# Simulación del comportamiento dinámico del fenómeno de remoción de masas para flujos rápidos utilizando el método de los elementos discretos

**Daivy Jhonander Mantilla Prada**

**Blass Alexander Troncoso Mendoza**

**Trabajo de Grado para Optar el título de ingeniero civil**

**Director**
**Jorge Alejandro Mendoza Rizo**
**Ph.D. en Ingeniería Civil**

**Codirector**
**Daniel Camilo Román Quintero**
**Ingeniero civil**

**Universidad Industrial de Santander**

**Facultad de Ingenierías Físico Mecánicas**

**Escuela de Ingeniería Civil**

**Bucaramanga**

**2018**

## Agradecimientos

Agradecemos a Dios ya que sin Él nada de esto hubiera sido posible, así como a nuestros padres que nos apoyaron desde el comienzo hasta el fin de nuestra carrera, y nuestros amigos y compañeros de la Universidad. También agradecemos al director del proyecto, Jorge A. Mendoza Rizo, por su dedicación, asesoría y motivación brindada a lo largo del desarrollo de este trabajo y de igual manera al codirector Daniel Román Quintero por su apoyo y contribución.

## Contenido

**Pág.**

## Lista de Figuras

**Pág.**

## Lista de Tablas

**Pág.**

## Resumen

**TITULO**: Simulación del comportamiento dinámico del fenómeno de remoción de masas para flujos rápidos utilizando el método de los elementos discretos[*]

**AUTORES**: Daivy Jhonander Mantilla Prada
Blass Alexander Troncoso Mendoza[**]

**PALABRAS CLAVES**: Método de elementos discretos, deslizamiento de tierra, ESyS-Particle, medio discontinuo.

**DESCRIPCIÓN:**

Los deslizamientos de tierra han sido siempre un problema que afecta a la sociedad, ocasionando pérdidas tanto materiales como humanas, por lo que se hace necesario encontrar mejores estrategias para predecir y medir su poder destructivo. Esta investigación se centra en medir el alcance y velocidad de una masa deslizante de material desagregado a través de simulaciones numéricas, con el fin de dejar las bases para futuras investigaciones que puedan ayudar a mitigar las pérdidas producidas a corto y mediano plazo por el fenómeno de remoción de masas. En este documento se muestra la implementación del Método de Elementos Discretos (Discrete Element Method, DEM), un método numérico usado en el estudio de medios discontinuos, para investigar el comportamiento dinámico de las partículas de tierra de un deslizamiento. El propósito principal es simular, a través de un software libre, un deslizamiento de partículas, calibrándolo con los datos obtenidos de una campaña experimental de un canal de flujo denominado Collaptes V1.0, desarrollado por el grupo de investigación de la UIS: INME. Se logró a través del software ESyS-Particle una simulación del comportamiento dinámico del fenómeno de remoción de masas, obteniendo las variables de desplazamiento y velocidad a lo largo de todo su recorrido, calibrando la simulación de forma cualitativa, modificando los parámetros de repulsión elástica y resistencia del medio. Además se logró estimar la fuerza promedio que tiene cada una de las partículas a lo largo de su trayectoria.

[*] Trabajo de grado

[**] Facultad de Ingenierías FísicoMecanicas. Escuela de Ingeniería Civil. Director: Director Jorge Alejandro Mendoza Rizo Codirector Daniel Camilo Román Quintero

## Abstract

**TITLE:** Simulation of the dynamic behavior of the phenomenon of mass removal for fast flows using the discrete element method [*]

**AUTHORS**: Daivy Jhonander Mantilla Prada
Blass Alexander Troncoso Mendoza[**]

**KEYWORDS:** Element Discrete Method, Landslides, ESyS-Particle, Discontinuum Mechanics.

**DESCRIPTION:**

Landslides have always been a problem that affects society. Causing human and monetary losses, it's necessary to look after better strategies to predict and measure its destructive power. This investigation will center in measure the reach and velocity of a landslide through numerical simulations and settle down bases for future investigations that can help to minimize the losses produced in short or medium term. It was investigated through different literary sources programming language and numerical methods, as well as some basic programming tutorials for the free software. This work implements the Discrete Element Method, DEM, used in the research of discontinuum mechanics, to investigate the particle dynamic behavior of a landslide. The principal end is simulating, through the free software, a model of a particles landslide, calibrating it with the data obtained in an experimental campaign of a flow canal named Collaptes V1.0, developed by the INME group. It has achieved through ESyS-Particle software a simulation of the dynamic behavior of a landslide, getting the variables of displacement and velocity in all its way, calibrating the simulation in a qualitative form, modifying the parameters of elastic repulsion and medium resistance. Besides, it has estimated the average force that every particle has through its trajectory.

