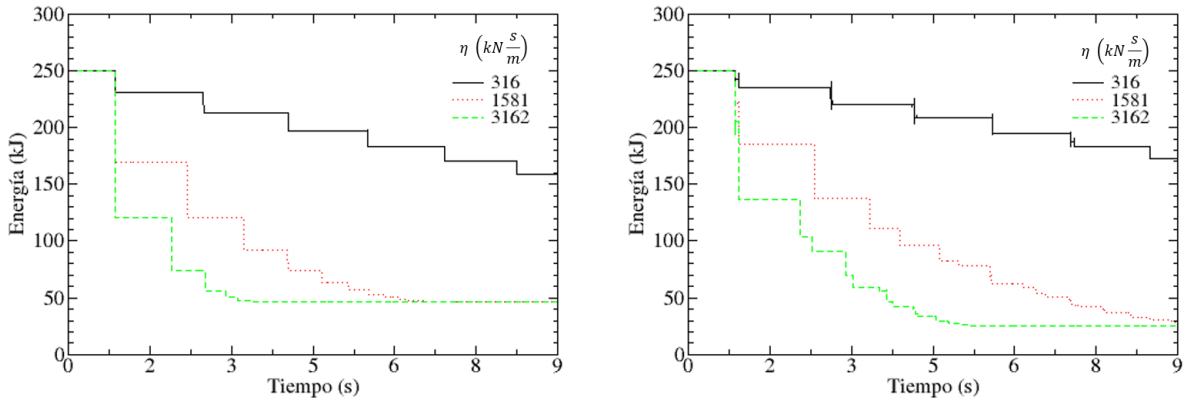
(o) $t = 3{,}57s$ (p) $t = 3{,}64s$ (q) $t = 3{,}70s$ (r) $t = 3{,}76s$ (s) $t = 3{,}83s$

Figura 2-17.: Animación del rebote elástico con rotación, $E = 20GPa$

$\eta/k = 50x10^{-5}$. En todos los casos $k = 20GPa$.

En la Figura **2-18** se observa la variación de la energía mecánica del mismo bloque anterior, pero ahora rebotando con fuerzas de choque viscoelásticas. En la parte (a) el bloque no gira, es como una pelota rebotando. El bloque inicia con una energía mecánica de 250kJ y va descendiendo hasta alcanzar una energía mínima, que equivale a la energía potencial del bloque en reposo. El carácter disipativo es claro y va aumentado según crece la relación η/k. En la parte (b), el bloque tiene la posibilidad de girar y también es posible observar el proceso de disipación. Al comparar los procesos de rebote con y sin rotación, se observa que, en el caso con rotación, hay un pequeño retardo en la disipación de energía. Esto se debe a que la rotación modifica el número de posibilidades de rebote. Lo que ocasiona un tiempo adicional para alcanzar la condición de reposo.

Para ilustrar la forma en que se mueve el bloque, en la Figura **2-19** se presenta el recorrido que sigue el bloque entre el tiempo $2{,}9s$ y $t = 4{,}8s$ cuando tiene la posibilidad de rotar. Los parámetros que modelan la fuerza de amortiguamiento son $k = 20GPa$ y $\eta = 3162kN\frac{s}{m}$.



(a) Bloque rebotando sin rotación (b) Bloque rebotando con rotación

Figura 2-18.: Variación de la energía mecánica con fuerzas viscoelásticas

(a) $t = 2{,}94s$ (b) $t = 3{,}04s$ (c) $t = 3{,}13s$ (d) $t = 3{,}23s$ (e) $t = 3{,}32s$

(f) $t = 3{,}42s$ (g) $t = 3{,}51s$ (h) $t = 3{,}60s$ (i) $t = 3{,}70s$ (j) $t = 3{,}79s$

(k) $t = 3{,}89s$ (l) $t = 3{,}98s$ (m) $t = 4{,}08s$ (n) $t = 4{,}17s$ (ñ) $t = 4{,}27s$

(o) $t = 4{,}36s$ (p) $t = 4{,}46s$ (q) $t = 4{,}55s$ (r) $t = 4{,}65s$ (s) $t = 4{,}74s$

Figura 2-19.: Animación rebote amortiguado con rotación, $E = 20GPa\ \eta/k = 50 \times 10^{-}5$

Aproximadamente en el segundo 3 el bloque colisiona, con ello cambia su velocidad y el rebote lo eleva hasta una altura cercana a los 1.6m. Luego, en el segundo 3.7, ocurre el siguiente rebote pero en esta ocasión se eleva en menor magnitud. Sobre los 5 segundos el bloque se encuentra en reposo.

Las fuerzas de choque normal son una parte del modelo de fuerzas que se utiliza en el programa *Toppling_UN*. Falta verificar el comportamiento de las fuerzas friccionales.

2.3.2. Fuerzas de choque tangencial

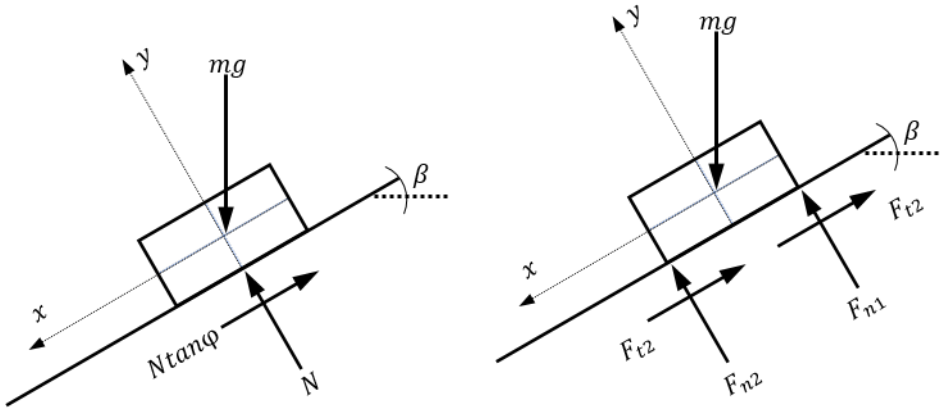
Con el objetivo de analizar la implementación del modelo friccional, en el programa *Toppling_UN*, se analiza un ejercicio de bloque deslizante. La idea es comparar los resultados teóricos con los resultados del programa. Se define un sistema de referencia inercial $\hat{x}\hat{y}$, que tiene su origen en el centro de masa del bloque, como se presenta en la Figura **2-20**. Las fuerzas que actúan sobre el bloque son las ocasionadas por su peso. El bloque parte de la condición de reposo.

2.3.2.1. Solución teórica

La solución teórica del problema se obtiene a partir del equilibrio de las fuerzas de la parte (a) la Figura **2-20**. Luego de realizar algunas operaciones algebraicas simples se puede deducir que:

$$N = mg\cos\beta \tag{2-107}$$

$$F_x = mg\cos\beta(\tan\beta - \tan\phi) \tag{2-108}$$



(a) Para el análisis teórico (b) Para el análisis DEM

Figura 2-20.: Diagrama de fuerza de un bloque deslizando

donde m es la masa del bloque y g es la gravedad. La aceleración en dirección x (a_x), se despeja de la Segunda Ley de Newton. La velocidad v_x y el desplazamiento x del bloque, se obtiene de la primera y la segunda derivada de a_x, respectivamente. En la solución teórica se considera que la aceleración del bloque es constante.

$$a_x = g\cos\beta(\tan\beta - \tan\phi) \quad (2\text{-}109)$$

$$v_x = v_0 + a_x t \quad (2\text{-}110)$$

$$x = x_0 + v_x t + \frac{1}{2}a_x t^2 \quad (2\text{-}111)$$

donde v_0 y x_0 son la velocidad y la posición en x del bloque en el tiempo $t = 0$.

2.3.2.2. Solución por el método DEM

Por otra parte, la fuerza F_t en un programa DEM, como *Toppling_UN*, es variable en el tiempo ya que depende del vector de superposición $\vec{d}$, y este a su vez es función del tiempo. Además, como se explica al principio de este capítulo, las fuerzas de choque se calculan en el vértice que colisiona. Esto quiere decir que el diagrama de fuerzas del bloque es el que se presenta en la parte (b) de la Figura **2-20**.

Al realizar equilibrio de las fuerzas en dirección x tenemos que:

$$F_x = F_{t1} + F_{t2} - mg\sin\beta \quad (2\text{-}112)$$

donde F_{t1} y F_{t2} se calculan por medio de la Ecuación 2-103. La aceleración a_x, la velocidad v_x, y el desplazamiento en x se hallan por medio de la segunda Ley de Newton. Se debe recordar que el método DEM es un procedimiento que marcha en el tiempo, en muy pequeños y sucesivos pasos Δt; de esta manera las variables cinemáticas son:

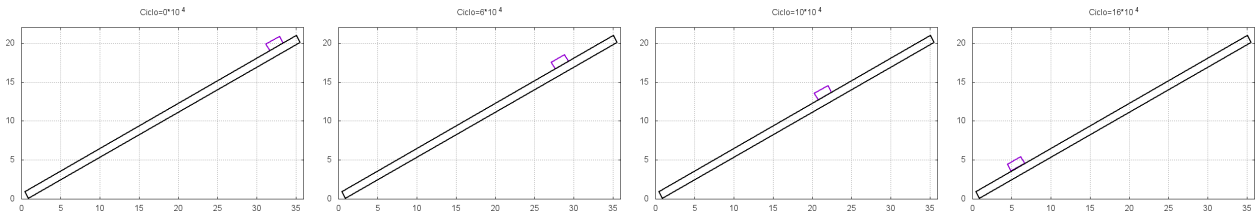
$$a_{x(t+\Delta t)} = \frac{1}{m}(F_{t1} + F_{t2}) - g\sin\beta \quad (2\text{-}113)$$

$$v_{x(t+\Delta t)} = v_t + a_{x(t+\Delta t)}\Delta t \quad (2\text{-}114)$$

$$x_{(t+\Delta t)} = x_t + v_{x(t+\Delta t)}\Delta t + \frac{1}{2}a_{x(t+\Delta t)}(\Delta t)^2 \quad (2\text{-}115)$$

2.3.2.3. Comparación de resultados

Considérese un bloque de masa $m = 5000kg$ que desliza sobre una superficie con una inclinación $\beta = 30°$. El ángulo de fricción entre el bloque y la superficie es $\phi = 15°$. La gravedad tiene un valor de $g = 9{,}81m/s$. En la condición inicial del problema $t = 0$, el bloque no está en movimiento. El análisis DEM se realiza con un paso de tiempo $\Delta t = 3{,}16 * 10^{-6}s$. En la Figura **2-21** se ilustra el movimiento del bloque los primeros 5 segundos, con los resultados del programa *Toppling_UN*.



(a) $t = 0{,}00s$ (b) $t = 1{,}90s$ (c) $t = 3{,}16s$ (d) $t = 5{,}06s$

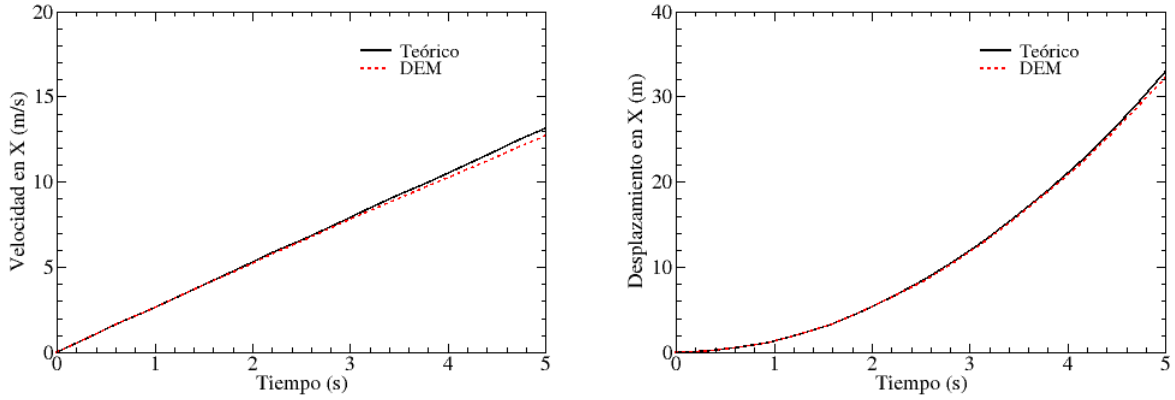
Figura 2-21.: Animación del bloque deslizando con $\phi = 15°$

En la Figura **2-22** se compara el desplazamiento y la velocidad del bloque durante los primeros 5 segundos. Nótese que en ambos casos los resultados teóricos y por DEM son muy similares. Hay diferencias porque los cálculos de DEM acumulan errores en cada paso de tiempo. Con el Δt que se utilizó, el programa *Toppling_UN* realiza más de 1.5 millones de ciclos de cálculo para llegar a 5 segundos. Disminuir el error es posible seccionando un adecuado algoritmo de integración y calibrando las variables de simulación: el paso de tiempo Δt; la rigidez de los resortes k_n y k_t; y la viscosidad del amortiguador η. Esta tarea se presenta en el capítulo 3, previo al análisis dinámico del proceso de *toppling*.

Por ahora, y con la intensión de cuantificar el error de este ejemplo, se calcula un error porcentual para la posición y uno para la velocidad así:

$$Err_{v_x} = \frac{v_x^{teorica} - v_x^{DEM}}{v_x^{teorica}} \tag{2-116}$$

$$Err_x = \frac{x^{teorica} - x^{DEM}}{x^{teorica}} \tag{2-117}$$



(a) Velocidad en eje x (b) Desplazamiento en eje x

Figura 2-22.: Comparación de resultados: bloque deslizante

En la Tabla **2-1** se puede observar que luego de 1500000 pasos de tiempo, se ha acumulado un error porcentual de 3.11 % en el cálculo de velocidad y de 1.80 % en el cálculo de posición.

Tabla 2-1.: Error en el cálculo de la fuerza tangente

Ciclos de cálculo	Tiempo (s)	Error Desplazamiento (%)	Error Velocidad (%)
0	0.00	0.00 %	0.00 %
500000	1.58	0.12 %	0.34 %
1000000	3.16	0.80 %	1.53 %
1500000	4.74	1.80 %	3.11 %

3. Dinámica de un proceso de volteo en un sistema de bloques

El talud que se estudiará es el de la Figura **3-1**. Este es el mismo que se analizó en el Capítulo 1 cuando se presentó el método de Goodman para el análisis estático de *toppling*. En el talud la única fuerza externa que actúa es la fuerza gravitacional. Las fuerzas de colisión entre bloque se consideran fuerzas internas. Los bloques tienen un peso unitario γ de $2600 kg/m^3$ y, en principio, se asume un ángulo de fricción ϕ de 30°. El talud está conformado por 11 bloques, de los cuales 3 resultan ser estables según el análisis de Goodman. Los demás están en una condición de inestabilidad y pueden rotar, desplazarse o hacer ambos movimientos en forma simultanea, como se observa en la Figura. Para el estudio del movimiento se define un sistema inercial de coordenadas XY, con origen el vértice inferior del bloque 0.

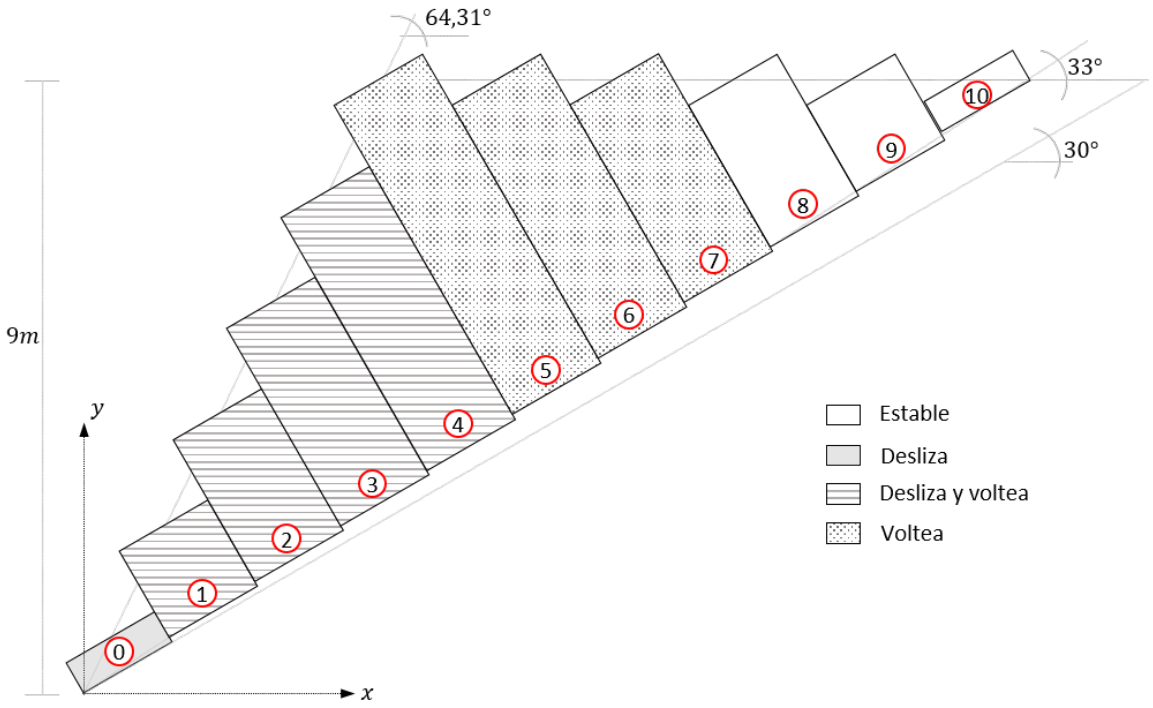


Figura 3-1.: Talud de referencia

El objetivo del problema es evaluar las características del movimiento del talud, hasta que este encuentre una condición de equilibrio. El estudio se realiza por medio del programa *Toppling_UN*, que se formuló como resultado de este Trabajo Final. En el programa se utiliza un total de 23 bloques, 11 pertenecen al talud y hay 12 adicionales para simular el "piso" como se muestra en la Figura **3-2**. Estos últimos siempre permanecen inmóviles y, por supuesto, interactúan con los bloques del talud como elementos de frontera. En general, el programa siempre utiliza un número de $N + 1$ bloques para para simular el piso siendo N, el número de bloques en el talud. La ubicación y el tamaño de cada bloque depende de los planos que limitan el talud. En este caso, el talud tiene una altura de 9m, una inclinación de 64,31°, un escalonamiento de 3° y un plano en la base de 30°.

En la condición inicial, todos los bloques se encuentran separados 2 milímetros para asegurar que los resortes inicien en su condición natural. Cada bloque se ubica por medio de un procedimiento geométrico simple que determina las coordenadas de su centro de masa y de sus vértices. Antes de presentar los resultados del programa es necesario calibrar las variables de simulación. También, se presenta el algoritmo del programa para conocer su funcionamiento.