[*] Bachelor Thesis

[**] Faculty of Mechanical and Mechanical Engineering. School of Civil Engineering. Director: Director Jorge Alejandro Mendoza Rizo Codirector Daniel Camilo Román Quintero

# Introducción

Los deslizamientos de tierra han sido siempre un problema que afecta a la sociedad, el cual ocasiona pérdidas tanto materiales como humanas, necesitando, a medida que avanza el tiempo, de mejores estrategias para predecir y medir su poder destructivo. En Colombia es muy común, especialmente en la región andina, en donde según datos del DANE del año 2010 se asienta el 75% de la población colombiana (García, 2014), que se presenten deslizamientos del suelo y/o roca ubicados en la parte superior de las capas superficiales de la tierra, ocasionado principalmente por la fuerza de gravedad y exacerbado por las lluvias, sismos, uso del suelo, deforestación, entre otros. Este fenómeno es conocido técnicamente como remoción de masas. Debido al clima húmedo tropical y una alta densidad poblacional, Colombia es uno de los puntos globales de más alto riesgo de remociones en masa accionados por La Niña o la actividad de volcanes glaciados (Mergili, Marchant Santiago, & Moreiras, 2015).

Los métodos numéricos dan resultados muy aproximados a la solución matemática exacta, esto es debido a las simplificaciones usadas para resolver el sistema de ecuaciones diferenciales en la simulación. Parte de la precisión de los cálculos depende de la facilidad de implementación del algoritmo usado, la geometría del problema y de las características especiales y las limitaciones que tenga el sistema usado para simular (los computadores) (Charris, 1999). Los métodos numéricos han sido ampliamente usados en los últimos años debido al fácil acceso a los computadores y a la continua optimización de los procesos realizados por estos para lograr obtener resultados en menor tiempo (Bobet, 2009).

En este trabajo de investigación se implementa el método de elementos discretos (DEM), desarrollado por Cundall y Strack (Cundall & Strack, 1979), ya que este es utilizado para el estudio de medios considerados como discontinuos, simulando el comportamiento mecánico de un medio formado por un conjunto de partículas que interaccionan entre sí a través de sus puntos de contacto. Además, se pueden formar medios de diferentes tamaños y formas, ya que la disposición de las partículas dentro del conjunto es aleatoria. Este método numérico ha demostrado ser prometedor para analizar el fenómeno de remoción de masas. Por ejemplo, se han realizado modelos acoplados de DEM y dinámica de fluidos para analizar el comportamiento del colapso de una columna de suelo (Zhao, 2014).

Debido a la alta amenaza que el fenómeno de remoción de masas representa para la sociedad, es importante crear estudios que permitan entender y describir las variables involucradas en el fenómeno de remoción de masas. En el presente proyecto se desea saber ¿Cuál sería la velocidad y el alcance de una masa de suelo y/o roca que se desliza por una ladera? y así contribuir al desarrollo de estrategias de mitigación necesarias para evitar los potenciales desastres.

# 1. Objetivos

## 1.1 Objetivo general

Representar por medio del Método de Elementos Discretos (DEM) el comportamiento del flujo de material desagregado en aras de la estimación del alcance y la velocidad de una masa deslizante,

con aplicación al fenómeno de remoción de masas para deslizamientos clasificados entre rápidos y extremadamente rápidos.

### 1.2 Objetivos específicos

- Desarrollar la simulación de elementos discretos en 2D en un software libre, que permita la obtención de las variables de desplazamiento y velocidad de una masa deslizante a lo largo de su trayectoria.
- Calibrar la simulación mediante el uso de los datos obtenidos en una campaña experimental del canal de flujo denominado Collaptes V1.0 desarrollado por el grupo INME.

## **2.** Marco Teórico

### 2.1 Taludes y deslizamientos

Un talud o ladera, se define como una masa de tierra en la cual su pendiente es diferente de cero. Es llamada ladera cuando su formación ha sido de forma natural, mientras que se denomina talud cuando su origen fue de forma artificial. Un talud estable puede fallar, dando como consecuencia un deslizamiento, por diversos factores, tales como los cambios topográficos, factores antrópicos, la actividad sísmica, flujos subterráneos de agua u otros factores que modifiquen el estado natural de la estabilidad (Suarez, 2012).

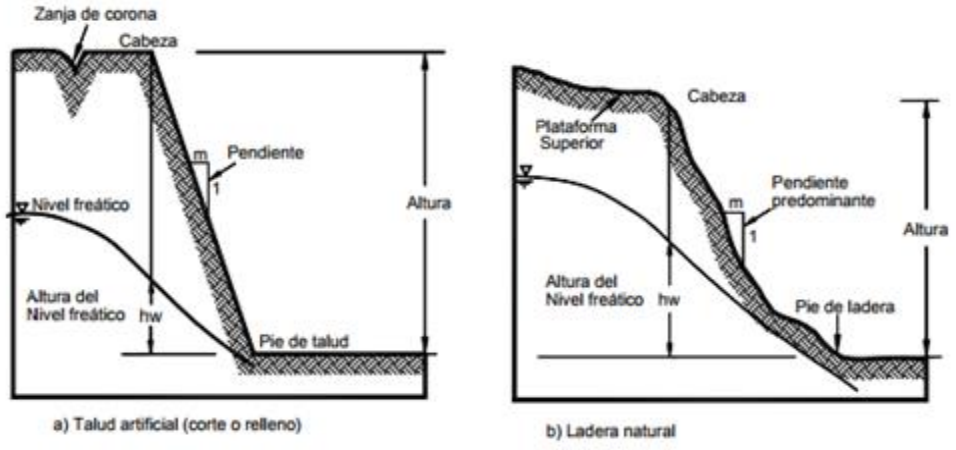



*Figura 1*. a-b: Diagrama para la descripción de las partes de un talud y ladera (Suarez, 2012).

Los deslizamientos, llamados “Landslides” en inglés, consisten en “movimientos de masas de roca, residuos de tierra, hacia debajo de un talud” (Cruden, 1991). Durante el deslizamiento, se pueden producir distintas fallas, debido a que el esfuerzo cortante del suelo es incapaz de mantener el equilibrio con el esfuerzo ejercido por la masa deslizante.

Los deslizamientos de tierra son procesos geológicos muy destructivos, causando cambios en la morfología del terreno, daños ambientales, daños en infraestructura, destrucción de viviendas, bloqueo de ríos y carreteras, muertes humanas, etcétera. Se han podido reportar daños por valores de billones de dólares cada año (Brabb & Harrod, 1989). A pesar de la amenaza que estos desastres representan, un gran porcentaje de ellos son evitables anticipándose al problema y tomando medidas preventivas (Suarez, 2012).

Algunos de los deslizamientos con más impacto socioeconómico que ocurrieron en el último siglo, se especifican en la tabla 1.

Tabla 1.

*Deslizamientos con más impacto socioeconómico en el último siglo. (Zhao, 2014)*

| Date | Place | Name | Trigger | Casualties |
|---|---|---|---|---|
| 19/05/1919 | Indonesia | Kelut lahars | Volcano | 5,110 |
| 16/12/1920 | China | Haiyuan landslide | Earthquake | >100,000 |
| 25/08/1933 | China | 1933 Diexi landslide | Earthquake | 3,100 |
| 13/12/1941 | Peru | Huaraz debris flow | Earth-dam break | 4,000-6,000 |
| 1953 | Japan | Arida river landslide | Typhoon | 1,046 |
| 1958 | Japan | Kanogawa landslides | Rainfall | 1,094 |
| 10/01/1962 | Peru | 1962 Nevado Huascaran debris avalanche | Collapse of glacier | 4,000-5,000 |
| 31/05/1970 | Peru | 1970 Nevado Huascaran debris avalanche | Earthquake | >22,000 |
| 13/11/1985 | Colombia | Armero tragedy | Volcano | 23,000 |
| 14/12/1999 | Venezuela | Vargas tragedy | Heavy storm | 30,000 |
| 09/08/2009 | China | Taiwan, Xiaolin | Typhoon | 600 |
| 08/08/2010 | China | Zhouqu country mudslide | Rainfall | 1,287 |

Existen varios factores que pueden desencadenar un deslizamiento, se pueden separar en 3 grandes categorías: procesos geológicos, eventos morfológicos y actividades humanas (Zhao, 2014), en donde las causas principales son:

- Infiltración del agua debido a fuertes lluvias, cambios en el nivel freático y presión de poros.
- La actividad sísmica provoca movimientos en el suelo que puede desencadenar el desprendimiento de tierra y roca.
- La lava expulsada por los volcanes derrite gran cantidad de suelo o roca en poco tiempo, lo cual hace que se acelere la cantidad de material sólido en el deslizamiento y cause catástrofes.