3.1. Calibración de las variables de simulación

En cualquier análisis de ingeniería es necesario garantizar que los datos que se ingresan a un programa sí representan la realidad del problema estudiado. Esta es una de las formas de reducir la incertidumbre y obtener resultados de buena calidad. Los valores de entrada en el programa *Toppling_UN* son: 1)la geometría del talud, 2)el ángulo de fricción ϕ, 3)la

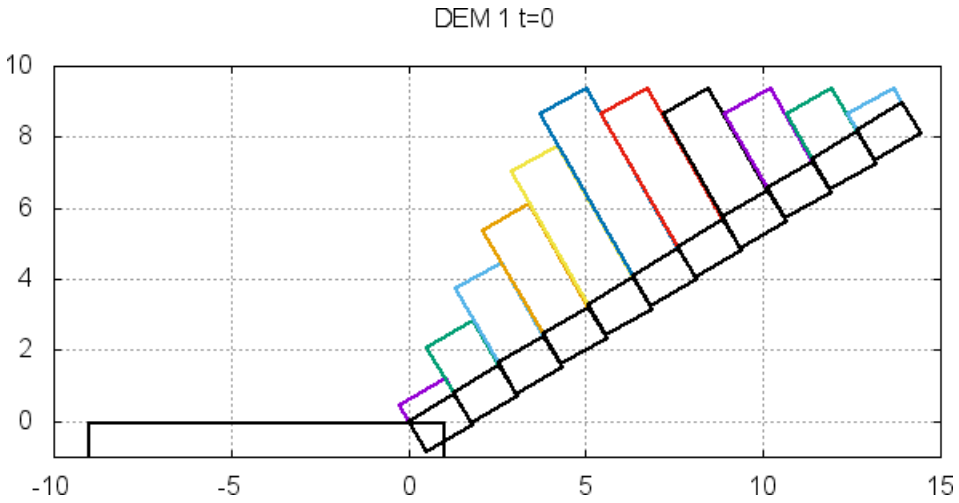


Figura 3-2.: Talud de referencia en *Toppling_UN*

rigidez k de los resortes, 4)el coeficiente de amortiguamiento η y 5)el paso de tiempo Δt. La geometría y la fricción son variables que se obtienen por mediciones en campo y por medio de análisis. Las otras 3 variables se consideran como *variables de simulación*. Estas se deben calibrar a partir de experimentación física o por medio de ejercicios con solución teórica.

En este trabajo se propone lo siguiente. Considérese un bloque como el de la Figura **3-3**. El bloque se ubica sobre una superficie inclinada 30° y tiene una altura de 5m y un ancho de 1m. La fricción que existe en el contacto es de $\phi = 35°$. En estas condiciones el bloque no desliza pero sí voltea. Cuando el bloque cae sobre la superficie se produce un pequeño proceso de rebote, que origina un corto desplazamiento del bloque sobre la superficie. La idea es seguir el desplazamiento, en dirección x, del centro de masa del bloque identificado como A. El problema se analiza con varios valores de Δt, k y η para seleccionar la mejor combinación. Luego, con los parámetros elegidos, se realiza un ejercicio de bloque deslizante para comparar los resultados con la solución teórica del problema.

El programa *Toppling_UN*, al igual que cualquier programa de computador, utiliza unas unidades propias de programación que no son las mismas unidades físicas. Esto se hace para evitar las operaciones entre valores muy grandes y muy pequeños, como es el caso de un módulo muy grande en comparación con un muy pequeño paso de tiempo. Al finalizar la calibración se presenta un análisis dimensional de unidades para verificar que el programa cumpla con las condiciones físicas del problema. Mientras tanto, las unidades de Δt, k y η no son relevantes.

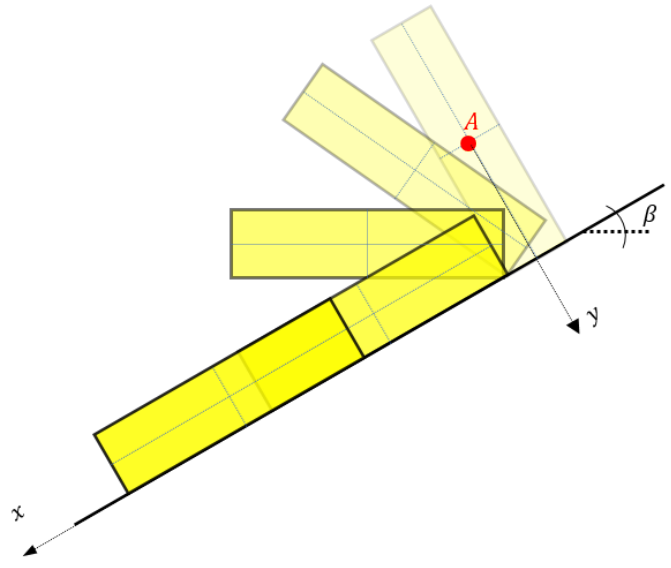


Figura 3-3.: Planteamiento del problema para la calibración de variables

3.1.1. Calibración del paso de tiempo Δt

El paso de tiempo Δt es fundamental en un programa tipo DEM, ya que de él depende el error que se acumula por la aplicación de las series de Taylor. Además, el valor de Δt define el número de ciclos de cálculo, lo que afecta el tiempo de ejecución de los programas. En el Capítulo 2.1.3 se llegó a la conclusión que el algoritmo de integración que mejor se comporta es el de Verlet modificado, que utiliza las ecuaciones de las Diferencias Finitas. Este algoritmo utiliza el término Δt^2 para el calculo de posición, lo que implícitamente significa que se está trabajando con una diferencia finita de segundo orden. En diferencias finitas se tiene que entre más pequeño sea el Δt más preciso es cálculo numérico. Sin embargo, luego de algún valor muy pequeño no se perciben cambios en la respuesta aunque se siga disminuyendo el Δt [25].

El valor de Δt que se debe seleccionar es el más grande que no produzca diferencias significativas en la respuesta. En la Figura **3-4** se presentan los resultados que se obtienen al correr el programa con 4 valores de Δt: 1×10^{-3}, 1×10^{-4}, 1×10^{-5} y 1×10^{-6}. Nótese que los valores más grandes de Δt originan respuestas muy diferentes entre sí. De otra parte, los Δt más pequeños convergen hacia una solución común. El problema se analizó para un periodo de 5 segundos, en ese intervalo el programa utilizó 110s en su ejecución con un Δt de 1×10^{-5} y 845s cuando el Δt es de 1×10^{-6}. Por sus resultados y considerando el tiempo de computo se selecciona un Δt de 1×10^{-5} que equivale a 3,16μs.

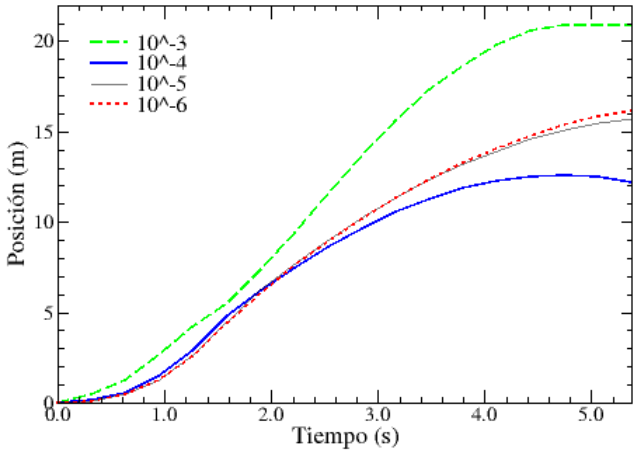


Figura 3-4.: Calibración del paso de tiempo Δt

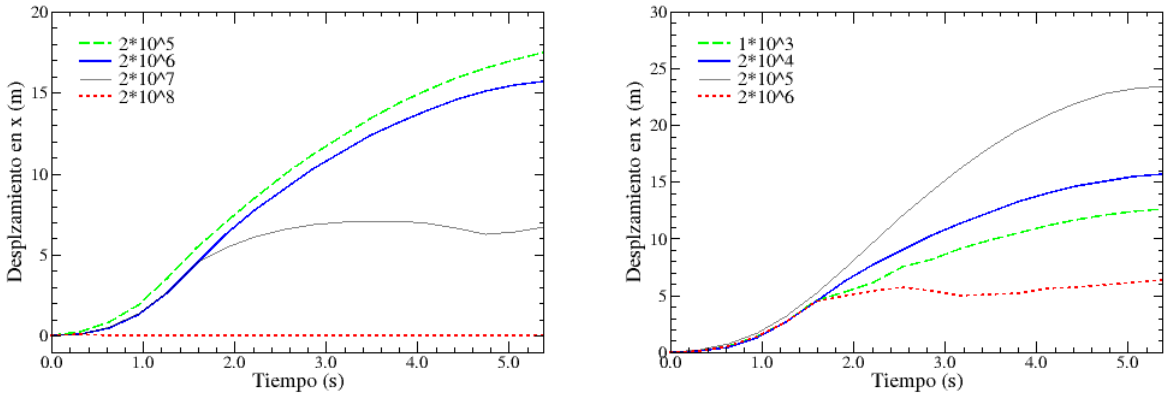
3.1.2. Calibración de los módulos k y η del modelo de fuerzas

El modelo de fuerzas del programa utiliza 3 módulos para calcular las fuerzas. En la fuerza normal se utiliza la rigidez k_n y el coeficiente de amortiguamiento η, mientras en la fuerza

tangencial se utiliza la rigidez k_t. La rigidez k_n y k_t definen con un mismo valor de k. En la Figura **3-5** se observa el desplazamiento en función del tiempo, variando k y η. En la Parte (a) se consideran 4 posibilidades de k: 2×10^{-5}, 2×10^{-6}, 2×10^{-7}, y 2×10^{-8}. El valor más alto de k no permite el desplazamiento en X ya que la fuerza tangencial es muy grande. Recuerde que el valor máximo de F_t es $F_n \tan\phi$ y F_n es directamente proporcional a k. A medida que k disminuye aumenta el desplazamiento del bloque y parece haber un límite entre 2×10^{-5} y 2×10^{-6}. El valor de $k = 2 \times 10^{-6}$ equivale a 20GPa, lo que se puede asimilar al módulo elástico del concreto, por esta razón se elige este valor para k.

En la parte (b) de la Figura **3-5** se varía el coeficiente de amortiguamiento η. Se evalúan 4 posibilidades de η: 1×10^{-3}, 1×10^{-4}, 1×10^{-5}, y 1×10^{-6}. Los resultados con 1×10^{-6} están en el rango del rebote sobreamortiguado, allí el amortiguador predomina en el cálculo de F_n y ocasiona que la Fuerza tangencial crezca a razón de $F_n \tan\phi$ y limite el movimiento. En los demás casos F_n es subamortiguada lo que implica que predomina el fuerza elástica. Nótese que a mayor η aumenta el desplazamiento tangencial. Esto tiene sentido ya que el amortiguamiento restringe la superposición entre bloques y con menor penetración la fuerza normal elástica disminuye y consecuentemente la fuerza friccional. El coeficiente η que se elige es de 1×10^{-4}, equivalente a $0{,}32N * s/m$.

Una vez seleccionadas las variables de simulación se resuelve un ejercicio de un bloque deslizando para compararlo con la solución la solución teórica. Anteriormente, en el capítulo 2.3, se presentó el ejemplo de un bloque deslizando, con el objetivo de verificar la aplicación del modelo de fuerzas tangente. Ese ejemplo se solucionó con un k de 20GPa, un η de $0{,}32N * s/m$ y un Δt de $3{,}16\mu s$: los mismos valores que se acaban de seleccionar. Además, el

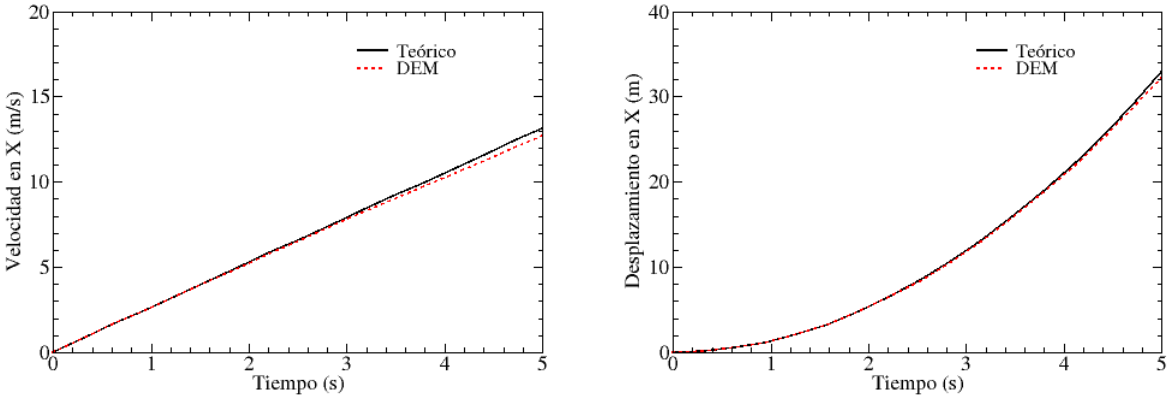


(a) Rigidez de los resortes k

(b) Coeficiente de viscosidad η

Figura 3-5.: Calibración de los módulos

algoritmo de integración que se usó fue el de Verlet Modificado. El resultado de ese problema se presenta en la Figura **3-6**. Los resultados son bastante buenos y se ajustan muy bien a la solución teórica.



(a) Velocidad en eje x

(b) Desplazamiento en eje x

Figura 3-6.: Verificación de los parámetros seleccionados

3.1.3. Análisis dimensional de unidades

En un programa de computador es común el uso de *unidades reducidas* con el objetivo de disminuir el error de precisión en los cálculos. Una unidad reducida es una unidad a la cual se aumenta -o disminuye- el número de cifras significativas para que se pueda operar con otras unidades. Por ejemplo, en el programa *Toppling_UN*, el modulo de los resortes que simulan las fuerzas es de $k = 20GPa = 2 * 10^{10}Pa$, mientras el paso de tiempo es de $\Delta t = 3{,}16\mu s = 3{,}16 * 10^{-6}s$. Hay 16 ordenes de magnitud entre k y Δt. Este hecho puede ocasionar grades errores numéricos, además de una posible omisión en las cifras significativas extremas. En este trabajo las unidades físicas se denominan "reales" mientras las unidades reducidas de computador se llaman "virtuales".

Los resultados del programa se dan en unidades virtuales y es necesario multiplicarlas por un factor de transformación a unidades reales. El factor de transformación se determina por medio del Teorema Π de Vaschy-Buckingham. Este teorema permite construir un conjunto de parámetros dimensionales a partir de las magnitudes físicas básicas: masa (M), longitud (L) y tiempo (T). El procedimiento para hallar los n parámetros adimensionales se detalla en el Anexo A.3. En la Tabla **3-1** se listan las variables dimensionales utilizadas en el programa. En total hay 11 magnitudes físicas diferentes, cada una con su respectivo factor de cambio entre unidades virtuales y reales.

Tabla 3-1.: Análisis dimensional de unidades

Unidad	símbolo	valor virtual	Factor de conversión	valor real
Rigidez resorte	k	$1(mPa)_v$	$1 * 10^7$	$1000Pa$
Masa	m	$1(g)_v$	$1 * 10^6$	$1000Kg$
Longitud	l	$1(m)_v$	$1 * 10^0$	$1m$
Tiempo	t	$1(s)_v$	$3{,}16 * 10^{-1}$	$0{,}316s$
Peso unitario	γ	$1(g/m^3)_v$	$1 * 10^6$	$1000kg/m^3$
Velocidad	v	$1(m/s)_v$	$3{,}16 * 10^0$	$3{,}16m/s$
Aceleración	a	$1(m/s^2)_v$	$1 * 10^1$	$10m/s^2$
Viscosidad	η	$1(g/s)_v$	$3{,}16 * 10^6$	$3{,}16kN * s/m$
Fuerza	F	$1(mN)_v$	$1 * 10^7$	$0{,}01MN$
Torque	τ	$1(mN * m)_v$	$1 * 10^7$	$0{,}01MN * m$
Energía	e	$1(mJ)_v$	$1 * 10^7$	$10kJ$

3.2. El algoritmo principal de "Toppling_UN"

El programa *Toppling_UN* fue escrito en lenguaje de programación C++ y desarrollado a partir del paradigma de la Programación Orientada a Objetos (POO). En su formulación se utilizan 3 librerías que son la base del algoritmo principal. Estos archivos pueden ser consultados en el Anexo B.2, de igual manera, en la tabla **3-2** se presentan las principales características de cada una de las librerías.

Tabla 3-2.: Librerías del programa principal

Nombre	Descripción
Clases.h	Hay dos clases en el programa: Block y Link. La clase Block se trató en el Capitulo 1.2.3. La clase Link contiene las funciones necesarias para que los bloques interactúen entre sí. Además, esta clase contiene las variables que permiten realizar un registro continuo de la posición de los bloques, de los vértices que están en contacto y de las longitudes de superposición.
Animacion.h	Contiene los procedimientos que permiten realizar las animaciones en formato ".gif". Crea un archivo de texto plano con una extensión ".gnu" que tiene los comandos necesarios para ser ejecutados en el programa *GNUPlot*.
vector.h	Permite realizar operaciones vectoriales y entre líneas paramétricas. Esta librería fue adaptada de las notas de clase del curso: Métodos de simulación física de la Universidad Nacional [28].

El código fuente del programa *Toppling_UN* se encuentra disponible en el Anexo B.1. Este código sigue el algoritmo que se muestra en el diagrama de flujo de la Figura **3-7**. Se puede observar que en esencia se conservan los tres pasos básicos del DEM: Calculo de Posición, Calculo de Fuerza y Calculo de Velocidad (pasos 4, 5, y 6). Estos pasos están dentro de un ciclo que va desde un tiempo t_0 hasta un tiempo t_f con incrementos de tiempo Δt. Los resultados del programa se entregan en forma de animación en el paso 4 o en forma numérica en el paso 8. En la Tabla **3-3** se describe en que consiste cada uno de los pasos del algoritmo.

Tabla 3-3.: Pasos del algoritmo principal

Paso	Descripción
1	Los datos que se ingresan en el programa son: geometría, módulos k y η, ángulo de fricción ϕ, peso unitario γ, paso de tiempo Δt y tiempo máximo t_f.
2	Se inician los atributos de cada bloque. Estos son: posición de los vértices, ubicación de los bordes $\vec{e}$, velocidad $\vec{v}$ y $\vec{\omega}$ en t_0, y las demás que se definieron en el capítulo 1.2.3.
3	Cada 1000 pasos de tiempo Δt se crea una imagen con la posición del sistema de bloques. Al finalizar el programa, las imágenes se unen en una animación con formato ".gif".
4	La posición y la orientación se calcula con base en las Ecuaciones del Capítulo 2.1: $r_{(t+\Delta t)} = r_{(t)} + v_{(t)}\Delta t + \frac{1}{2}\frac{F_{(t)}}{m}\Delta t^2$ (3-1) $\theta_{(t+\Delta t)} = \theta_{(t)} + \omega_{(t)}\Delta t + \frac{1}{2}\frac{\alpha_{(t)}}{I_o}\Delta t^2$ (3-2)
5	Las fuerzas y los torques se calculan con base en las unidades reológicas del Capítulo 2.3. En este paso hay un subproceso que incluye un algoritmo para la detección de contactos que se describirá en la siguiente sección.
6	La velocidad lineal y angular se calcula con base en las ecuaciones del Capítulo 2.1: $v_{(t+\Delta t)} = v_{(t)} + \frac{F_{(t)}}{m}\Delta t$ (3-3) $\omega_{(t+\Delta t)} = \omega_{(t)} + \frac{\alpha_{(t)}}{I_o}\Delta t$ (3-4)
7	Los resultados del programa se dan en un archivo de texto plano con extensión ".dat". Los resultados son las características dinámicas de cada bloque durante el tiempo de análisis.