## 2.4 Método de Elementos Discretos

Los métodos numéricos son “técnicas mediante las cuales es posible formular problemas de tal forma que sean resueltos por operaciones aritméticas” (Chapra & Canale, 1996). Estos no buscan una solución exacta del problema sino una aproximación con un margen de error pequeño.

La importancia de su uso radica en el tiempo en que se encuentra una solución, ya que un cálculo manual que puede tomar varios días, una computadora lo puede realizar en cuestión de segundos con una buena precisión.

Los sistemas para modelar se dividen en dos grupos: continuos o discontinuos. En un sistema continuo, el dominio (e.g., conjunto de partículas) se comporta como un único cuerpo, en el cual existe una condición de continuidad que impide, en general, la división del dominio. En un sistema continuo, el dominio está dividido en varios elementos, los cuales están conectados entre sí hasta donde lo indiquen sus fronteras. En nuestro estudio, en caso de modelar la masa como continuo, cada partícula no podrá moverse libremente, pues estará sujeta a una condición que no permite que se desvincule (tengas grandes deformaciones) de sus vecinos. El conjunto de elementos formados actuará como un único sólido. Por otro lado, en un sistema discreto, los elementos pueden o no estar conectados, y el movimiento de cada uno dependerá de las interacciones que haya entre ellos o en el medio (Vergara & Franco, 2012). En el caso de los deslizamientos de tierra, no es convincente tomar el suelo como un elemento que se mueve de forma uniforme y permite deformaciones pequeñas en la misma masa, sino como un conjunto de partículas con libertad de movimiento, que interactúan entre sí, por lo que es conveniente modelar un sistema discontinuo.

El método de Elementos Discretos (conocido como DEM por sus siglas en inglés) se encarga de modelar los sistemas discontinuos. El DEM es una familia de métodos numéricos que determina el movimiento de cada partícula de un medio, a través de la interacción entre ellas, las cuales están afectadas por fuerzas como la gravedad. De acuerdo con Cundall y Hart, un Método de Elementos Discretos debe: 1) Permitir desplazamientos y rotaciones finitas de cuerpos rígidos, incluyendo separación; y 2) reconocer automáticamente nuevos contactos entre los cuerpos durante los cálculos.

La forma de cada una de las partículas es variable, ya que pueden ser discos circulares regulares (2D), esferas (3D) y polígonos o poliedros irregulares. Incluso algunos métodos pueden modelar las partículas como deformables, pero sólo es usado cuando la deformación de cada una de ellas sea considerable.

Una de las ventajas de DEM sobre otros métodos de naturaleza continua es que toma el movimiento de cada partícula como independiente, lo cual es mucho más cercano a la realidad para el modelamiento de problemas granulares, especialmente en el campo de mecánica de rocas y suelos.

## 2.5 ESyS-Particle

ESyS-Particle es un software libre para modelos numéricos basados en partículas. El software implementa el método de elementos discretos (DEM), una técnica muy usada para modelar procesos que involucran deformaciones, deslizamientos de tierra o fragmentaciones. ESyS-Particle está diseñado para ejecutarse en supercomputadores paralelos, o computadoras con múltiples núcleos que usen sistemas operativos basados en Windows o Linux (ESyS-Particle,

2004). ESyS-Particle ha sido usado en muchos modelos que requieren un análisis del comportamiento de las partículas desagregadas tales como deslizamientos de tierra, fragmentación de roca o partículas en deslizamiento, por nombrar algunas.

El Centro para Computación de Geociencia en la Universidad de Queensland, Brisbane, Australia es el desarrollador y responsable de ESyS-Particle, el cual fue desarrollado dentro de la empresa desde 1994. Las bases de los algoritmos del software fueron desarrolladas por el profesor Peter Mora y sus compañeros de trabajo desde 1992 en IPG, París. Fue originalmente llamado Lattice Solid Model y luego LSMEarth antes de que la gran instalación nacional de investigación: El Simulador Australiano de los sistemas computacionales de la Tierra (ACcESS) lo lanzara de forma libre bajo la Licencia Apache 2.0 (ESyS-Particle, 2004).

## 3. Metodología

Se realizó una investigación en donde se tuvieron en cuenta tres softwares que han sido usados para implementar el método de elementos discretos: LMGC90, ESyS-Partice y LIGGGHTS. Se decidió realizar la simulación en ESyS-Particle debido a que existe más material literario acerca de sus líneas de código, con un tutorial básico con varios ejemplos de forma organizada y explicada, y además con un foro de preguntas en su página de Internet (ESyS-Particle, 2004). De la investigación se concluyó que el sistema operativo óptimo para el uso del software libre era Ubuntu 16.04.3 LTS, además de ser necesario un editor de texto (Spyder), puesto que el programa

no cuenta con una interfaz propia, por lo que las líneas de código se escriben en el editor y la simulación se corre a través del terminal del sistema operativo usando las bibliotecas del software libre y el lenguaje de Python. El programa genera unos archivos con toda la información obtenida en la simulación, lo que por medio de otra línea de código se pueden convertir en archivos que pueden ser leídos por medio de un visualizador como Paraview 5.0.1.

Se realizó la programación teniendo en cuenta las medidas del canal de flujo del grupo INME Collaptes V1.0. Con el informe (Román, 2017) se determinó el tamaño y la densidad de las partículas a simular. A través de múltiples pruebas se ajustan los datos obtenidos de la simulación con los hallados en dicho informe.

Primeramente, se estudió el lenguaje de programación de Python, ya que este es el utilizado por el sistema operativo Ubuntu, para luego proceder con la instalación del software libre llamado ESyS-Particle.

El programa ESyS-Particle no cuenta con una interfaz, por lo que todo el código de programación fue escrito en un editor de texto llamado Spyder, usando el lenguaje de programación de Python. ESyS-Particle funciona como una biblioteca, en donde sus librerías deben ser importadas antes de ser usadas. Para nuestra simulación, se importaron las librerías mostradas en el Apéndice A

Toda simulación en ESyS-Particle comienza con la creación de un objeto de simulación llamado LsmMpi, el cual funciona como un contenedor al que se le agregan todos los componentes de la simulación y las interacciones que existen entre ellas. En esta línea de código se coloca el número de procesadores que se van a usar para la simulación, usando uno en esta investigación. También se especifica el tipo de partículas a usar y el algoritmo usado para detectar las interacciones alrededor de la partícula. Luego, es necesario especificar el número de pasos en el tiempo a

computar durante la simulación y el incremento del paso en segundos. Para la simulación se computaron un total de 10,000 pasos con un incremento de 0.001 segundos en cada uno.

Luego, es necesario definir el dominio espacial de la simulación, ya sea en 2D o 3D. Toda partícula o pared creada fuera de este dominio es eliminada y no se tiene en cuenta en ningún tipo de cálculo. Para la simulación, se utiliza un dominio espacial de 40 metros de alto, 40 metros de largo y 2 metros de ancho, como se aprecia en la figura 2:

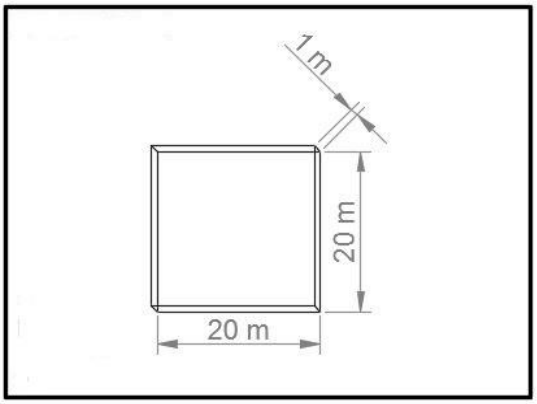



*Figura 2*. Domino espacial de la simulación.

Después de crear el dominio espacial de la simulación y el algoritmo de interacción entre los vecinos, se procede a crear los elementos que interactúan en la simulación. El programa ESyS-Particle primero crea todos elementos de la simulación sin tener en cuenta las interacciones entre ellas, por lo que el orden de los comandos al momento de crear los elementos es irrelevante. Para el código de programación, primeramente, se crean las partículas por medio de una 'caja de partículas' (opción inicial para la generación de partículas para la primera iteración), teniendo en cuenta las partículas usadas en el proyecto del grupo INME, las cuales tenían un radio de 0.01 m y se tomó una densidad de 1550 kg/$m^3$. Tomando el punto de referencia 0 desde el comienzo del pie del talud, se calculan las coordenadas en donde se creará la caja de partículas.