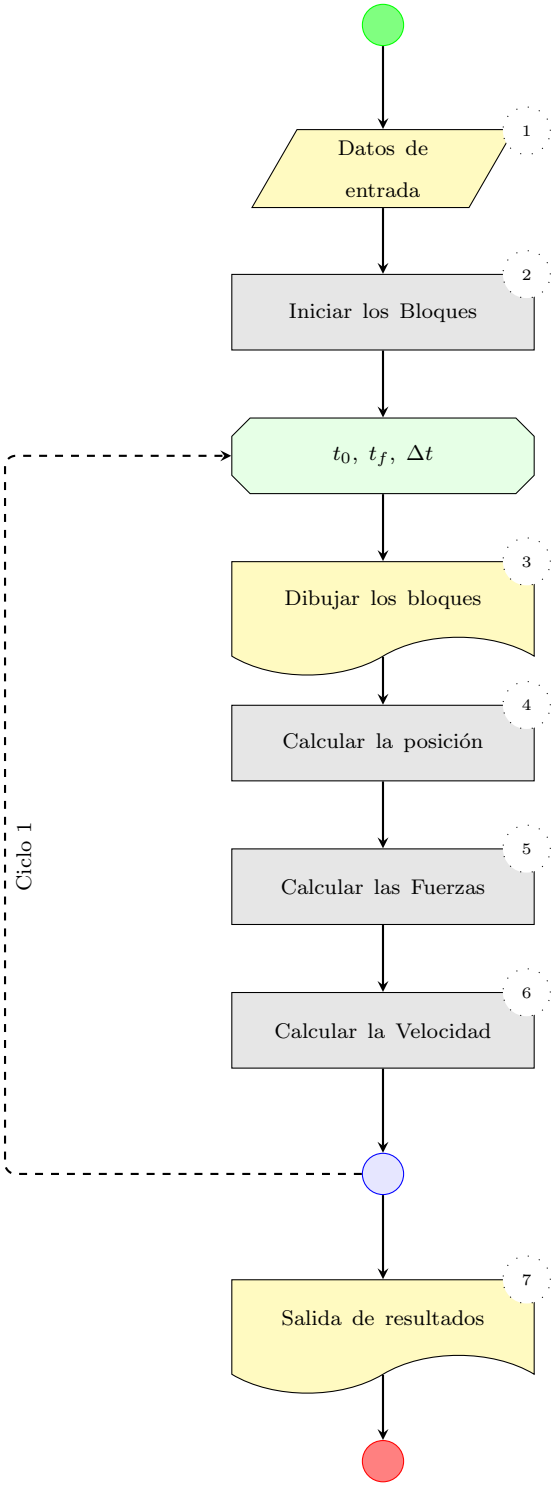


Figura 3-7.: Diagrama de flujo del programa principal

3.2.1. Algoritmo para el cálculo de fuerzas

El calculo de fuerzas se realiza de par en par, considerando todas las combinaciones posibles de a 2 bloques en el sistema. En el programa hay N bloques que están en movimiento y M bloques fijos que representan el *piso* del problema. En total hay $N + M$ bloques que se numeran desde 0 hasta $(N + M - 1)$. Una pareja de bloques está conformada por un bloque i que genera el choque y un bloque j que recibe el impacto. i y j son números enteros que conservan la numeración del respectivo bloque. Los vértices del bloque i se denominan k y se numera de 0 a 3. Para numerar los posibles choques del sistema de bloques se define una variable Id como:

$$Id = 4 * (N + M) * j + 4 * i + k \tag{3-5}$$

La variable Id surge del análisis de los tipos de choque se presentó en el Capítulo 2.2. En la Figura **3-8** se presenta el diagrama de flujo para el cálculo de fuerzas y detección de contactos. Los pasos del algoritmo se explican en la Tabla **3-4**. Nótese que el algoritmo inicia con un triple ciclo que analiza la posible colisión del vértice k del bloque i con el bloque j. El código fuente de este algoritmo puede ser consultado en la librería *clases* en el Anexo B.2.

Paso	Descripción
1	Ciclo que analiza la posible colisión de cada uno de los bloques del sistema como actuante.
2	Ciclo que analiza la posible colisión de cada uno de los bloques del sistema como pasivo.
3	Ciclo que analiza la posible colisión de cada uno de los vértices del bloque actuante.
4	Calcular el valor de Id de acuerdo con la Ecuación 3-5. Luego busca si hay contacto entre vértice k del bloque i y el bloque j. Para saber si hay contacto se sigue el procedimiento del Capítulo 2.2.
5	Si en el tiempo de análisis t hay contacto y también había en el instante anterior $t - \Delta t$, se calcula la Fuerza de Choque y su correspondiente Torque, de acuerdo a las Ecuaciones del Capítulo 2.3: $\vec{F_n} = k_n \vec{d_n} + \eta \vec{v_n}$ (3-6) $\vec{F_t} = \begin{cases} k_t \vec{d_t} & si \quad k\vec{d_t} \leqslant F_n \tan\phi \\ \vec{F_n} \tan\phi & si \quad k\vec{d_t} > F_n \tan\phi \end{cases}$ (3-7) donde $\vec{d_t}$, $\vec{d_t}$ y $\vec{v_n}$ se calculan con el procedimiento del capítulo 2.2.2.

Tabla **3-4** Continua en la página siguiente.

Paso	Descripción
6 y 8	En cada instante de tiempo, y para cada colisión Id, el programa rellena 3 matrices con la siguiente información: la identificación del borde $\vec{e}$, que pertenece al bloque j, por el cual se presentó el choque; el punto sobre $\vec{e}$ por donde se inició el contacto; y un registro si en el instante previo había contacto.
7	Si el contacto Id es nuevo, el programa evalúa sobre cual de los bordes e del bloque j ocurrió. Para determinar el borde se sigue el procedimiento de cruce de dos líneas paramétricas que se presentó en el Capítulo 2.2.
9	Calcular la Fuerza gravitacional. La fuerza Neta es la suma vectorial de la Fuerza Gravitacional más todas las fuerzas de choque que se produjeron. Asimismo, el Torque Neto es la suma de todos los torques.

Tabla 3-4.: Pasos del algoritmo para el cálculo de fuerzas y detección de contactos

3.3. Resultados del programa

Los bloques del talud de referencia de la Figura **3-1** se encuentran separados $2mm$ en la condición inicial del problema. Esto produce que la solución se tenga que desarrollar en dos pasos. En el primer paso se ejecuta un proceso inicial llamado *PreDEM*. Este proceso se realiza en un periodo de 500.000 pasos de tiempo, aproximadamente 1.6 segundos. Durante este tiempo no se considera el giro de los bloques: el sistema únicamente está sometido a fuerzas y no a torques. Este proceso es fundamental para el correcto funcionamiento del programa *Toppling_UN* ya que el proceso permite a los bloques encontrar una condición de equilibrio inicial. Si no se realiza el PreDEM, el sistema de bloques arranca con los resortes comprimidos lo que ocasiona excesos de fuerzas y resultados erróneos.

Luego del PreDEM los bloques ya están en contacto, la distancia de $2mm$ desaparece y se crea una condición inicial en equilibro de fuerzas pero no de momentos. Esta es el escenario inicial para realizar el análisis dinámico. El segundo paso de la solución es realizar el proceso DEM, incluyendo las fuerzas y los torques. El talud de referencia se analiza con los siguientes parámetros de simulación, que surgen de la calibración del Capítulo 3.1:

- Módulo de los resortes $k_n = k_t = 20Gpa$
- Coeficiente de amortiguamiento $\eta = 0{,}32N * s/m$
- Paso de tiempo $\Delta t = 3.16 \times 10^{-6}s$

Además, el ángulo de fricción que hay en los contactos es $\phi = 30°$.

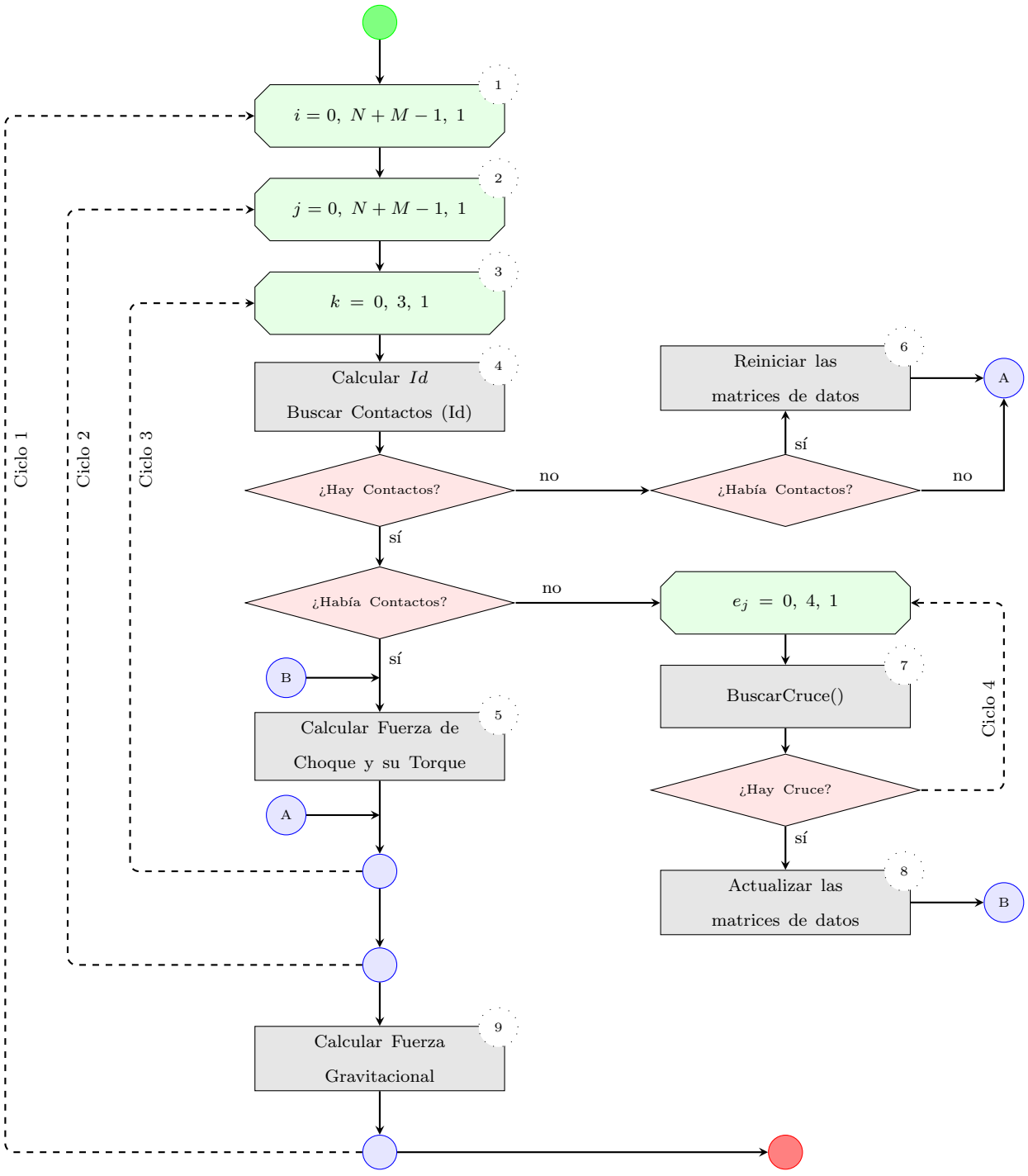


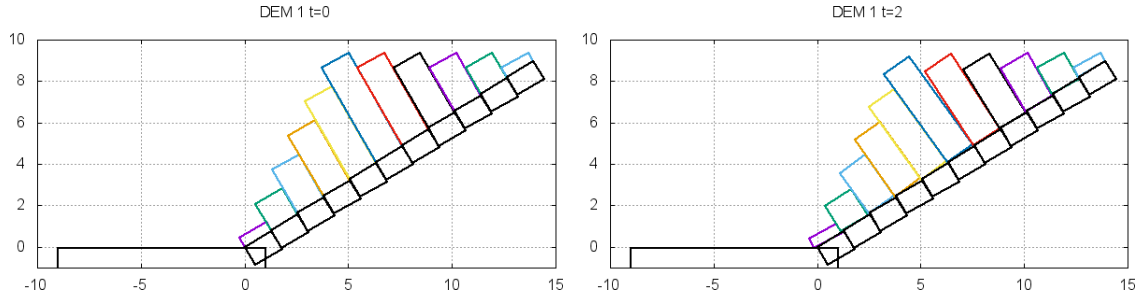
Figura 3-8.: Diagrama de flujo para el calculo de fuerzas

El tiempo final del PreDEM es el tiempo inicial del DEM. El análisis dinámico del talud se realiza desde un tiempo inicial $t_0 = 0$ hasta un tiempo final $t_f = 10{,}12s$ que equivalen a 3.2 millones de pasos de tiempo. Durante este periodo ocurre el proceso de volteo de bloques, como se observa en la Figura **3-9**. El proceso es muy rápido, se puede ver que a los 2.5s el sistema ha encontrado una nueva condición de equilibrio.

La condición inicial del problema, luego del proceso PreDEM, se encuentra en la parte (a) de la Figura **3-9**. El talud es inestable, así se mostró en el análisis de equilibrio límite por el método de Goodman, del Capítulo 1.1. Existen varios bloques con la tendencia a girar, lo que indica la posibilidad que se originen algunas fallas locales dentro del sistema. Los bloques inestables transmiten fuerzas que desencadenan el desplazamiento del bloque 0 y, en consecuencia, el movimiento en masa del talud. Los primeros bloques en movilizarse son aquellos en los cuales el momento actuante es superior al resistente; estos son los bloques: 3, 4, 5, 6, 7 y 8.

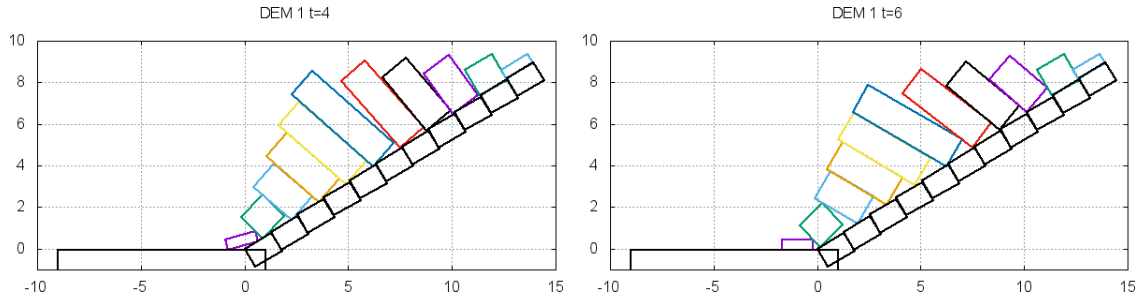
En la parte (b) y (c) de la Figura **3-9** se muestra el estado del talud cuando han transcurrido 0.63s y 1.26s, respectivamente. Se puede ver que el bloque 5 gira más rápido que el bloque 6 lo que ocasiona una separación momentánea de los bloques. El bloque 5 empuja al bloque 4, quien a su vez empuja a los que le siguen, produciendo un proceso de transmisión sucesiva de fuerzas. En los primeros segundo el bloque 5 es el que mayor velocidad angular presenta, lo que es de esperarse ya que es el más esbelto. Los bloques 6, 7 y 8 también giran pero con una velocidad angular menor. En este periodo el bloque 0 choca con la superficie horizontal produciendo un cambio en su orientación. Desde la parte (d) de la Figura **3-9** y hasta la parte (f), se puede ver que el movimiento del bloque 0 disminuye y, por tanto, también disminuye el movimiento de los bloques a su respaldo.

El movimiento de los bloques se presenta de forma individual, es decir, cada bloque se mueve de acuerdo con su balance de fuerzas. En las Figuras **3-10** y **3-11** se grafica la magnitud de la velocidad de traslación del centro de masa y de la fuerza resultante de cada bloque, ambas en función del tiempo. Los choques entre bloques ocasionan unos picos de fuerza que se reflejan en los cambios de velocidad. Note que el bloque 0 es el primero en detenerse, sin embargo el movimiento de los demás bloques continua. El último bloque en detenerse es el bloque 8 casi en simultaneo con el bloque 7. La máxima velocidad del proceso está cercana a los $1{,}8m/s$ y se presenta en los bloques 0 y 6, en los tiempos 1.3s y 2.3s respectivamente. Considérese que el centro de masa del bloque 6 se encuentra en una altura de aproximada de 7m, si este bloque cayera en caída libre su velocidad sería $\sqrt{2gh} = 11{,}7m/s$. Esto indica que la velocidad máxima que se presenta en el problema es del orden de 15 % de la velocidad de caída libre de un bloque a 7m de altura. A continuación se realiza un análisis del talud con diferentes condiciones de ϕ.



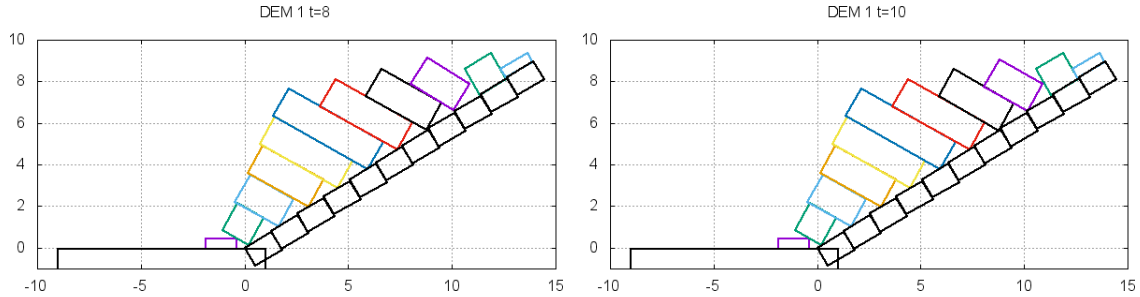
(a) Condición inicial, 0 ciclos, $t = 0s$

(b) 2×10^5 ciclos, $t = 0{,}63s$



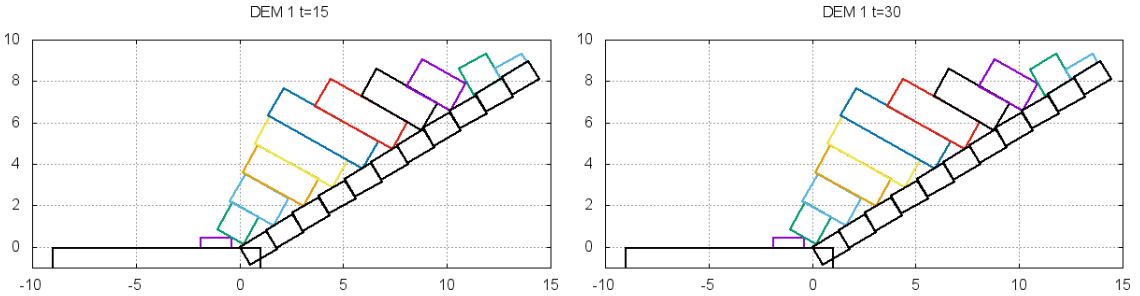
(c) 4×10^5 ciclos, $t = 1{,}26s$

(d) 6×10^5 ciclos, $t = 1{,}90s$



(e) 8×10^5 ciclos, $t = 2{,}53s$

(f) 1×10^6 ciclos, $t = 3{,}16s$



(g) 1.5×10^6 ciclos, $t = 4{,}74s$

(h) Condición final, 3×10^6 ciclos, $t = 9{,}49s$

Figura 3-9.: Animación de la dinámica de *toppling* para un talud con $\phi = 30°$

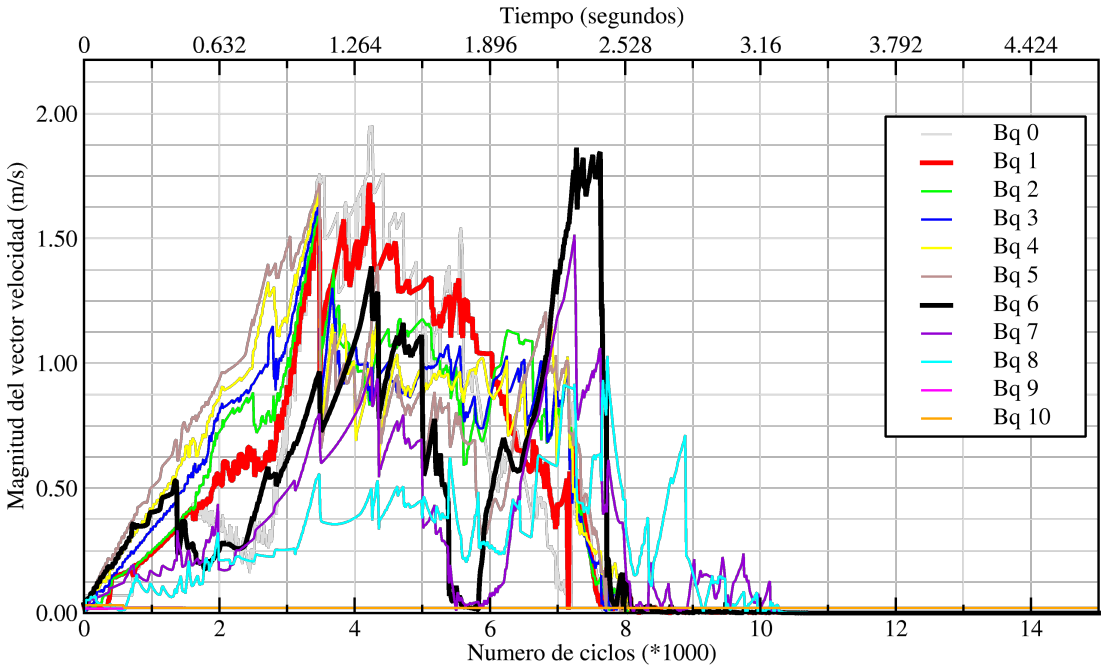


Figura 3-10.: Velocidad de los bloques del talud

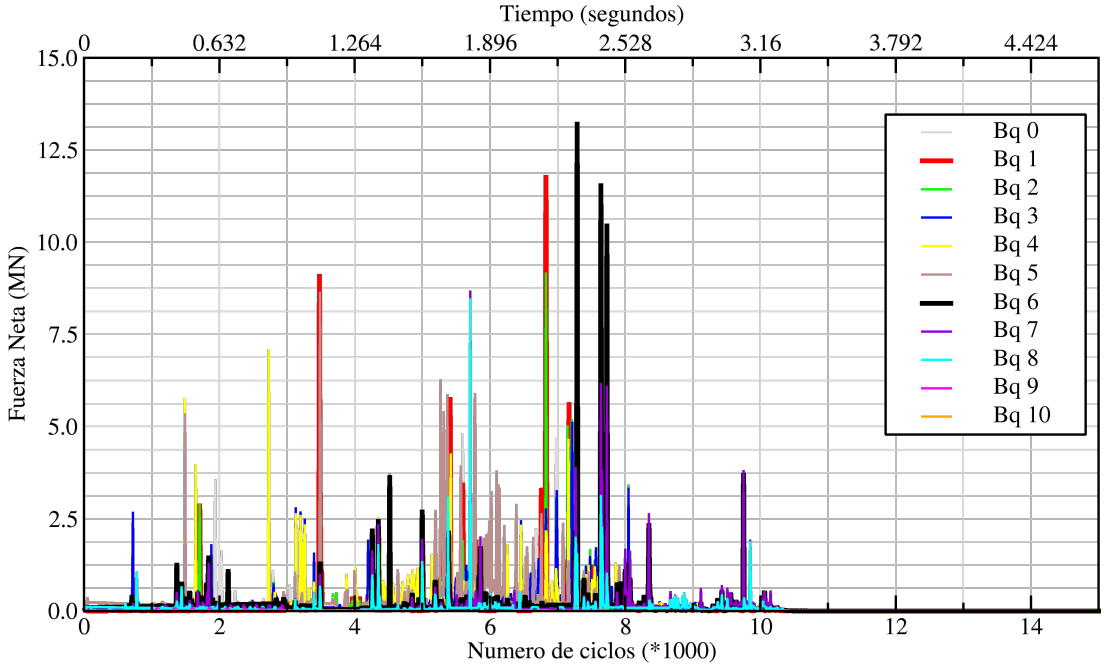
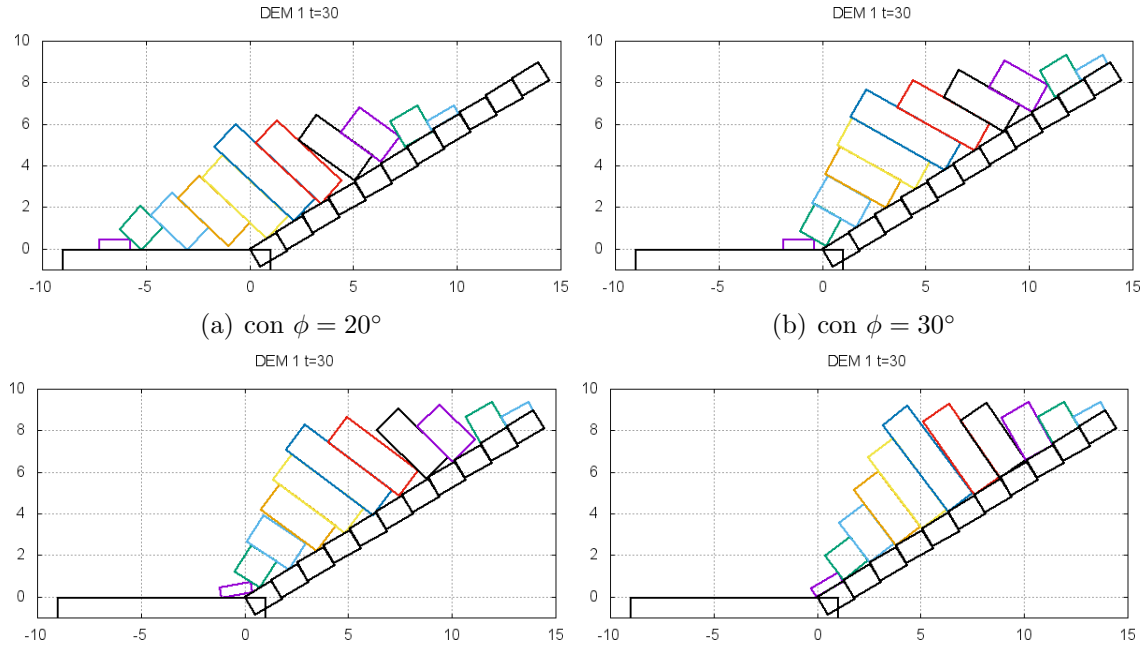


Figura 3-11.: Fuerza Neta de los bloques del talud

3.4. Análisis de resultados

Con el objetivo de conocer el comportamiento del talud para diferentes condiciones de fricción, se evalúan los escenarios con $\phi = 20°$, $\phi = 30°$, $\phi = 40°$ y $\phi = 45°$. En la Figura **3-12** se muestra la posición final de los bloques luego de que transcurran 10 segundos para cada escenario. Se observa que el mayor desplazamiento de los bloques se produce con el ϕ de 20°. En el escenario de 45° los bloques se movilizan muy poco, sin embargo se observa que sí ocurre un pequeño desplazamiento.

Para evaluar la exposición a la que están expuestos los posibles elementos cercanos al talud, se analiza la distancia de viaje del bloque 0. Esta distancia se grafica en función del tiempo en la Figura **3-13**. Cuando el talud tiene un ϕ de 20° el desplazamiento del bloque 0 es de 6.5m. Con ϕ de 30° es de 1.1m. Con ϕ de 40° es de 0.45m y con ϕ de 45° es de 0.05m. Ahora bien, nótese que en todos los escenarios el bloque 0 se desplaza alguna cantidad. Es importante definir un margen de desplazamiento admisible ya que, dependiendo del tipo de obras expuesta, incluso un desplazamiento de 1cm podría ocasionar grandes afectaciones a las estructuras, por ejemplo en líneas de transmisión de hidrocarburos. Además, es importante señalar que los resultados muestran que un descenso de 10° en el ángulo de fricción provoca un incremento en el desplazamiento de más de 5m.



(a) con $\phi = 20°$ (b) con $\phi = 30°$

(c) con $\phi = 40°$ (d) con $\phi = 45°$

Figura 3-12.: Condición final de un talud con varios ángulos de fricción ϕ

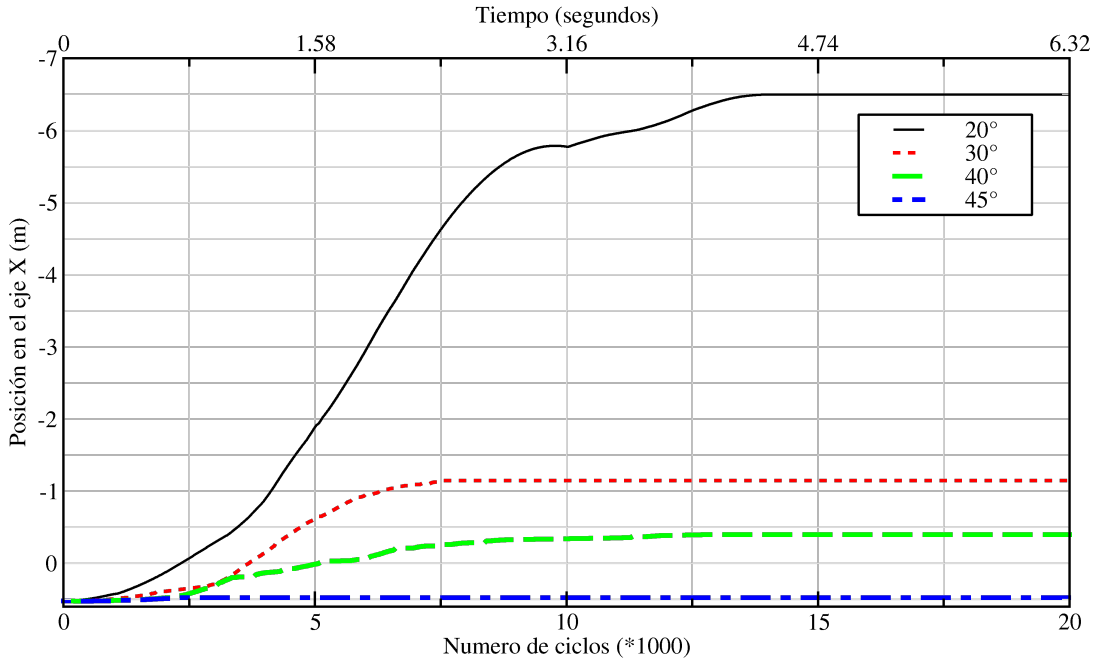


Figura 3-13.: Trayectoria en X del bloque 0 para diferentes ϕ

La velocidad y la fuerza con que el bloque 0 se desplaza definen también el grado de afectación de las estructuras expuestas. En las Figuras **3-14** y **3-15** se grafican estas variables en función del tiempo. La mayor velocidad se presenta en el escenario donde ϕ es igual a 20°. En el caso de $\phi = 45°$ el bloque 0 se mueve un corto periodo de tiempo pero rápidamente todo el sistema encuentra un equilibrio. Nótese que las mayores fuerzas en x, que se presentan en el bloque 0, ocurren con $\phi = 40°$ y no con $\phi = 45°$, como se estaría tentado a pensar. Esto ocurre porque con 45° la fuerza resistente de cada bloque es mayor que en el caso de 40° y, por tanto, la resultante que llega al bloque 0 es menor. Con ϕ menores a 30° los bloques experimentan grandes desplazamientos y las fuerzas se disipan con el movimiento.

Nótese que en el caso de $\phi = 20°$ se presenta un descenso abrupto en la velocidad cuando hay una colisión en el tiempo 3.16s. Esto es un ejemplo de de la influencia de los choques en la velocidad de los bloques. Durante el proceso hay bloques que experimentan fuerzas superiores a $10MN$, toda esta fuerza se concentra en los vértices y seguramente, en la realidad, las esquinas de los bloques se fracturan. Lo que no sucede en el programa.

El programa ofrece buenos resultados cuando el ángulo de fricción ϕ de los bloques es menor o igual a 45°. Con valores mayores la fuerza de choque tangente, $F_t = F_n \tan\phi$, resulta ser mayor que F_n.

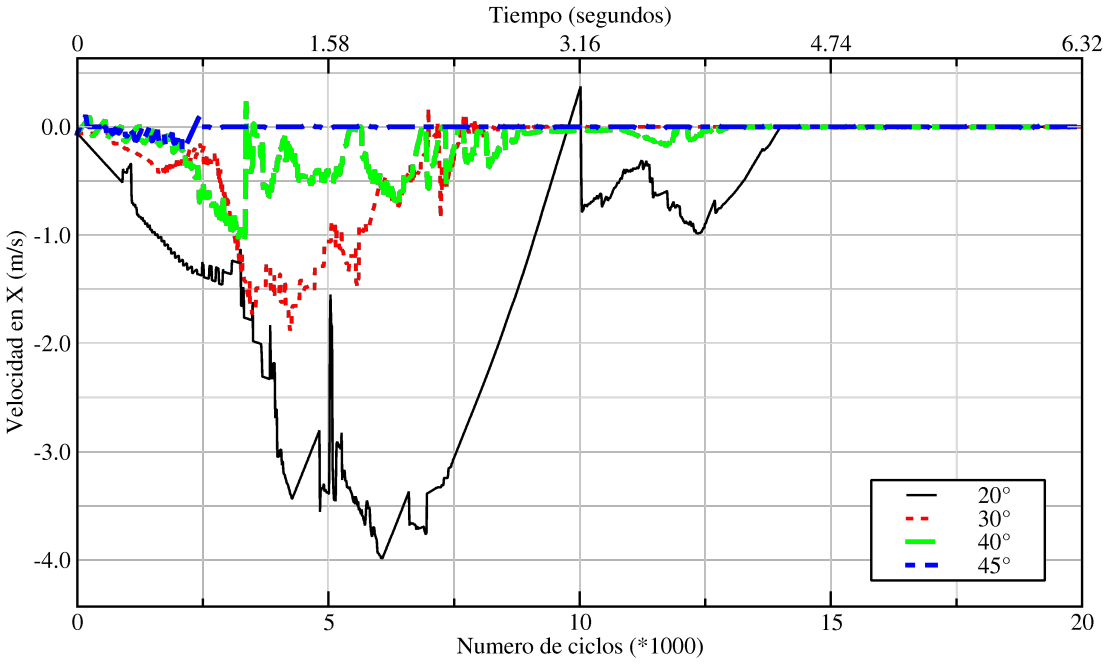


Figura 3-14.: Velocidad en X del bloque 0 para diferentes ϕ

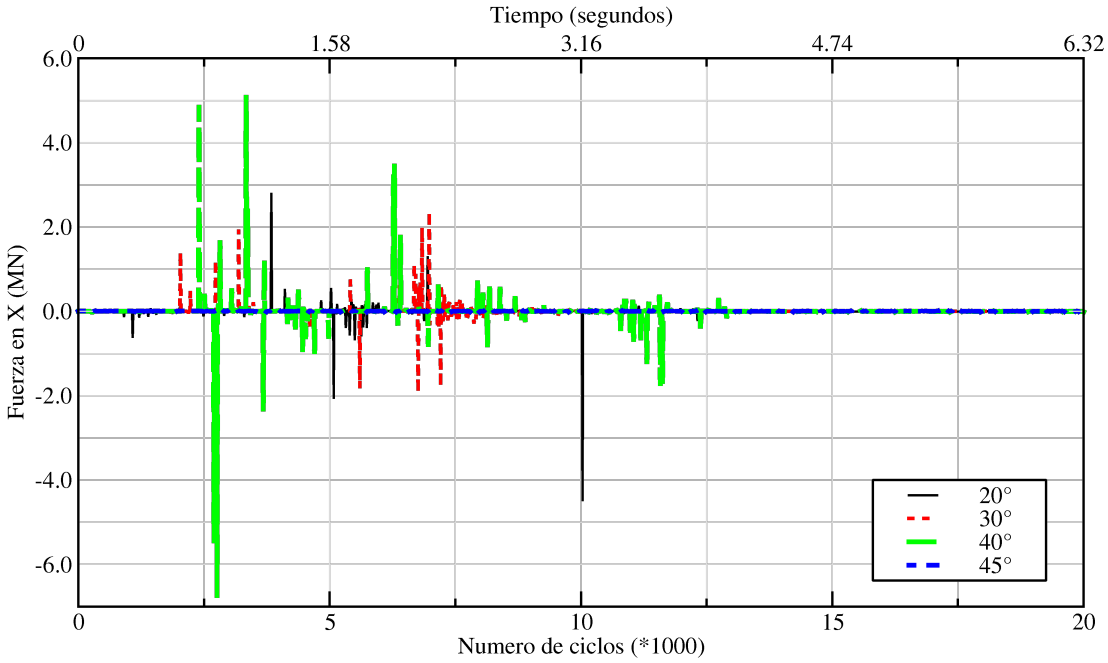


Figura 3-15.: Fuerza en X del bloque 0 para diferentes ϕ

4. Conclusiones

En este trabajo se diseñó un programa de computador basado en el Método de los Elementos Discretos (DEM), que permite estudiar el movimiento de un sistema de bloques. El programa lleva por nombre *Toppling_UN* y es capaz de simular con precisión la dinámica de un macizo rocoso sometido al proceso de volteo de bloques.

El método DEM es un procedimiento numérico utilizado para analizar problemas en medios discontinuos, como es el caso de los problemas de la mecánica de rocas. Es indudable que, en este caso, los Elementos Finitos y otras técnicas fundamentadas en el continuo, no son suficientes y, en consecuencia, el DEM se convierte una alternativa de análisis muy útil. Existen varias versiones del método, las más populares son el DEM y el Análisis de Deformaciones Discontinuas (DDA). Ambas versiones emplean el concepto de *elemento discreto* y, en algunos textos, se considera que el DDA es una variación del método DEM. En este trabajo se utilizó la versión DEM y se mostró que es un procedimiento muy sencillo que marcha en el tiempo y no recurre a sistemas de ecuaciones implícitos, como sí sucede en el DDA.

En mecánica de rocas, y en general en geotecnia, es común realizar los análisis de estabilidad de taludes por medio de métodos basados en el equilibrio límite. Esto desde una condición estática, dejando de lado el comportamiento dinámico que se presenta durante una eventual falla. Lo que se pudo observar en el desarrollo de este trabajo es que el método DEM utiliza un equilibrio dinámico, es decir, en cada incremento de tiempo realiza un balance de fuerzas y, con base en la fuerza resultante, mueve los bloques a una nueva posición. En este método el movimiento se calcula por medio de la Segunda Ley de Newton y cada uno de los elementos del sistema se desplaza de forma autónoma, de acuerdo a su balance de fuerzas.

La condición de inicio en un programa tipo DEM es un problema particular que se analiza por aparte. No es conveniente iniciar desde una condición en movimiento ni en condición de falla. Siempre es necesario garantizar una condición de equilibrio estático en el tiempo 0 del problema. Para lograr esto se realiza un precalculo, en el cual se ubican todos los bloques en su posición, sin fuerzas diferentes a su peso propio, y se deja avanzar el programa hasta que el sistema esté en equilibrio. Luego, para desencadenar el movimiento, basta con disminuir las propiedades de resistencia de los bloques o aumentar las acciones en uno o en todos los bloques. Este procedimiento inicial es muy importante para evitar exceso de fuerzas que

puedan poner en riesgo la estabilidad numérica de los cálculos.

En el modelo de fuerzas que se usa en el programa *Toppling_UN* se utilizan 3 parámetros para simular la interacción entre bloques: rigidez de los resortes k, coeficiente de amortiguamiento η y ángulo de fricción ϕ. Este último, el ángulo de fricción ϕ, es una propiedad de resistencia de la roca y puede ser medida por medio de ensayos. Por su parte, los parámetros k y η son puramente matemáticos y no se relacionan con ninguna propiedad del material. Caso similar ocurre con el paso de tiempo Δt, que es una variable numérica que se utiliza en las Series de Taylor. Todas estas variables fueron analizadas de forma independiente en este trabajo, obteniendo como resultado su correcta implementación en el código del programa.