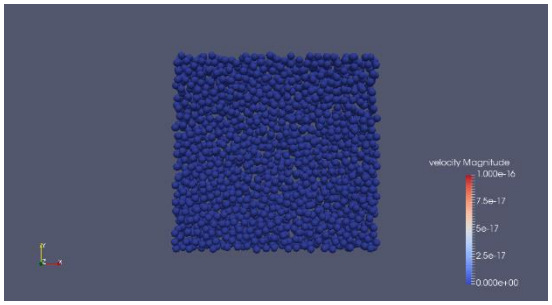



*Figura 3*. Caja de partículas.

Consecuentemente, se crearon las paredes que actuarán como el talud por el que se deslizan las partículas, es decir, la simulación computacional del canal de flujo Collaptes V1.0. El modelo simulado por el programa se asemeja a la ilustración mostrada en la figura 4:

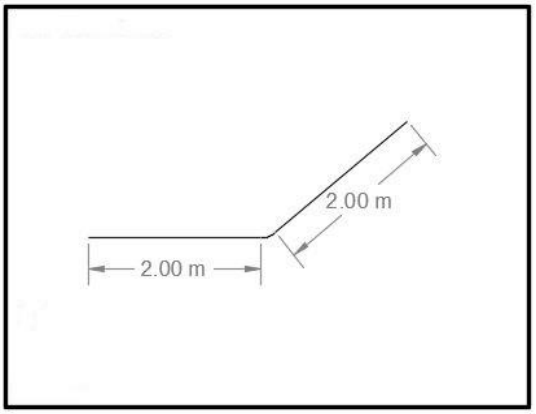



*Figura 4***.** Medidas del canal de flujo en la simulación. Fuente: Autores.

Para la simulación se utilizaron dos tipos de paredes: con planos finitos y con planos infinitos. En el programa ESyS-Particle, ambas paredes tienen un lado activo y un lado inactivo, en donde este último no produce comportamientos de interacción prescritos en la simulación. La forma correcta de colocar los lados de la pared depende del tipo de pared que se esté programando.

Para crear una pared de plano infinito, es necesario conocer un punto dentro del plano y un vector normal a ese plano, en donde el sentido es el que define los lados activo e inactivo. Las paredes de plano infinito fueron usadas para definir las fronteras y la parte horizontal del canal.

Para los planos finitos, el procedimiento es un poco más extenso, puesto que el software utiliza un formato de malla a través de triángulos para definir los segmentos de una pared, en donde se especifica el número de nodos y de triángulos creados. En el Apéndice B se encuentra un ejemplo de este tipo de muros.

Estas paredes son usadas para simular el talud. En cuanto a la parte inferior de éste, para tratar de simular de una forma más precisa al modelo realizado por el grupo INME, se realizó una serie de paredes finitas que simulaban la curva que suavizaba el choque de las partículas, Un ejemplo de pared finita se aprecia en la figura 5. Las medidas del canal de flujo se pueden apreciar en el Apéndice C.

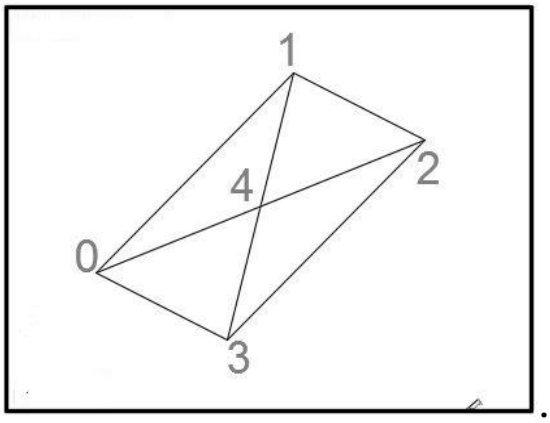



*Figura 5*. Ejemplo de pared finita. Autores.

Con las partículas y las paredes creadas, lo siguiente es definir las interacciones que existen: entre partículas y entre las paredes y las partículas. Empezando con las interacciones entre

partículas, la manera óptima para modelar una masa de tierra es estableciendo una fricción entre ellas, necesitando de algunos parámetros en su código. El parámetro usado para definir la repulsión elástica es *normalK*, mientras que los usados para definir la fricción son el coeficiente de fricción (*dynamicMu*) y la resistencia al corte (*shearK*). Cuando dos partículas chocan, se crea un punto de corte. Las fuerzas que rodean las partículas causarán que estas se comiencen a deslizar unas sobre otras con el punto de corte resistiendo el movimiento. Cuando la fuerza cortante es mayor a la fuerza normal multiplicada por el coeficiente de fricción, se desencadena el deslizamiento (debido a que la máxima fuerza cortante es gobernada por la fuerza normal y el coeficiente de fricción, es decir, la ley de Coulomb) (Weatherly, Hancock, & Boros, 2014). En el momento de crear el modelo, se usan los parámetros de otras simulaciones (Weatherly, Hancock, & Boros, 2014) y de ahí se establece el punto de partida para el proceso de calibración, del cual se hablará más adelante.