El programa *Toppling_UN* fue validado por medio de varios ejercicios de la dinámica que cuentan con una solución analítica. En primer lugar, se realizaron varias pruebas de rebote elástico y amortiguado, con el propósito de comprobar el carácter viscoelástico de la Fuerza normal de choque. Luego, se analizaron múltiples escenarios de un bloque deslizando, de allí fue posible comparar la solución del programa con una solución teórica. Finalmente, una vez se definió un escenario de referencia para *toppling*, se comparó la respuesta dinámica del programa con el análisis de equilibrio estático propuesto por Gooodman. Se encontró que, como era de esperarse, los mismos bloques inestables en el análisis estático fueron los que presentaron movimiento. Además, los resultados del proceso de volteo muestran situaciones absolutamente lógicas. Por ejemplo, se encontró que a mayor ángulo de fricción, menor es el desplazamiento del talud. Así mismo, la velocidad de los bloques siempre permanece por debajo de la velocidad de un bloque cayendo en caída libre($\sqrt{2gh}$). De hecho, como se mostró, la velocidad máxima que se alcanzó en el sistema equivale al 20 % de la velocidad de un bloque que cae libremente desde 8m de altura.

La aplicación del programa mostró que la dinámica de los bloques produce una serie de movimientos locales, dentro del movimiento general del talud. Por ejemplo, cuando los bloques del talud tienen un ϕ de 45°, el equilibrio estático concluyó que el talud es estable. Sin embargo, el equilibrio dinámico muestra que varios de los bloques, incluyendo el bloque de la pata del talud, presentan pequeños desplazamientos. Así pues, asumir la condición estática como inicial es un error en el que no debemos incurrir. Nuevamente se resalta el hecho que la condición inicial se da cuando el sistema por sí solo encuentra una condición de equilibrio.

Los resultados dinámicos que se obtienen con el programa son muy importantes para realizar análisis cuantitativos de vulnerabilidad y de riesgo, en los elementos expuestos al movimiento del talud. No es común que en la práctica se realicen análisis que describan las características del movimiento de un talud. Por el contrario, el desplazamiento y la velocidad de un movimiento en masa se suele estimar a través de información cualitativa o semicuantitativo.

Con este trabajo, se trató de mostrar que métodos como el DEM son herramientas de fácil uso y brindan información muy precisa acerca del movimiento de un talud.

Uno de los mayores logros de este trabajo es la construcción de un programa propio de código abierto. Esto, en términos académicos, resulta muy práctico para comprender e interiorizar el funcionamiento del Método de los Elementos Discretos. El algoritmo principal del programa y sus respectivas librerías son muy sencillas de entender y por tanto de modificar. Quizás lo más interesante del algoritmo es el procedimiento para identificar contactos y calcular la longitud de superposición. Este procedimiento se puede utilizar para cualquier pareja de bloques que tengan 3 o más lados, siempre y cuando su forma sea la de un polígono convexo. Esto es una ventaja muy grande del programa ya que, con pequeñas modificaciones, es posible utilizarlo para analizar otros problemas además del de *toppling*. Por ejemplo, el código sirve de base para diseñar otros programas que simulen una caída de rocas o algún ensayo de resistencia de bloques fracturados.

Adicionalmente, del algoritmo para identificar contactos vale la pena mencionar que este considera todas las combinaciones de choque posibles entre dos bloques: borde-borde, vértice-borde y vértice-vértice. Este proceso lo hace para cada una de las posibles parejas de bloques en el sistema. Así pues, las posibilidades de colisión son bastantes. Este procedimiento resulta ser muy eficiente en problemas con pocos bloques pero en sistemas más complejos el uso de recursos computacionales lo hace ineficiente. Esta situación se presenta como una oportunidad de mejora para futuros trabajos.

El programa fue diseñado para una condición bidimensional, sin embargo todas las variables vectoriales se escribieron en 3 dimensiones; con lo cual ya hay un avance para llevar el programa a 3D. Los bloques del programa se comportan como sólidos rígidos, esto quiere decir que no se deforman ni se fracturan. Para incluir deformación basta con agregar una malla de elementos finitos en cada uno de los bloques, lo que aumentaría el tiempo de ejecución del programa pero permitiría aproximar aún más la solución a la realidad. En cuanto a las fracturas, es posible utilizar la superposición de bloques como un área de acumulación de esfuerzo y así identificar cuando se alcancen los esfuerzos de falla. Esta es otra de las posibilidades para trabajos que se desarrollen a partir de este.

Por último, cabe mencionar que en el programa se usó un conjunto de fuerzas que simulan la fricción, la repulsión y el amortiguamiento entre bloques. También se podrían modelar otras fuerzas temporales como las producidas por voladuras, impactos, sismo o flujos; y fuerzas permanentes como el empuje hidroestático o como la cohesión. Con todo esto es posible concluir que el campo de los Elementos Discretos tiene muchas opciones para la mecánica de rocas y nos invita a realizar análisis desde una perspectiva dinámica.

Bibliografía

[1] Servicio Geológico Colombiano. *Guía metodológica para estudios de amenaza, vulnerabilidad y riesgo por movimientos en masa*. Bogotá, Colombia, 2015. ISBN: 978-958-99528-5-6.

[2] Claudia Ximena Basto Gonzales. Desplazamiento viscoplástico de taludes fallados. Master's thesis, Universidad Nacional de Colombia, Bogotá, Colombia, 2009.

[3] Raúl Hernán Moreno Montoya. Cinética de taludes en suelos compresibles viscoplásticos. Master's thesis, Universidad Nacional de Colombia, Bogotá, Colombia, 2016. http://www.bdigital.unal.edu.co/59357/.

[4] David Alejandro Aponte Rojas. Simulación de medios granulares. Master's thesis, Universidad Nacional de Colombia, Bogotá, Colombia, 2018. http://www.bdigital.unal.edu.co/70381/.

[5] Andrés Gerardo Eraso Baena. Respuesta dinámica de taludes en macizos rocosos, aplicando el método de análisis de deformación en medios discretos d.d.a. Master's thesis, Universidad Nacional de Colombia, Bogotá, Colombia, 2004.

[6] Alejandro Edilberto Granados Becerra. Resistencia de sistemas fracturados mediante análisis de deformaciones en medios discretos, dda. Master's thesis, Universidad Nacional de Colombia, Bogotá, Colombia, 2018. http://www.bdigital.unal.edu.co/65734/.

[7] Peter A. Cundall. A computer model for simulating progressive, large scale movements in blocky rock systems. In *Int. Symp. Rock Fracture*, volume 1, Nancy, France, 1971. International Society for Rock Mechanics.

[8] B. J. Alder and T. E. Wainwright. Studies in molecular dynamics. i. general method. *The journal of chemical physic*, 31(2):459–466, 1959. doi: 10.1063/1.1730376.

[9] Jian-Hong Wu. Seismic landslide simulations in discontinuous deformation analysis. *Computers and Geotechnics*, 37(5):594–601, 2010. doi: 10.1016/j.compgeo.2010.03.007.

[10] Chao-LungTang et al. The tsaoling landslide triggered by the chi-chi earthquake, taiwan: Insights from a discrete element simulation. *Engineering Geology*, 106(1-2):1–19, 2009. doi: 10.1016/j.enggeo.2009.02.011.

[11] William Fernando Oquendo. Simulación de la compresión edométrica de un material granular y determinación de la relación de vacíos. Master's thesis, Universidad Nacional de Colombia, Bogotá, Colombia, 2007.

[12] Sergio Andres Galindo Torres and Jose Daniel Muñoz Castaño. Simulation of the hydraulic fracture process in two dimensions using a discrete element method. *Phys. Rev. E*, 75(6):066109, 2010. doi: 10.1103/PhysRevE.75.066109.

[13] Ling Qian Choo et al. Hydraulic fracturing modeling using the discontinuous deformation analysis (dda) method. *Computers and Geotechnics*, 76:12–22, 2016. doi: 10.1016/j.compgeo.2016.02.011.

[14] Xikui Li, Yuntian Feng, and Graham Mustoe. *Proceedings of the 7th International Conference on Discrete Element Methods*. Springer Proceedings in Physics, 1 edition, 2017. doi : 10.1007/978-981-10-1926-5.

[15] M Reza Salami, D C Banks, and Gen hua Shi. *Discontinuous deformation analysis (DDA) and simulations of discontinuous media: proceedings of the first International Forum on Discontinuous Deformation Analysis (DDA) and Simulations of Discontinuous Media*. TSI Press, 1 edition, 1996. ISBN: 9780962745171.

[16] J. Davis Varnes. *Landslide, anaysis and control*, chapter Slopes movement types and processes. Transportation research board, 1978. Special report 176, ISBN: 9780309028042.

[17] Richard E. Goodman and Jhon W. Bray. Toppling of rock slopes. In *Proceedings of the specialty conference on Rock Engineering for foundations and Slopes*, Boulder, Colorado, 1976. American Society of Civil Engineers.

[18] Jhon Ashby. Sliding and toppling modes offailure in models andjointed rock slopes. Master's thesis, University of London, Imperial College, London, England, 1971.

[19] Charles A. Kliche. *Rock slope stability*, chapter kinetic slope stability analysis of toppling failure. Society for Mining, Metallurgy, and exploration, Inc., 1999. ISBN 0-203-49908-5.

[20] Gen-hua Shi. *Discontinuous deformation analysis—a new numerical model for the statics and dynamics of block systems*. PhD thesis, University of California, Berkeley, California, USA, 1988.

[21] Lanru Jing and Stephansson Ove. *Fundamentals of discrete element methods for rock engineering*, volume 85. Elsevier Science, 1 edition, 2007. ISBN: 978-0-08-055185-2.

[22] H. Hatzor Yossef, Ma Guowei, and Gen-hua Shi. *Discontinuous Deformation Analysis in Rock Mechanics Practice*, volume 5. International Society for Rock Mechanics, 1 edition, 2018. ISBN: 978-1-315-68703-2.

[23] Sergio M Fernández. *fundamentos del diseño y la programación orientado a objetos*. McGraw-Hill, 1 edition, 1995.

[24] Thomas G. Mezger. *The Rheology Handbook*. Vincentz Network, 4 edition, 2014. ISBN: 978-3-86630-650-9.

[25] Tito Flórez Calderón. *Métodos numéricos que se deben saber*. Universidad Nacional de Colombia, 2016. eISBN: 9789587758320.

[26] Ling Qian Choo et al. The leapfrog method and other "symplectic" algorithms for integrating newton's laws of motion. *physics review*, 155/242, 2013.

[27] Inc Itasca Consulting Group. *Manual UDEC version 6.0(Universal Distinct Element Code)*. web https://www.itascacg.com/software/udec.

[28] Universidad Nacional de Colombia. Notas de clase: Métodos de simulación física. Código 2020174-1, Departamento de Física - Programa de Maestría en Física, 2017-II.

A. Anexo 1: Procedimientos y memorias de cálculo

A.1. Solución de un problema de toppling en condición estática

Siguiendo el método de Goodman [17] se desarrolló un programa simple para calcular la estabilidad estática de un sistema de bloques. Los resultados del programa son la fuerzas que se transmite de bloques a bloque y la definición de los bloque que deslizan, deslizan-voltea o voltean. En la Figura **A-1** se presentan las posibilidades de movimiento del talud de referencia. A continuación se presenta el código del programa que se utilizó.

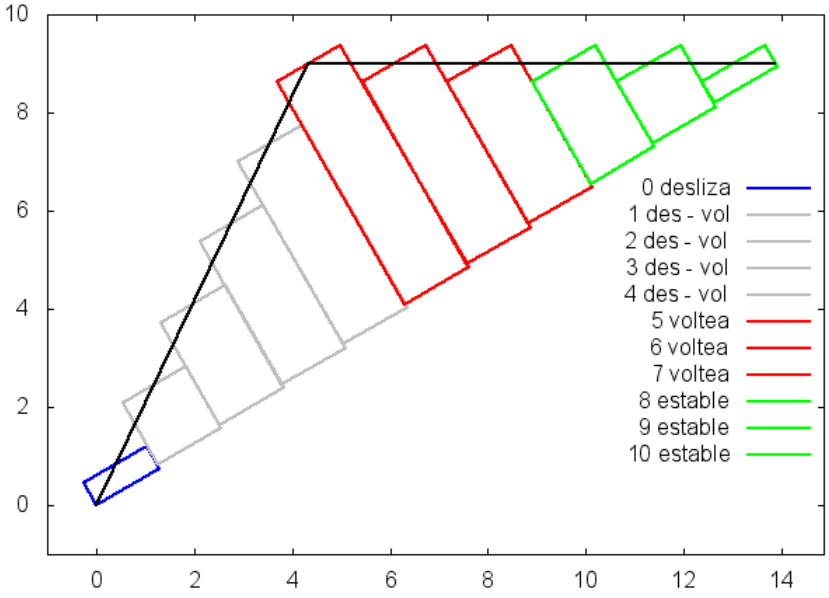


Figura A-1.: Posibilidades de movimiento en condición estática

A.1.1. Código del programa para la solución estática

```
1  //*********************************************************************
2  //Toppling por equilibrio limite - Metodo de Goodman
3  //Parte 1: Programa principal
4  //*********************************************************************
5  /*Notas:
6      todos los bloque son de igual ancho 't'
7      No hay cohesion entre bloques
8      los bloques no se fracturan
9      */
10
11
12 //---------------------------------------
13 //----------Librerias---------------------
```

```
14  //----------------------------------------
15  #include <iostream>
16  #include <cmath>
17  #include <fstream>
18  #include "Animacion.h"
19  #include "Clases.h"
20  #include "Resultados.h"
21
22  using namespace std;
23
24
25  //----------------------------------------
26  //----------Lista de variable globales----
27  //----------------------------------------
28  int N;      //numero de bloques
29  double Beta;
30  double Theta;
31  double Alpha1;
32  double Alpha2;
33  double AlturaH;
34  double Gamma;       //en kN/m3
35  double Phi;
36  double xi=0;
37  double yi=0;
38  double X,Y;
39
40
41  //----------------------------------------
42  //--------Funciones del programa----------
43  //----------------------------------------
44  void InicieVariables(double &Lea){
45      Lea=2;
46      cout<<"Si desea ejecutar ejemplo presione 0, si no, presione 1  ";
47      while (Lea!=0 && Lea!=1)    cin>>Lea;
48
49      if(Lea==0){     //indica que se toman los valores del ejemplo
50          N=11;       //numero de bloques
51          Beta=30.0*M_PI/180;
52          Theta=3.0*M_PI/180;
53          Alpha1=64.31*M_PI/180;
54          Alpha2=0.0*M_PI/180;
55          AlturaH=9.0;
56          Gamma=26;       //kN/m3
57          Phi=35*M_PI/180;
58      }
59
60      else{           //indica que el usuario ingresara datos
61          cout<<"Por favor ingrese la siguiente informacion: "<<endl;
62          cout<<"Numero de bloques ";          cin>>N;
63          cout<<"Angulo Beta (grados)";        cin>>Lea;Beta=Lea*M_PI/180;
64          cout<<"Angulo Theta (grados)";       cin>>Lea;Theta=Lea*M_PI/180;
65          cout<<"Angulo Alpha1 (grados)"; cin>>Lea;Alpha1=Lea*M_PI/180;
66          cout<<"Angulo Alpha2 (grados)"; cin>>Lea;Alpha2=Lea*M_PI/180;
67          cout<<"Altura de Talud (metros)";    cin>>Lea;AlturaH=Lea;
68          cout<<"Peso Unitario de la roca ";  cin>>Gamma;
69          cout<<"Angulo de friccion (grados)";        cin>>Lea;Phi=Lea*M_PI/180;
70      }
71  }
72
```

```
73
74  //----------------------------------------
75  //----------Programa Principal------------
76  //----------------------------------------
77  int main(void){
78      system("cd >Dir.inf");       //lee la ubicacion en el computador
79      system("del *.dat");         //borra la anterior ejecucion
80      system("del *.gnu");         //borra la anterior ejecucion
81      system("cls");
82
83      double Lea;     //variable para leer informacion del usuario
84
85      do{     //ciclo para reiniciar el programa
86
87      InicieVariables(Lea);
88      Block Bloques[N];
89      Link Fuerza;
90      ofstream OFuerzasDat("Fuerzas.dat");
91      int volteo=0;
92      double Ps=0, Pt=0;
93
94      for(int i=0;i<N;i++)
95          Bloques[i].Inicie(i, AlturaH,Beta,Theta,N,Alpha1,Alpha2,xi,yi,X,Y,Gamma);
96
97      //--------
98      //Analisis de deslizamiento unitario
99      for(int i=0;i<N;i++)
100         if(Fuerza.Pslide(Bloques, i, 0,Beta,Phi)>0){
101             cout<<endl<<"El bloque "<<i<<" desliza en el analisis unitario"<<endl;
102             system("pause");
103             return 1;   //no es posible realizar el analisis estatico
104         }
105
106     //--------
107     //Analisis de Volteo unitario - halla el primero en voltear
108     for(int i=N-1;i>=0;i--)
109         if(Bloques[i].Gett()/Bloques[i].Geth()<tan(Beta)){volteo=i; i=0;}
110
111     //--------
112     //Analisis de Toppling
113     OFuerzasDat<<"#Fuerza por delizamiento y toppling en kN"<<endl;
114     OFuerzasDat<<"#Bloque - Ps - Pt - P"<<endl;
115     for(int i=N-1;i>=0;i--){
116         if(i>volteo) {Ps=0; Pt=0; Bloques[i].AgregueP(Ps,Pt);}
117         else{
118             Ps=Fuerza.Pslide(Bloques, i, Bloques[i+1].GetP(),Beta,Phi);
119             Pt=Fuerza.Ptoppling(Bloques, i, Bloques[i+1].GetP(),N,Beta,Phi);
120             Bloques[i].AgregueP(Ps,Pt);
121         }
122         OFuerzasDat<<i<<" "<<Ps<<" "<<Pt<<" "<<Bloques[i].GetP()<<endl;
123     }
124
125     //--------
126     //Resultados graficos
127     InicieDibujo(X,Y);
128     for(int i=0;i<N;i++)InicieColor(Bloques[i].GetID(),i);
129     FinInicieDibujo();
130     for(int i=0;i<N;i++) Bloques[i].Dibujese(Beta);
131     DibujeTalud(xi, yi,X,Y, AlturaH, Alpha1, Beta, Theta);
```

```
132       Resultados();
133       system("Dibujar.cmd");
134       system("cls");
135
136       cout<<"Si desea reiniciar presione 0, si desea salir presione 1 ";cin>>Lea;
137
138   }while(Lea==0);
139
140   return 0;
141   }
142
143
144
145
146
147   //*********************************************************************
148   //Toppling por equilibrio limite - Metodo de Goodman
149   //Parte 2: Clase del programa principal
150   //*********************************************************************
151
152
153   //----------------------------------------
154   //----------Librerias---------------------
155   //----------------------------------------
156   #include <iostream>
157   #include <cmath>
158   #include <fstream>
159
160   using namespace std;
161
162
163   //----------------------------------------
164   //----------Clase Block-------------------
165   //----------------------------------------
166   class Block{
167   private:
168       int Id;                 //0:estable 1:desliza 2:volteo 3:desliza y voltea
169       double t,h,b,W,P;       //unidades depende de Gamma
170       double x1, y1;          //vertice inferior izquierdo
171   public:
172       void Inicie(    int i, double AlturaH, double Beta, double Theta, int N,
173               double Alpha1,double Alpha2, double xi, double yi,
174                                           double &X, double &Y,       double Gamma);
175       void AgregueP(double Ps, double Pt);
176       void Dibujese(double Beta);
177       int GetID(void){return Id;};
178       double Geth(void){return h;};
179       double Gett(void){return t;};
180       double GetP(void){return P;};
181       double Getxi(void){return x1;};
182       double Getyi(void){return y1;};
183       friend class Link;
184   };
185
186   void Block::Inicie( int i, double AlturaH, double Beta, double Theta, int N,
187               double Alpha1,double Alpha2,
188               double xi, double yi, double &X, double &Y,
189               double Gamma){
190
```

```
191       double hi=0, hb=0, hsobra=0, tc=0;
192       double m1=tan(Alpha1), m2=tan(Alpha2), m12=tan(Beta+Theta);
193
194       Id=0;
195
196       X=AlturaH*(1-m2/m1)/(m12-m2);
197       Y=m12*X;
198
199       t=Y/sin(Beta+Theta)/N;  b=t*tan(Theta);
200       tc=AlturaH*cos(Alpha1-Beta)/sin(Alpha1);
201
202       hi=(i+0.5)*t*tan(Alpha1-Beta);          hb=i*b;
203
204       if(t*(0.5+i)>tc)
205               hsobra=((i+0.5)*t-tc)*(tan(Alpha1-Beta)+tan(Beta-Alpha2));
206
207       h=hi-hb-hsobra;     W=h*t*Gamma;
208
209       x1=xi+t*i*cos(Theta+Beta)/cos(Theta);
210       y1=yi+t*i*sin(Theta+Beta)/cos(Theta);
211   }
212
213   void Block::AgregueP(double Ps, double Pt){
214       if(Ps>Pt)P=Ps;      else P=Pt;
215       if(Ps<0&&Pt<0) Id=0;     //estable
216       if(Ps>0&&Pt<0) Id=1;     //desliza
217       if(Ps<0&&Pt>0) Id=2;     //voltea
218       if(Ps>0&&Pt>0) Id=3;     //desliza y voltea
219   }
220
221   void Block::Dibujese(double Beta){
222       fstream ODibujoDat("Dibujo.gnu", ios::app);
223       double x2,x3,x4,y2,y3,y4;
224       x2=x1+t*cos(Beta);      y2=y1+t*sin(Beta);
225       x3=x2-h*sin(Beta);      y3=y2+h*cos(Beta);
226       x4=x3-t*cos(Beta);      y4=y3-t*sin(Beta);
227       //x1=x4+h*sin(Beta);        y1=y4-h*cos(Beta);  //debe cerrar
228
229       ODibujoDat<<x1<<" "<<y1<<endl;
230       ODibujoDat<<x2<<" "<<y2<<endl;
231       ODibujoDat<<x3<<" "<<y3<<endl;
232       ODibujoDat<<x4<<" "<<y4<<endl;
233       ODibujoDat<<x1<<" "<<y1<<endl;
234       ODibujoDat<<"e"<<endl;
235
236       ODibujoDat.close();
237   }
238
239   //----------------------------------------
240   //----------Clase Interactue--------------
241   //----------------------------------------
242   class Link{
243   private:
244   public:
245       double Pslide(Block *Bloque, int i, double Ps, double Beta, double Phi);
246       double Ptoppling(Block *Bloque,int i,double Pt,int N,double Beta,double Phi);
247   };
248
249   double Link::Pslide(Block *Bloque, int i, double Ps, double Beta, double Phi){
```

```
250       return Ps+Bloque[i].W*cos(Beta)*(tan(Beta)-tan(Phi))/(1-tan(Phi)*tan(Phi));
251   }
252
253   double Link::Ptoppling(Block *Bloque,int i,double Pt,int N,double Beta,double Phi){
254       double m, l;
255       if(i+1>(N+1)/2) {
256           if(i==N-1)m=Bloque[i].h/2;  //bloque superior
257           else m=Bloque[i+1].h+Bloque[i].b;
258           l=Bloque[i].h;
259       }
260       if(i+1<(N+1)/2) {
261           m=Bloque[i].h;
262           if(i==0) l=Bloque[i].h/2;   //bloque inferior
263           else l=Bloque[i-1].h-Bloque[i].b;
264       }
265       if(i+1==(N+1)/2) {
266           m=Bloque[i+1].h+Bloque[i].b;
267           l=Bloque[i-1].h-Bloque[i].b;
268       }
269       return      (0.5*Bloque[i].W*(Bloque[i].h*sin(Beta)-Bloque[i].t*cos(Beta))
270                   +Pt*(m-Bloque[i].t*tan(Phi))    )/l;
271   }
```