Para la interacción entre las partículas y la pared, se establece sólo la repulsión elástica, puesto que el programa aún no es capaz de simular paredes con fricción, como se evidencia en el Apéndice A.

En ESyS-Particle, la rigidez elástica se denota por el parámetro *normalK*, que es la interacción entre la partícula y la pared, en N/m. Es importante saber que, si la rigidez elástica es muy pequeña, la pared no podrá soportar el peso de la partícula y la atravesará; por otro lado, si es demasiado grande, la partícula rebotará de forma irreal (Weatherly, Hancock, & Boros, 2014).

La aceleración de la gravedad es implementada de la misma manera que las anteriores interacciones, en donde se especifica un nombre dado para la fuerza y la dirección, sentido y magnitud. Para la simulación, se utilizó el valor de 9.81 $m/s^2$.

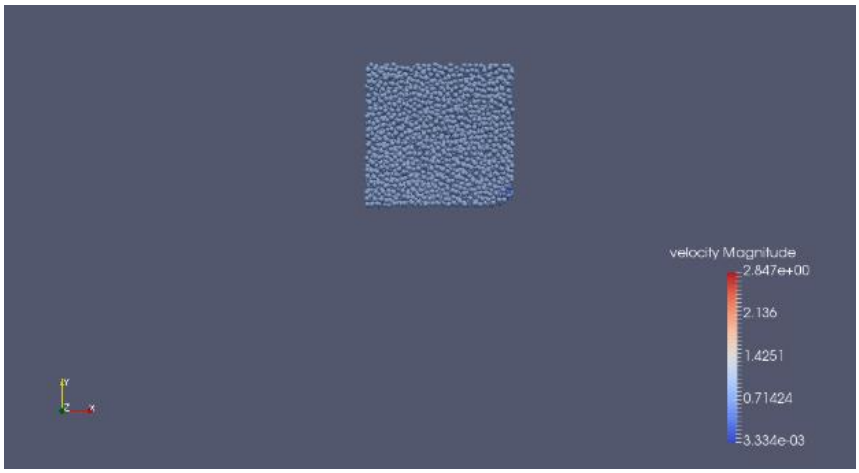
Por último, se añade un parámetro llamado *Bulk Viscosity*, el cual es una fuerza de amortiguamiento proporcional a la velocidad instantánea de cada partícula en dirección contraria

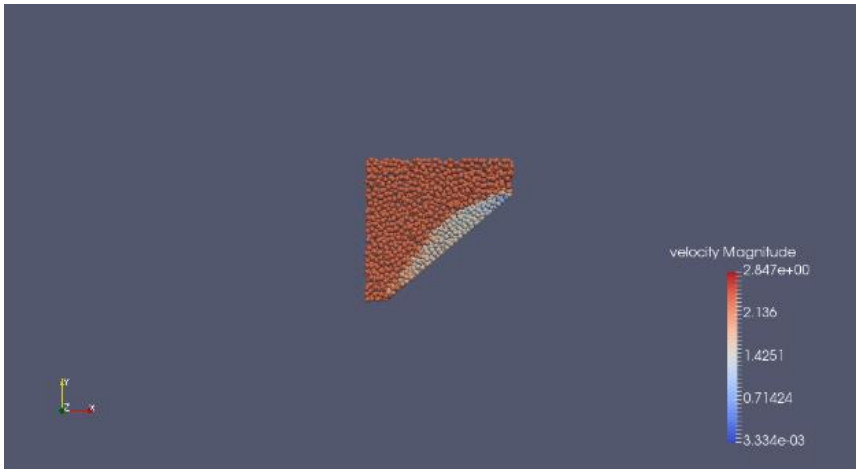
al movimiento (Weatherly, Hancock, & Boros, 2014). Se podría entender que esta fuerza es lo más aproximado a la resistencia del aire en el medio.