A.2. Calculo de la inercia de un bloque rectangular

Se define un sólido rígido $\mathcal{S}$, con dimensiones (L_x, L_y, L_z), donde L_x es el ancho, L_y es la altura y L_z es la profundidad, tal como se muestra en la parte (a) de la figura **A-2**. Cada punto de $\mathcal{S}$ tiene una masa dm y una posición dada por $\vec{r_i}$ referida al marco inercial $\mathcal{A}$. El marco $\mathcal{A}$ se estableció en coordenadas cartesianas con ejes (x, y, z).

El bloque $\mathcal{S}$ se encuentra girando al rededor de un punto arbitrario o y, según de definió en el capítulo 2.1.2, la inercia que genera el bloque respecto al giro en o es

$$I_o = \int_{\mathcal{S}} |\vec{r} - \vec{r_o}|^2 dm \tag{A-1}$$

donde $|\vec{r} - \vec{r_o}|$ es la distancia de cada punto 'i' al punto de giro 'o'. Si dicha distancia se define como r entonces tenemos:

$$r = |\vec{r} - \vec{r_o}| \tag{A-2}$$

y la ecuación A-1 quedaría

$$I_o = \int_{\mathcal{S}} r^2 dm \tag{A-3}$$

como $dm = \rho dV$, donde ρ es la densidad del bloque y dV es el diferencial de volumen $(dV = dxdydz)$, la ecuación A-3 se reescribe como:

$$I_o = \int_{\mathcal{S}} \rho r^2 dV \tag{A-4}$$

Por comodidad en los cálculos es conveniente definir un segundo sistema de referencia que esté situado en el centro de masa del bloque, como se observa en la parte (b) de la figura

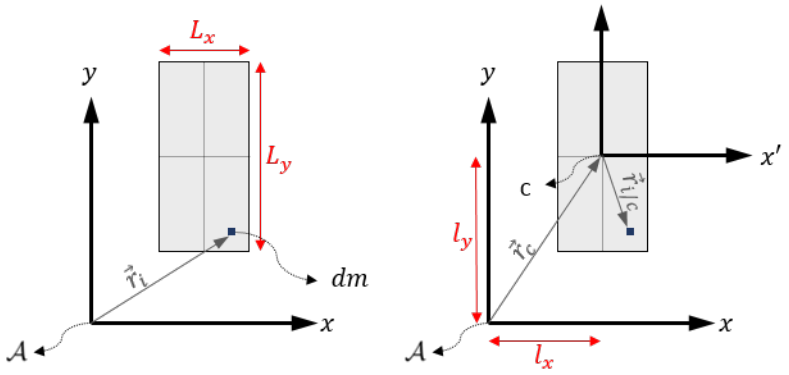


Figura A-2.: Sistemas coordenados para el cálculo de inercia

A-2. Este sistema de referencia se denominará $\mathcal{C}$ y será un sistema cartesiano fijo de ejes (x', y', z'). Las distancia entre los eje x y x' es l_x, mientras la distancia entre los ejes y y y' es l_y. La posición de cualquier punto de $\mathcal{S}$ en el sistema $\mathcal{C}$ es $\vec{r}_{i/c}$, por tanto la posición en el sistema $\mathcal{A}$ sería

$$\vec{r_i} = \vec{r_c} + \vec{r}_{i/c} \tag{A-5}$$

A.2.1. Giro al rededor del centro de masa

Cuando el sólido gira al rededor de su centro de masa, punto 'c', la distancia entre cualquier punto de $\mathcal{S}$ y c se puede calcular por el teorema de Pitágoras así:

$$r^2 = x^2 + y^2 \tag{A-6}$$

donde $x \in [-L_x/2, L_x/2]$ y $y \in [-L_y/2, L_y/2]$. Con x y y referidos al sistema coordenado $\mathcal{C}$.

Remplazando A-6 en A-4 y considerando que el bloque tiene una profundidad $z \in [0, L_z]$, tenemos:

$$I_o = \int_0^{L_z} \int_{-L_y/2}^{L_y/2} \int_{-L_x/2}^{L_x/2} \rho(x^2 + y^2) dx dy dz \tag{A-7}$$

Realizando la primera integral en dx y considerando que ρ es constante en todo $\mathcal{S}$, debido a que el sólido es homogéneo, se tiene:

$$I_o = 2\rho \int_0^{L_z} \int_{-L_y/2}^{L_y/2} (\frac{L_x^3}{24} + \frac{L_x}{2} y^2) dy dz \tag{A-8}$$

Realizando la integral en dy,

$$I_o = \frac{4\rho}{48} \int_0^{L_z} (L_x^3 L_y + L_x L_y^3) dz \tag{A-9}$$

Por último, integrando en dz:

$$I_o = \frac{1}{12} \rho L_z (L_x^3 L_y + L_x L_y^3) \tag{A-10}$$

Como la masa de $\mathcal{S}$ es $M = L_x L_y L_z \rho$ la ecuación A-10 se reescribe como:

$$I_o = \frac{1}{12} M (L_x^2 + L_y^2) \tag{A-11}$$

Esta ecuación permite calcular la inercia del sólido $\mathcal{S}$ rotando sobre su centro de masa. Es evidente que la Inercia es una cantidad constante en el tiempo, ya que depende de la masa y la geometría del bloque. Cada vez que se hable de la inercia respecto al giro en el centro de masa se usará el subíndice cm así:

$$I_{cm} = \frac{1}{12} M (L_x^2 + L_y^2) \tag{A-12}$$

A.2.2. Giro al rededor de un punto arbitrario 'o'

Si el sólido gira alrededor de un punto diferente del centro de masa, por ejemplo el origen del marco de referencia $\mathcal{A}$, la distancia entre cualquier punto de $\mathcal{S}$ y el punto de giro se puede calcular así:

$$r^2 = (x + l_x)^2 + (y + l_y)^2 \tag{A-13}$$

donde x y y están referidos al sistema coordenado $\mathcal{C}$ y son las coordenadas de cualquier punto en $\mathcal{S}$. Remplazando A-13 en A-4 y expresando la integral en el volumen de $\mathcal{S}$, tenemos:

$$I_o = \int_0^{L_z} \int_{-L_y/2}^{L_y/2} \int_{-L_x/2}^{L_x/2} \rho((x + l_x)^2 + (y + l_y)^2) dx dy dz \tag{A-14}$$

Expandiendo el polinomio de la triple integral se tiene:

$$I_o = \int_0^{L_z} \int_{-L_y/2}^{L_y/2} \int_{-L_x/2}^{L_x/2} \rho(x^2 + 2xl_x + l_x^2 + y^2 + 2yl_y + l_y) dx dy dz \tag{A-15}$$

Integrando en dx, luego en dy y realizando los cálculos algebraicos necesarios tenemos:

$$I_o = \frac{4\rho}{48} \int_0^{L_z} (L_x^3 L_y + 12 L_x L_y l_x^2 + L_x L_y^3 + 12 L_x L_y l_y^2) dz \tag{A-16}$$

Por último, integrando en dz y recordando que la masa de $\mathcal{S}$ es $M = L_x L_y L_z \rho$,

$$I_o = \frac{1}{12} M(L_x^2 + L_y^2 + 12 l_x^2 + 12 l_y^2) \tag{A-17}$$

Remplazando la ecuación A-12 en A-17 se obtiene la inercia del sólido $\mathcal{S}$ girando alrededor de 'o' en función de la inercia de $\mathcal{S}$ girando en su centro de masa,

$$I_o = I_{cm} + M(l_x^2 + l_y^2) \tag{A-18}$$

A esta ecuación se le conoce con el teorema de Steiner o teorema de ejes paralelos [ref]. Este teorema dice que la inercia de un sólido rotando alrededor de un punto cualquier 'o' es igual a la inercia del sólido rotando sobre su centro de masa (punto 'c') más el producto entre su masa y la distancia al cuadrado entre los puntos o y c. Remplazando la ecuación A-2 en A-18 se obtiene la expresión general de I_o como se citó en el capítulo 2.1.2.

$$I_o = \frac{1}{12} M(L_x^2 + L_y^2) + M|\vec{r}_{cm} - \vec{r_o}|^2 \tag{A-19}$$

A.3. Cálculo de los factores dimensionales por el Teorema Π de Vaschy-Buckingham

En programa *Toppling_UN* se utilizan 10 dimensiones diferentes. Las variables que representan cada dimensión son las siguientes:

Tabla A-1.: Variables dimensionales en el programa *Toppling_UN*

Variables	símbolo	Unidades elementales	Unidades en el SI
Masa	m	M	(g)
Longitud	l	L	(m)
Tiempo	t	T	(s)
Rigidez resorte	E	$ML^{-1}T^{-2}$	(Pa)
Peso unitario	γ	ML^{-3}	(g/m^3)
Velocidad	v	LT^{-1}	(m/s)
Aceleración	a	LT^{-2}	(m/s^2)
Viscosidad	η	MT^{-1}	(g/s)
Fuerza	F	MLT^{-2}	(N)
Torque	τ	ML^2T^{-2}	$(N*m)$

El Teorema Π de Vaschy-Buckingham nos indica que hay 8 grupos adimensionales que se pueden formar con base en las 3 unidades elementales: masa, longitud y tiempo. En un proceso de *toppling* la masa movilizada y el desplazamiento son medibles, el tiempo no. Por este motivo, envés del tiempo se considera como unidad elemental el módulo de elástico de la roca E. Las unidades de E contienen a la unidad de tiempo. Nótese que el módulo de elasticidad y la rigidez de los resortes tienen las misma unidades (Pa). Esto ocurre porque se asume que los resortes actúan en 1m de profundidad.

La unidades del programa no son las mismas unidades físicas. A las unidades físicas les denominamos *virtuales* con un subíndice v y a las unidades físicas *reales* con un subíndice r. La relación que existe entre las unidades reales y virtuales se denomina χ. Por ejemplo, para la variable de tiempo la relación es:

$$t_v \chi_t = t_r \tag{A-20}$$

El propósito es definir todos los factores χ para las 10 variables. Como punto de partida se definen los χ para la longitud, la masa y el módulo de elasticidad:

$$\chi_l = \frac{l_r}{l_v} = 1 \qquad \chi_m = \frac{m_r}{m_v} = 1 \times 10^6 \qquad \chi_E = \frac{E_r}{E_v} = 1 \times 10^7 \tag{A-21}$$

Los demás χ se obtienen por medio de grupos adimensionales en función de l, m y E con cada una de las variables dependientes. El primer grupo Π_1, para la variable de tiempo, es:

$$\Pi_1 = tl^a m^b E^c \rightarrow [T][L^a][M^b][(ML^{-1}T^{-2})^c] = [L^{a-c}][M^{b+c}][T^{1-2c}] \tag{A-22}$$

con lo cual se crea un sistema de ecuaciones de 3x3:

$$a - c = 0 \qquad b + c = 0 \qquad 1 - 2c = 0 \tag{A-23}$$

y la solución al sistema es:

$$c = 1/2 \qquad b = -1/2 \qquad a = 1/2 \tag{A-24}$$

con lo cual el grupo adimensional Π_1 queda como:

$$\Pi_1 = t\sqrt{\frac{lE}{m}} \tag{A-25}$$

Con base en esta ecuación, es posible definir una función por medio de una constante C, así:

$$t = C_1\sqrt{\frac{m}{lE}} \tag{A-26}$$

De acuerdo a la Ecuación A-20, y considerando un t_v y t_r según la Ecuación A-26, se tiene que:

$$t_v\chi_t = t_r \rightarrow C_1\sqrt{\frac{m_v}{l_v E_v}}v\chi_t = C_1\sqrt{\frac{m_r}{l_r E_r}} \tag{A-27}$$

al despejar χ_t y remplazando los resultado de la Ecuación A-23:

$$\chi_t = \sqrt{\frac{\chi_m}{\chi_l \chi_E}} = \sqrt{\frac{1 \times 10^6}{1 \times 10^7}} = 0{,}3162 \tag{A-28}$$

Los otros grupos adimensionales siguen el mismo procedimiento para hallar χ. En la Tabla se resumen los coeficientes a, b, y c, y se define el χ para cada caso.

Tabla A-2.: Matriz Dimensional

	Π_1(t)	Π_2 (γ)	Π_3(v)	Π_4(a)	Π_5 (η)	Π_6(F)	Π_7 (τ)
a	1/2	3	−3/2	−2	−1/2	−2	−1
b	−1/2	−1	1/2	1	−1/2	0	0
c	1/2	0	−1/2	−1	−1/2	−1	−1
χ	3.16×10^{-1}	1×10^6	3.16	1×10^1	3.16×10^6	1×10^7	1×10^7

B. Anexo 2: Código fuente del programa Toppling_UN

B.1. Código del programa principal

```
1  //Programa: Toppling_UN     Version 1       Fecha 19 de Julio de 2019
2  //Realizado por Juan Pablo Romero Bermudez
3  //Trabajo de Grado: Dinamica de un proceso de volteo de rocas
4  //Universidad Nacional de Colombia - Maestria en Geotecnia
5  //Contacto juanpasjp@gmail.com o juapromerober@unal.edu.co
6  //o en redes @JuanpasJP
7
8  #include <iostream>
9  #include <fstream>
10 #include <cmath>
11 #include "Clases.h"          //Anexo B.2.1
12 #include "Animacion.h"       //Anexo B.2.2
13
14 using namespace std;
15
16 //*************************************************************************
17 //----------Lista de variables y funciones globales----------------------
18 //*************************************************************************
19 double      X=0,        Y=0,            dist=0.002;
20 int         N=11,       Npisos=N+1,     N_DEM=0;
21 double      E=2e6,      ro=2.5,         Phi=40;
22 double      t,          dt=1e-5;
23 double      tdibujo,    tDat;
24
25 bool InicioManual=true;          //True son las coordenadas del ejemplo
26 double Beta=30.0*M_PI/180,       Delta=3.0*M_PI/180;
27 double Alpha1=64.31*M_PI/180,    Alpha2=0.0*M_PI/180;
28 double AlturaH=9.0;
29
30 Block* Bloques;
31
32 void DEM(Link& Fuerza, double tmax, double Ndib, double datCada,
33                                                   double Phi0, bool bool_giro);
34 void InicieManual(void);
35
36
37 //*************************************************************************
38 //----------Programa Principal-------------------------------------------
39 //*************************************************************************
40 int main(void){
41     //-------------------------------------------------------------------
42     system("cd >Dir.inf");       //lee la ubicacion en el computador
43     system("del *.dat");         //borra la anterior ejecucion
44     system("del *.gnu");         //borra la anterior ejecucion
45     system("del *.txt");         //borra la anterior ejecucion
46     //-------------------------------------------------------------------
47     Bloques=new Block[N+Npisos];
48     Link Fuerza(N,Npisos);
49     //-------------------------------------------------------------------
50     if(InicioManual==false)
51         for(int i=0;i<N+Npisos;i++)
52             Bloques[ i].InicioAutomatico(i,N,dist,
53                                          AlturaH, Beta, Delta, Alpha1, Alpha2,
54                                          ro,Phi, E);
55     else InicieManual();
56     //bloque del piso Incia con(x0,y0,z0,Theta0,    t0,h0,ro0,Phi0,E0)
```

```
57      Bloques[N+Npisos-1].Inicie (-4,-0.5-0.05,0.0,-0.001*M_PI/180,
58                                  10,1, ro,Phi,E);
59      //-------------------------------------------------------------------------
60      InicieAnimacion(X,Y);
61      //DEM(Link, tmax, Ndib, datCada, phi, giro)
62      DEM(Fuerza,  3,  30, 1000,  40, false);//PreDEM: un phi mayor a beta en 10
63      DEM(Fuerza, 30, 300, 1000, Phi,  true); //DEM: se debe usar el phi real
64      //-------------------------------------------------------------------------
65      system("Dibujar.cmd");
66      return 0;
67  }
68
69
70  //*************************************************************************
71  //----------Funcion DEM----------------------------------------------------
72  //*************************************************************************
73  void DEM(Link& Fuerza, double tmax, double Ndib, double datCada,
74                                                      double Phi0, bool bool_giro){
75      fstream O_Pos("DAT_Pos.dat", ios::app); //datos de link
76      fstream O_Vel("DAT_Vel.dat", ios::app); //datos de link
77      fstream O_Fue("DAT_Fue.dat", ios::app); //datos de link
78
79      for(int i=0;i<N+Npisos;i++) Bloques[i].CambiePhi(Phi0);
80
81      for(t=0, tdibujo=0, tDat=0; t<tmax; t+=dt, tdibujo+=dt, tDat+=dt){
82
83          //------------------------------
84          //-----Resultados graficos------
85          //------------------------------
86          if(t==0||tdibujo>tmax/Ndib){
87              InicieCuadro(N_DEM,t);
88              for(int i=0;i<N;i++) InicieUnaImagen(i);
89              FinCuadro(Npisos);
90              for(int i=0;i<N+Npisos;i++) Bloques[i].Dibujese();
91              tdibujo=0; //cout<<t<<endl;
92          }
93
94          //------------------------------
95          //---Resultados numericos DAT---
96          //------------------------------
97          if(t==0||tDat>  dt*datCada-dt){
98              for(int i=0;i<N;i++){
99                  O_Pos<<N_DEM<<" "<<t<<" "<<Bloques[i].Getx(4)
100                                   <<" "<<Bloques[i].Gety(4)
101                                   <<" "<<Bloques[i].GetTheta();
102                 O_Vel<<N_DEM<<" "<<t<<" "<<Bloques[i].GetVx()
103                                   <<" "<<Bloques[i].GetVy()
104                                   <<" "<<Bloques[i].GetV()
105                                   <<" "<<Bloques[i].GetOmega();
106                 O_Fue<<N_DEM<<" "<<t<<" "<<Bloques[i].GetFx()
107                                   <<" "<<Bloques[i].GetFy()
108                                   <<" "<<Bloques[i].GetF()
109                                   <<" "<<Bloques[i].Gettau();
110                               }
111             O_Pos<<endl; O_Vel<<endl; O_Fue<<endl;
112             tDat=0;
113         }
114
115         //------------------------------
```

```
116          //-----Programa DEM-------------
117          //------------------------------
118          Fuerza.CalcularFuerzasyTorques(Bloques,dt,bool_giro);
119          for(int i=0;i<N;i++)    Bloques[i].Mueva_V_Omega(dt);
120          for(int i=0;i<N;i++)    Bloques[i].Mueva_r_Theta(dt);
121
122      }
123
124      O_Pos.close();  O_Vel.close();  O_Fue.close();
125      N_DEM++;
126  }
127
128
129  //*************************************************************************
130  //----------Funcion Inicio Manual-------------------------------------------
131  //*************************************************************************
132  void InicieManual(){
133      //Es el ejemplo de talud de referencia (11 bloques)
134      //las coordenadas y medidas fueron tomadas de autocad.
135      double Theta0=30*M_PI/180;
136              //(    x0,       y0,z0,   heta0,      t0,      h0,ro0,Phi0,E0
137  Bloques[ 0].Inicie ( 0.524026, 0.598602,0.0, Theta0, 1.50225,0.512573, ro,Phi,E);
138  Bloques[ 1].Inicie ( 1.550720, 1.828810,0.0, Theta0, 1.50225,1.458990, ro,Phi,E);
139  Bloques[ 2].Inicie ( 2.577410, 3.059010,0.0, Theta0, 1.50225,2.405410, ro,Phi,E);
140  Bloques[ 3].Inicie ( 3.604100, 4.289220,0.0, Theta0, 1.50225,3.351830, ro,Phi,E);
141  Bloques[ 4].Inicie ( 4.630790, 5.519420,0.0, Theta0, 1.50225,4.298240, ro,Phi,E);
142  Bloques[ 5].Inicie ( 5.661550, 6.742580,0.0, Theta0, 1.50225,5.228390, ro,Phi,E);
143  Bloques[ 6].Inicie ( 7.161360, 7.153320,0.0, Theta0, 1.50225,4.282330, ro,Phi,E);
144  Bloques[ 7].Inicie ( 8.661170, 7.564060,0.0, Theta0, 1.50225,3.336280, ro,Phi,E);
145  Bloques[ 8].Inicie (10.161000, 7.974800,0.0, Theta0, 1.50225,2.390230, ro,Phi,E);
146  Bloques[ 9].Inicie (11.660800, 8.385540,0.0, Theta0, 1.50225,1.444180, ro,Phi,E);
147  Bloques[10].Inicie (13.160600, 8.796290,0.0, Theta0, 1.50225,0.498129, ro,Phi,E);
148
149  Bloques[ N+0].Inicie ( 0.903169,-0.058094,0.0, Theta0, 1.50275,1, ro,Phi,E);
150  Bloques[ N+1].Inicie ( 2.166460, 0.762300,0.0, Theta0, 1.50275,1, ro,Phi,E);
151  Bloques[ N+2].Inicie ( 3.429760, 1.582690,0.0, Theta0, 1.50275,1, ro,Phi,E);
152  Bloques[ N+3].Inicie ( 4.693060, 2.403090,0.0, Theta0, 1.50275,1, ro,Phi,E);
153  Bloques[ N+4].Inicie ( 5.956350, 3.223480,0.0, Theta0, 1.50275,1, ro,Phi,E);
154  Bloques[ N+5].Inicie ( 7.219650, 4.043880,0.0, Theta0, 1.50275,1, ro,Phi,E);
155  Bloques[ N+6].Inicie ( 8.482940, 4.864270,0.0, Theta0, 1.50275,1, ro,Phi,E);
156  Bloques[ N+7].Inicie ( 9.746240, 5.684660,0.0, Theta0, 1.50275,1, ro,Phi,E);
157  Bloques[ N+8].Inicie (11.009500, 6.505060,0.0, Theta0, 1.50275,1, ro,Phi,E);
158  Bloques[ N+9].Inicie (12.272800, 7.325450,0.0, Theta0, 1.50275,1, ro,Phi,E);
159  Bloques[N+10].Inicie (13.536100, 8.145850,0.0, Theta0, 1.50275,1, ro,Phi,E);
160      }
```

B.2. Código de las librerías

B.2.1. Clases.h

```
1  //Libreria "Clases.h"       Version 1       Fecha 19 de Julio de 2019
2  //Forma parte del programa: Toppling_UN
3  //Realizado por Juan Pablo Romero Bermudez
4  //Trabajo de Grado: Dinamica de un proceso de volteo de rocas
5  //Universidad Nacional de Colombia - Maestria en Geotecnia
6  //Contacto juanpasjp@gmail.com o juapromerober@unal.edu.co
7  //o en redes @JuanpasJP
8
9  #include <iostream>
10 #include <cmath>
11 #include <fstream>
12 #include "Vector.h"     //Anexo B.2.3
13
14 using namespace std;
15
16 const double g=1;
17 const double Eta=1e4;
18 const double ERR=1e-12;
19 /*La variable ERR se definie para aliviar el calculo numerico en las funciones
20 trigonometricas. Por ejemplo, la tangente de pi/2 no esta definicada. Sin embargo,
21 la tangente de (pi/2-ERR) si existe.*/
22
23
24 //*************************************************************************
25 //----------------------01 Clase Block-----------------------------------
26 //*************************************************************************
27 class Block{
28 private:
29     double t,h,m,Phi,E;                 //Ancho, alto, masa, phi, modulo Elastico
30     vector3D r[5],rold[5],V[5],F;       //variables lineales
31     Line edge[4];                       //bordes
32     double I,Theta,Omega,Tau;           //Variables angulares
33 public:
34     void Inicie(double x0, double y0, double z0, double Theta0,
35                 double t0, double h0, double ro0, double Phi0, double E0);
36     void InicioAutomatico(  int i, int N0, double dist0,
37                     double AlturaH0, double Beta0, double delta0,
38                     double Alpha10,double Alpha20,
39                     double ro0, double Phi0, double E0);
40     void CambiePhi(double Phi0){Phi=Phi0*M_PI/180;};
41     friend class Link;
42     //----Funciones Principales
43     void BorreFuerzayTorque(void);
44     void AgregueFuerza(vector3D F0);
45     void AgregueTorque(double tau0);
46     void Mueva_V_Omega(double dt);
47     void Mueva_r_Theta(double dt);
48     void UbiqueVertices(double x0, double y0, double z0);
49     void Dibujese(void);
50     double Inercia(int PosGiro);
51     //----Funciones Inline
52     double Getx(int Vertex){return r[Vertex].x();};
53     double Gety(int Vertex){return r[Vertex].y();};
54     double GetV(void){return norma(V[4]);};
```

```
55      double GetVx(void){return V[4].x();};
56      double GetVy(void){return V[4].y();};
57      double GetF(void){return norma(F);};
58      double GetFx(void){return F.x();};
59      double GetFy(void){return F.y();};
60      double GetTheta(void){return Theta;};
61      double GetOmega(void){return Omega;};
62      double Gettau(void){return Tau;};
63      double GetE(void){return E;};
64      double Geth(void){return h;};
65      double Gett(void){return t;};
66      double Getm(void){return m;};
67      double GetI(void){return I;};
68  };
69
70
71  //*************************************************************************
72  //----------------------02 Clase Link-------------------------------------
73  //*************************************************************************
74  class Link{
75  public:
76      int N, Npared;
77      bool *M_Past;
78      bool *M_PastOld;
79      bool *M_PastNew;
80      int *M_edge;
81      double *M_t;
82      double *M_dn_old;
83  public:
84      Link(int NumBloques, int NumPared);
85      void CalcularFuerzasyTorques(Block* Bloque, double dt, bool DEM);
86      void CalcularCollision(Block &Bq1,Block &Bq2,
87                             int Numbloq1, int Numbloq2,
88                             double dt, int giro,bool DEM);
89      bool BuscarContactos(vector3D V_i, Block &Bq2);
90      bool BuscarCruce(Block &Bq1, Line &L2, int v, int Id,bool Linea_Linea);
91      double HalleDist(Line ej, vector3D P);
92      vector3D CalcularFuerzaColision(vector3D &Tau1, vector3D &Tau2,
93                                      vector3D &vec_Ft0,  Block &Bq1, Block &Bq2,
94                                      int Id, int vi, int giro, bool DEM);
95  };
96
97
98  //*************************************************************************
99  //********01 Metodos de la Clase Block*************************************
100 //********01 Metodos de la Clase Block*************************************
101 //*************************************************************************
102 void Block::Inicie( double x0, double y0, double z0, double Theta0,
103                     double t0, double h0, double ro0, double Phi0,double E0){
104
105     double Vx0=0, Vy0=0, Vz0=0, Omega0=0;
106     r[4].cargue(x0,y0,z0);
107     for (int i=0;i<5;i++) V[i].cargue(Vx0,Vy0,Vz0);
108     t=t0; h=h0; m=ro0*t*h; Phi=Phi0*M_PI/180; E=E0;
109     Omega=Omega0; Theta=Theta0;
110     UbiqueVertices(x0,y0,z0);
111     I=Inercia(4);
112
113 }
```

```
114 //--------------------------------------------------------------------------
115 void Block::InicioAutomatico(   int i, int N0, double dist0,
116                     double AlturaH0, double Beta0, double delta0,
117                     double Alpha10,double Alpha20,
118                     double ro0, double Phi0,double E0){
119
120     double Vx0=0, Vy0=0, Vz0=0, Omega0=0;
121     double C=0.0, hi=0, hb=0, hsobra=0, tc=0, b=0, xc=0, yc=0;
122     double m1=tan(Alpha10), m2=tan(Alpha20), m12=tan(Beta0+delta0);
123     double Y, X; bool piso=false;
124
125     X=AlturaH0*(1-m2/m1)/(m12-m2);
126     Y=m12*X;
127     t=Y/sin(Beta0+delta0)/N0;   b=t*tan(delta0);
128     tc=AlturaH0*cos(Alpha10-Beta0)/sin(Alpha10);
129     if(i<N0){
130         hi=(i+0.5)*t*tan(Alpha10-Beta0);            hb=i*b;
131         if(t*(0.5+i)>tc)
132                 hsobra=((i+0.5)*t-tc)*(tan(Alpha10-Beta0)+tan(Beta0-Alpha20));
133         h=hi-hb-hsobra;
134     }else{  h=1; i=i-N0; piso=true;}
135
136     xc=C*cos(Beta0);                          yc=C*sin(Beta0);
137     xc+=t*i*cos(delta0+Beta0)/cos(delta0);  yc+=t*i*sin(delta0+Beta0)/cos(delta0);
138     xc+=t/2*cos(Beta0)-h/2*sin(Beta0);       yc+=t/2*sin(Beta0)+h/2*cos(Beta0);
139     xc+=(i+1)*dist0*cos(delta0+Beta0);       yc+=(i+1)*dist0*sin(delta0+Beta0);
140
141     if(piso){xc+=(dist0+h)*sin(Beta0);       yc-=(dist0+h)*cos(Beta0);}
142     if(piso)t=t+0.0005; //el piso no debe estar tan separado, por eso el 0.5mm
143     r[4].cargue(xc,yc,0);
144
145     for (int i=0;i<5;i++) V[i].cargue(Vx0,Vy0,Vz0);
146     m=ro0*t*h; Phi=Phi0*M_PI/180; E=E0;
147     Omega=Omega0; Theta=Beta0;
148     UbiqueVertices(xc,yc,0);
149     I=Inercia(4);
150     }
151 //--------------------------------------------------------------------------
152 double Block::Inercia(int PosGiro){
153     double lx, ly;
154     lx=r[4].x()-r[PosGiro].x();
155     ly=r[4].y()-r[PosGiro].y();
156     return 1.0/12.0*m*(t*t+h*h)+m*(lx*lx+ly*ly);                }
157 //--------------------------------------------------------------------------
158 void Block::BorreFuerzayTorque(void){F.cargue(0,0,0);   Tau=0;}
159 //--------------------------------------------------------------------------
160 void Block::AgregueFuerza(vector3D F0){F+=F0;}
161 //--------------------------------------------------------------------------
162 void Block::AgregueTorque(double tau0){Tau+=tau0;}
163 //--------------------------------------------------------------------------
164 void Block::Mueva_V_Omega(double dt){V[4]+=(F/m)*(dt);Omega+=Tau*(dt/I);}
165 //--------------------------------------------------------------------------
166 void Block::Mueva_r_Theta(double dt){
167     for(int i=0;i<5;i++) rold[i]=r[i];      //guarde anterior
168     r[4] +=V[4]*dt+0.5*( F/m )*dt*dt;
169     Theta+=Omega*dt+0.5*(Tau/I)*dt*dt;
170     UbiqueVertices(r[4].x(),r[4].y(),r[4].z());             }
171 //--------------------------------------------------------------------------
172 void Block::UbiqueVertices(double xc, double yc, double zc){
```

```
173       double x0,x1,x2,x3,y0,y1,y2,y3;
174       x0=xc-t/2*cos(Theta)+h/2*sin(Theta);
175       x1=xc+t/2*cos(Theta)+h/2*sin(Theta);
176       x2=xc+t/2*cos(Theta)-h/2*sin(Theta);
177       x3=xc-t/2*cos(Theta)-h/2*sin(Theta);
178       y0=yc-t/2*sin(Theta)-h/2*cos(Theta);
179       y1=yc+t/2*sin(Theta)-h/2*cos(Theta);
180       y2=yc+t/2*sin(Theta)+h/2*cos(Theta);
181       y3=yc-t/2*sin(Theta)+h/2*cos(Theta);
182
183       r[0].cargue(x0,y0,zc);       //V[1].cargue(0,0,0);
184       r[1].cargue(x1,y1,zc);       //V[2].cargue(0,0,0);
185       r[2].cargue(x2,y2,zc);       //V[3].cargue(0,0,0);
186       r[3].cargue(x3,y3,zc);       //V[4].cargue(0,0,0);
187
188       edge[0].InicieLine(x0,y0,x1-x0,y1-y0);
189       edge[1].InicieLine(x1,y1,x2-x1,y2-y1);
190       edge[2].InicieLine(x2,y2,x3-x2,y3-y2);
191       edge[3].InicieLine(x3,y3,x0-x3,y0-y3);       }
192   //-------------------------------------------------------------------------------
193   void Block::Dibujese(void){
194       fstream ODibujoDat("Dibujo.gnu", ios::app);
195       ODibujoDat<<r[0].x()<<" "<<r[0].y()<<endl;
196       ODibujoDat<<r[1].x()<<" "<<r[1].y()<<endl;
197       ODibujoDat<<r[2].x()<<" "<<r[2].y()<<endl;
198       ODibujoDat<<r[3].x()<<" "<<r[3].y()<<endl;
199       ODibujoDat<<r[0].x()<<" "<<r[0].y()<<endl;
200       //ODibujoDat<<r[4].x()<<" "<<r[4].y()<<endl; //esto crea una linea al CM
201       ODibujoDat<<"e"<<endl;
202       ODibujoDat.close();                  }
203
204
205   //*************************************************************************
206   //********02 Metodos de la Clase Link*************************************
207   //********02 Metodos de la Clase Link*************************************
208   //*************************************************************************
209   Link::Link(int NumBloques, int NumPared){
210       N=NumBloques; Npared=NumPared;
211       int Size=(N+Npared)*(N+Npared)*4;
212       M_Past=new bool [Size];          //En Uso
213       M_PastOld=new bool [Size];       //En Uso
214       M_PastNew=new bool [Size];       //En Uso
215       M_edge=new int [Size];           //En Uso
216       M_t=new double [Size];           //En Uso
217       M_dn_old=new double [Size];      //En Uso
218       for (int i=0; i<Size;i++) { M_Past[i]=false;
219                                   M_PastOld[i]=false;
220                                   M_PastNew[i]=false;
221                                   M_edge[i]=-1;
222                                   M_t[i]=-1.0;
223                                   M_dn_old[i]=-1.0;
224   }   }
225   //-------------------------------------------------------------------------------
226   void Link::CalcularFuerzasyTorques(Block* Bloque, double dt, bool DEM){
227       vector3D vec_g; vec_g.cargue(0,-g,0);
228       vector3D vec_sismo; vec_sismo.cargue(-0.0*g,0,0);
229       vector3D vec_dir_W, Torque;
230       int giro=4, Id=0;
231
```

```
232     //Borrar todas las fuerzas y torques
233     for (int i=0;i<N+Npared;i++){   Bloque[i].BorreFuerzayTorque(); }
234
235     //Buscar Nuevos Contactos
236     for(int i=0;i<N+Npared;i++)         //como bloques actuantes
237         for(int j=0;j<N+Npared;j++)     //como bloques pasivos
238             for(int vi=0;vi<4;vi++){
239                 Id=(4*(N+Npared))*j+4*i+vi;
240                 M_PastNew[Id]=BuscarContactos(Bloque[i].r[vi],Bloque[j]);
241                 M_PastOld[Id]=M_Past[Id];
242                 }
243     //interaccion con otros bloques (fuerza interna AA y fuerza externa PA)
244     for(int i=0;i<N+Npared;i++)     //como bloques actuantes
245         for(int j=0;j<N;j++){       //como bloques pasivos
246                 if(i!=j) CalcularCollision(Bloque[i],Bloque[j],i,j,dt,4,DEM);
247                     }
248     //interaccion con la frontera (fuerza externa AP)
249     for(int i=0;i<N;i++)                //como bloques actuantes
250         for(int j=N;j<N+Npared;j++){    //como bloques pasivos
251             if(i>=N && j>=N);
252             else
253                 if(i!=j) CalcularCollision(Bloque[i],Bloque[j],i,j,dt,4,DEM);
254                     }
255     //agreguegravedad
256     for(int i=0;i<N;i++)    Bloque[i].AgregueFuerza(Bloque[i].m*vec_g);
257     //agreguesismo
258     for(int i=0;i<N;i++)
259         Bloque[i].AgregueFuerza(Bloque[i].m*vec_sismo);
260 }
261 //---------------------------------------------------------------------------
262 void Link::CalcularCollision(Block &Bq1,Block &Bq2,
263                              int Numbloq1, int Numbloq2,
264                                                  double dt, int giro, bool DEM){
265
266     bool Collision, Cruce,Linea_Linea=true;
267     int Id, Id_A, Id_B, Nchoques=0, Ncruce=0, IdCruce=0, IdCruceAUX=0;
268     double DistCruce=0, DistCruceMin=0,DistDesliza=0;
269     vector3D vec_Fn; vec_Fn.cargue(0,0,0);
270     vector3D vec_Ft; vec_Ft.cargue(0,0,0);
271     vector3D Torque1; Torque1.cargue(0,0,0);
272     vector3D Torque2; Torque2.cargue(0,0,0);
273     for(int vi=0;vi<4;vi++){ Ncruce=0; DistDesliza=0;
274         Id=(4*(N+Npared))*Numbloq2+4*Numbloq1+vi;
275         Collision=BuscarContactos(Bq1.r[vi],Bq2);
276         if(Collision){
277             if(M_Past[Id]){         }
278             else{
279                 for(int ej=0; ej<4; ej++){
280                     Cruce=BuscarCruce(Bq1,Bq2.edge[ej],vi,Id,Linea_Linea);
281                     if(Cruce){  Ncruce++;
282                         DistCruce=HalleDist(Bq2.edge[ej],Bq1.r[vi]);
283                         //La primera vez se debe llenar con algun valor
284                         if(Ncruce==1) {DistCruceMin=DistCruce;IdCruce=ej;}
285                         if(DistCruce<DistCruceMin){
286                             DistCruceMin=DistCruce;
287                             IdCruce=ej;
288                         }
289                     }
290                 }
```

```
291                 if(IdCruce==3)IdCruceAUX=0; else IdCruceAUX=IdCruce+1;
292                 Id_A=(4*(N+Npared))*Numbloq1+4*Numbloq2+IdCruce;
293                 Id_B=(4*(N+Npared))*Numbloq1+4*Numbloq2+IdCruceAUX;
294
295                 if(M_PastOld[Id_A] && !M_PastNew[Id_A])
296                     if(norma(Bq1.r[vi]-Bq2.r[IdCruce])<0.01){Linea_Linea=false;
297                         IdCruce=IdCruce-1; if(IdCruce==-1) IdCruce=3;
298                         BuscarCruce(Bq1,Bq2.edge[IdCruce],vi,Id,false); }
299
300                 if(M_PastOld[Id_B] && !M_PastNew[Id_B])
301                     if(norma(Bq1.r[vi]-Bq2.r[IdCruceAUX])<0.01){Linea_Linea=false;
302                         IdCruce=IdCruce+1; if(IdCruce==4) IdCruce=0;
303                         BuscarCruce(Bq1,Bq2.edge[IdCruce],vi,Id,false); }
304
305                 if(Linea_Linea){
306                     BuscarCruce(Bq1,Bq2.edge[IdCruce],vi,Id,Linea_Linea);   }
307
308                 M_edge[Id]=IdCruce;
309                 M_t[Id]=Bq2.edge[IdCruce].t;
310                 M_Past[Id]=true;
311                 }
312
313         Nchoques++;
314         vec_Fn+=CalcularFuerzaColision(Torque1,Torque2,vec_Ft,
315                                                         Bq1,Bq2,Id,vi,giro,DEM);
316         }
317         else{
318             if(M_Past[Id]){ M_edge[Id]=-1;
319                             M_t[Id]=-1;
320                             M_Past[Id]=false;
321                             M_dn_old[Id]=-1;
322                             }
323         }
324     }
325
326     Bq1.AgregueFuerza(vec_Fn+vec_Ft);           Bq1.AgregueTorque(Torque1.z());
327     Bq2.AgregueFuerza((-1)*(vec_Fn+vec_Ft));    Bq2.AgregueTorque(Torque2.z());
328 }
329 //-------------------------------------------------------------------------------
330 bool Link::BuscarContactos(vector3D V_i, Block &Bq2){
331     vector3D A,B,C,D, Z;
332     A=Bq2.r[0]-V_i;
333     B=Bq2.r[1]-V_i;
334     C=Bq2.r[2]-V_i;
335     D=Bq2.r[3]-V_i;
336     //true es que hay contacto; false que no lo hay
337     Z=A^B;      if(Z.z()<0) return false;
338     Z=B^C;      if(Z.z()<0) return false;
339     Z=C^D;      if(Z.z()<0) return false;
340     Z=D^A;      if(Z.z()<0) return false;
341     return true;
342 }
343 //-------------------------------------------------------------------------------
344 bool Link::BuscarCruce(Block &Bq1, Line &L2,int v, int Id, bool Linea_Linea){
345     double m=0; Line L1;
346     if(Linea_Linea){
347         L1.InicieLine(  Bq1.r[v].x(),   Bq1.r[v].y(),
348                         Bq1.rold[v].x()-Bq1.r[v].x(),
349                         Bq1.rold[v].y()-Bq1.r[v].y()    );}
```

```
350       else{
351           L1.InicieLine(  Bq1.r[v].x(),   Bq1.r[v].y(),
352                           1,
353                           1*(-L2.a/L2.b));}
354
355       if(L1.a==0&&L2.a==0) goto LineasParalelas;
356       if(L1.a==0){
357           m=(L2.a*L1.b-L1.a*L2.b)/L2.a;
358           if(m==0) goto LineasParalelas;
359           L1.t=1/m*(L2.b/L2.a*(L1.x-L2.x)-(L1.y-L2.y));
360           L2.t=(L1.x+L1.a*L1.t-L2.x)/L2.a;
361       }
362       else{
363           m=(L1.a*L2.b-L2.a*L1.b)/L1.a;
364           if(m==0) goto LineasParalelas;
365           L2.t=1/m*(L1.b/L1.a*(L2.x-L1.x)-(L2.y-L1.y));
366           L1.t=(L2.x+L2.a*L2.t-L1.x)/L1.a;
367       }if(0<=L2.t&&L2.t<=1) return true;
368       LineasParalelas:
369
370       return false;
371   }
372   //---------------------------------------------------------------------------
373   double Link::HalleDist(Line ej, vector3D P){
374       vector3D Q, PQ, Vec_ej;
375       Q.cargue(ej.x,ej.y,0);      PQ=P-Q;     Vec_ej.cargue(ej.a,ej.b,0);
376       return norma(Vec_ej^PQ)/norma(Vec_ej);
377   }
378   //---------------------------------------------------------------------------
379   vector3D Link::CalcularFuerzaColision(  vector3D &Tau1, vector3D &Tau2,
380                                           vector3D &vec_Ft0,
381                                           Block &Bq1, Block &Bq2,
382                                           int Id, int vi, int giro, bool DEM){
383       vector3D vec_z;     vec_z.cargue(0,0,1);
384       vector3D vec_Fn, vec_Ft, vec_F; vec_Fn.cargue(0,0,0);
385       vector3D vec_dir;               //para torque
386
387       double xc,yc,xv,yv;             //Coordenadas de entrada
388       vector3D vec_e, vec_d;          //Vectores
389       vector3D vec_dn, vec_dt;        //Componente del vector d
390       double alpha, e, d, dn, dt;     //Magnitudes
391
392       vector3D vec_vel,vec_velnormal; //Vector velocidad y componentes
393       double eta, vel, vel_n;         //Magnitudes
394
395       vector3D vec_ayuda;
396       double chi;             //sentido de las fuerzas (amortiguada - friccion)
397
398       vector3D vec_peso;  vec_peso.cargue(0,-1,0);
399       double Beta, Wt, Ftmax1, Ftmax2, Ft;
400
401       double Arg=0,phi=0;             //Argumento de las funciones trigonometricas
402
403       //proceso para vectores normal y tangente
404       xc=Bq2.edge[M_edge[Id]].x+M_t[Id]*Bq2.edge[M_edge[Id]].a;
405       yc=Bq2.edge[M_edge[Id]].y+M_t[Id]*Bq2.edge[M_edge[Id]].b;
406       xv=Bq1.r[vi].x();
407       yv=Bq1.r[vi].y();
408
```

```
409       vec_e.cargue(Bq2.edge[M_edge[Id]].a,Bq2.edge[M_edge[Id]].b,0);
410       e=norma(vec_e);      vec_e=vec_e/e;       //Vec_e unitario
411       vec_d.cargue(xc-xv,yc-yv,0);
412       d=norma(vec_d);      vec_d=vec_d/d;       //Vec_d unitario
413
414       Arg=(vec_e*vec_d)/1/1;
415       if(Arg >= ( 1-ERR)) alpha=acos( 1);                    //radianes
416       if(Arg <= (-1+ERR)) alpha=acos(-1);                    //radianes
417       if(Arg <  (1-ERR) && Arg > (-1+ERR)) alpha=acos(Arg);   //radianes
418
419       if(alpha>M_PI/2){alpha-=M_PI/2; dn=d*cos(alpha); dt=d*sin(alpha); }
420       else{                           dn=d*sin(alpha); dt=d*cos(alpha); }
421
422       vec_dn=vec_e^vec_z;          //Vec_dn Unitario
423       vec_dt=d*vec_d-dn*vec_dn;
424       //para evitar division en cero, caso choche perpendicular
425       if(norma(vec_dt)==0) {vec_dt=vec_dt; cout<<"PASO1"<<endl;}
426       else vec_dt=vec_dt/norma(vec_dt);
427
428       //Fuerza normal elastica
429       vec_Fn=dn*vec_dn*Bq2.E;
430
431       //Fuerza normal amortiguada
432       vec_vel=Bq1.V[4]-Bq2.V[4];
433       vel=norma(vec_vel);
434       //para evitar division en cero, caso inicio adentro
435       if(vel==0){vec_vel=vec_vel;cout<<"PASO2 "<<Id<<endl;}
436       else vec_vel=vec_vel/norma(vec_vel);        //vec_vel unitario
437
438       Arg=(vec_e*vec_vel)/1/1;
439       if(Arg >= ( 1-ERR)) eta=acos( 1);                    //radianes
440       if(Arg <= (-1+ERR)) eta=acos(-1);                    //radianes
441       if(Arg <  (1-ERR) && Arg > (-1+ERR)) eta=acos(Arg); //radianes
442
443       if(eta>M_PI/2){eta-=M_PI/2; vel_n=vel*cos(eta);  }
444       else{                       vel_n=vel*sin(eta);  }
445
446       vec_ayuda=vec_e^vec_vel;
447       if(vec_ayuda.z()>0) vec_velnormal=(1)*vec_dn;
448       else vec_velnormal=(-1)*vec_dn;
449
450       vec_Fn+=Eta*vel_n*vec_velnormal;
451
452       //torque por Fuerza normal
453       if(DEM==true){
454       vec_dir=Bq1.r[4]-Bq1.r[vi];
455       Tau1+=vec_Fn^vec_dir;
456
457       vec_dir=Bq2.r[4]-Bq1.r[vi];
458       Tau2+=((-1)*vec_Fn)^vec_dir;
459       }
460
461       //Fuerza tangente
462       Arg=(vec_vel*vec_d)/1/1;
463       if(Arg >= ( 1-ERR)) chi=acos( 1);                    //radianes
464       if(Arg <= (-1+ERR)) chi=acos(-1);                    //radianes
465       if(Arg <  (1-ERR) && Arg > (-1+ERR)) chi=acos(Arg); //radianes
466
467       if(chi>M_PI/2){ ( 1)*vec_dt;}
```

```
468       else{           (-1)*vec_dt; }
469
470       vec_Ft=dt*vec_dt*Bq2.E;     //FTangente es por phi del bq2
471
472       Ftmax1=norma(vec_Fn)*tan(Bq1.Phi);
473       if(Ftmax1<norma(vec_Ft)) vec_Ft=Ftmax1*vec_dt;
474       vec_Ft0+=vec_Ft;
475
476       //torque por Fuerza tangente
477       if(DEM==true){
478       vec_dir=Bq1.r[giro]-Bq1.r[vi];
479       Tau1+=vec_Ft^vec_dir;
480
481       vec_dir=Bq2.r[giro]-Bq1.r[vi];
482       Tau2+=((-1)*vec_Ft)^vec_dir;
483       }
484
485       M_dn_old[Id]=dn;            //se debe hacer al final
486       return vec_Fn;
487   }
```

B.2.2. Animacion.h

```
1   //Libreria "Animacion.h"        Version 1       Fecha 19 de Julio de 2019
2   //Forma parte del programa: Toppling_UN
3   //Realizado por Juan Pablo Romero Bermudez
4   //Trabajo de Grado: Dinamica de un proceso de volteo de rocas
5   //Universidad Nacional de Colombia - Maestria en Geotecnia
6   //Contacto juanpasjp@gmail.com o juapromerober@unal.edu.co
7   //o en redes @JuanpasJP
8
9   #include <iostream>
10  #include <cmath>
11  #include <fstream>
12  using namespace std;
13
14
15  //*************************************************************************
16  //-------------------------Funcion 1-----------------------------------------
17  //*************************************************************************
18  void InicieAnimacion(double &X, double &Y){
19      fstream ODibujoDat("Dibujo.gnu", ios::app);
20      ifstream IUbicacion("Dir.inf");
21      string pwd;
22      getline(IUbicacion,pwd);
23      ODibujoDat<<"cd '"<<pwd<<"'"<<endl;
24      ODibujoDat<<"set xrange [-10:15]"<<endl;
25      ODibujoDat<<"set yrange [-1:10]"<<endl;
26      ODibujoDat<<"set grid xtics"<<endl;
27      ODibujoDat<<"set grid ytics"<<endl;
28      ODibujoDat<<"set size ratio -1"<<endl;
29      ODibujoDat<<"set terminal gif animate"<<endl;
30      ODibujoDat<<"set output 'Animacion.gif'"<<endl;
31      ODibujoDat.close();
32      IUbicacion.close();
33  }
```

```
34
35
36  //*************************************************************************
37  //-------------------------Funcion 2-------------------------------------------
38  //*************************************************************************
39  void InicieCuadro(int DEM, double t){
40      fstream ODibujoDat("Dibujo.gnu", ios::app);
41      ODibujoDat<<"set title 'DEM "<<DEM<<" t="<<t<<"'"<<endl;
42      ODibujoDat<<"plot ";
43      ODibujoDat.close();
44  }
45
46  void InicieUnaImagen(int i){
47      fstream ODibujoDat("Dibujo.gnu", ios::app);
48      ODibujoDat<<" '-' w l lw 2 notitle 'Bloque "<<i<<"',";
49      ODibujoDat.close();
50  }
51
52  void FinCuadro(int Npared){
53      fstream ODibujoDat("Dibujo.gnu", ios::app);
54      for(int  i=0;i<Npared;i++)
55      ODibujoDat<<" '-' w l lw 2 lc 'black' notitle 'Piso',";
56      ODibujoDat<<endl;
57      ODibujoDat.close();
58  }
59
60  //*************************************************************************
61  //-------------------------Funcion 3-------------------------------------------
62  //*************************************************************************
63  void DibujeTalud(double x1, double y1, double x2, double y2){
64      fstream ODibujoDat("Dibujo.gnu", ios::app);
65      /*ODibujoDat<<"plot";
66      ODibujoDat<<" '-' w l lw 2 lc 'black' notitle ,";*/
67      ODibujoDat<<x1<<" "<<y1<<endl;
68      ODibujoDat<<x2<<" "<<y2<<endl;
69      ODibujoDat<<"e"<<endl;
70      ODibujoDat <<endl;
71      ODibujoDat.close();
72
73  }
```

B.2.3. Vector.h

```
1   //Libreria "Vector.h"       Version 1       Fecha 19 de Julio de 2019
2   //Forma parte del programa: Toppling_UN
3   //Tomado del curso de Metodos de simulacion fisica. Codigo: 2020174-1. 2017-II
4   //Trabajo de Grado: Dinamica de un proceso de volteo de rocas
5   //Realizado por Juan Pablo Romero Bermudez
6   //Universidad Nacional de Colombia - Maestria en Geotecnia
7   //Contacto juanpasjp@gmail.com o juapromerober@unal.edu.co
8   //o en redes @JuanpasJP
9
10  #include <iostream>
11  #include <cmath>
12  using namespace std;
13  //*************************************************************************
```

```
14  //-----------------------------01 Clase vector3D---------------------------------
15  //*****************************************************************************
16  class vector3D{
17    double v[3];
18   public:
19    void   cargue(double x0, double y0, double z0);
20    void   show(void);
21    // Funciones de salida de componentes
22    double x(void){return v[0];};
23    double y(void){return v[1];};
24    double z(void){return v[2];};
25    //Lectura de Elementos
26    double & operator[](int i){return v[i];};
27
28    // Operaciones vectoriales
29    vector3D    operator= (vector3D v2);
30    vector3D    operator+ (vector3D v2);
31    vector3D    operator+=(vector3D v2);
32    vector3D    operator- (vector3D v2);
33    vector3D    operator-=(vector3D v2);
34    // Producto por escalar
35    vector3D    operator* (double a);
36    vector3D    operator*=(double a);
37    friend  vector3D    operator* (double a,vector3D v1);
38    // Division por escalar
39    vector3D    operator/ (double a);
40    // Producto cruz
41    vector3D    operator^ (vector3D v2);
42    // Producto punto
43    double operator* (vector3D v2);
44    // Norma
45    friend  double norma2(vector3D v1);
46    friend  double norma(vector3D v1);
47  };
48  // Metodos de la clase vector3D
49  void vector3D::cargue(double x0, double y0, double z0){
50    v[0]=x0; v[1]=y0; v[2]=z0;
51  }
52  void vector3D::show(void){
53    cout << "(" <<v[0]<< "," <<v[1]<< "," <<v[2]<< ")" << "    ";
54  }
55  vector3D vector3D::operator=(vector3D v2){
56    for(int i=0;i<3;i++)
57      v[i] = v2.v[i];
58    return *this;
59  }
60  vector3D vector3D::operator+(vector3D v2){
61    vector3D total;
62    for(int i=0;i<3;i++)
63      total.v[i] = v[i] + v2.v[i];
64    return total;
65  }
66  vector3D vector3D::operator+=(vector3D v2){
67    *this = *this + v2;
68    return *this;
69  }
70  vector3D vector3D::operator*(double a){
71    vector3D total;
72    for(int i=0;i<3;i++)
```

```
73      total.v[i] = a*v[i];
74    return total;
75  }
76  vector3D vector3D::operator*=(double a){
77    *this = (*this)*a;
78    return *this;
79  }
80  vector3D vector3D::operator/(double a){
81    double inver = 1.0/a;
82    vector3D total;
83    for(int i=0;i<3;i++)
84      total.v[i] = inver*v[i];
85    return total;
86  }
87  vector3D vector3D::operator-(vector3D v2){
88    return *this + v2*(-1);
89  }
90  vector3D vector3D::operator-=(vector3D v2){
91    *this = *this - v2;
92    return *this;
93  }
94  double vector3D::operator*(vector3D v2){
95    double p=0;
96    for(int i=0;i<3;i++)
97      p += v[i]*v2.v[i];
98    return p;
99  }
100 vector3D vector3D::operator^(vector3D v2){
101   vector3D c;
102   c.v[0] = v[1]*v2.v[2]-v[2]*v2.v[1];
103   c.v[1] = v[2]*v2.v[0]-v[0]*v2.v[2];
104   c.v[2] = v[0]*v2.v[1]-v[1]*v2.v[0];
105   return c;
106 }
107 vector3D operator*(double a,vector3D v1){
108   vector3D total;
109   total = v1*a;
110   return total;
111 }
112 double norma2(vector3D v1){
113   double n=0;
114   for(int i=0;i<3;i++)
115     n += v1.v[i]*v1.v[i];
116   return n;
117 }
118 double norma(vector3D v1){
119   return sqrt(norma2(v1));
120 }
121 //*************************************************************************
122 //----------------------------02 Clase Line-----------------------------------
123 //*************************************************************************
124 class Line{
125 public:
126     double x,y,a,b,t;
127 public:
128     InicieLine() {x=0;y=0;a=0;b=0;t=0;};//constructor
129     void InicieLine(double x0,double y0,double a0,double b0);};
130 void Line::InicieLine(double x0,double y0,double a0,double b0){
131     x=x0; y=y0; a=a0;b=b0;  t=0;    }
```