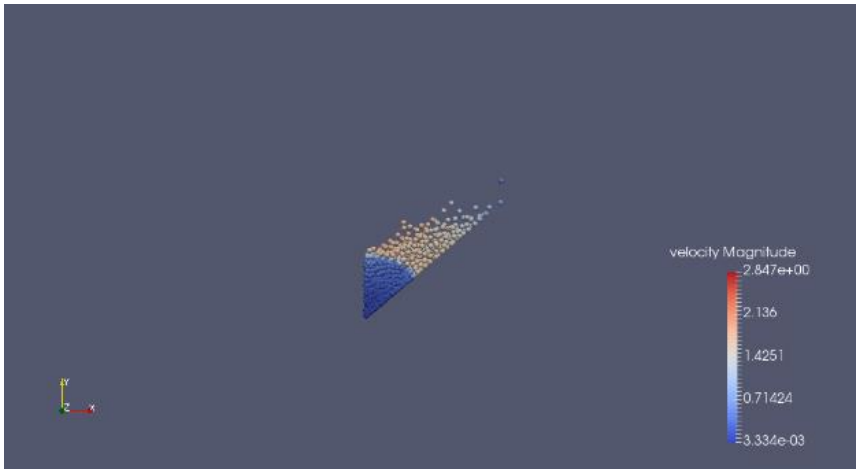
Con los componentes básicos para ejercer la simulación, es necesario definir un intervalo de tiempo en donde tomar los datos necesarios para su calibración. ESyS-Particle posee un módulo conocido como CheckPointer. Este módulo genera unos archivos con información de las partículas, que luego, a través de la terminal del sistema operativo Ubuntu, pueden ser transformados en archivos que pueden ser leídos por un visor, en este caso Paraview. El intervalo de tiempo escogido es de 10 s desde que empieza el deslizamiento, con un registro de datos en intervalos de 0.1 s. El procedimiento para correr la simulación de manera correcta se encuentra explicado en el A.

### 3.1 Calibración

Debido a que el software crea ‘cajas de partículas’, es necesario ajustar su forma al modelo Collaptes V1.0. Para lograr esto, se crea una pared vertical de plano infinito inmediatamente al lado de donde se crean las partículas, en donde se espera que lleguen al estado de reposo estático por medio de observaciones en distintas simulaciones, y una vez encontrado este instante de tiempo, se mueve dicha pared en la dirección del desplazamiento de las partículas a una velocidad muy alta, para que no interfiera con el movimiento de estas


velocity Magnitude
2.847e+00
2.136
1.4251
0.71424
3.334e-03


velocity Magnitude
2.847e+00
2.136
1.4251
0.71424
3.334e-03


velocity Magnitude
2.847e+00
2.136
1.4251
0.71424
3.334e-03

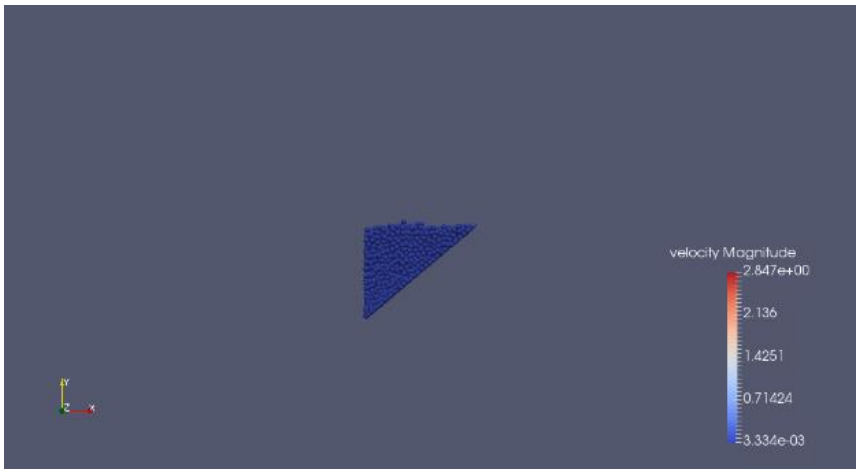



*Figura 6***.** Estabilización de las partículas.

El proceso de calibración se hace de forma cualitativa, observando las distintas fases del deslizamiento. Hasta la fecha, el programa ESyS-Particle no es capaz de simular paredes friccionantes, por lo que se hizo el ajuste a través de dos parámetros: la repulsión elástica entre las partículas y la pared, y la viscosidad del medio.

Para la repulsión elástica, es necesario ajustar el parámetro normalK, el cual se conoce como la rigidez elástica entre la pared y la partícula. Para el parámetro *Bulk Viscosity* es necesario ajustar el valor de *viscosity*, en donde si adquiere un valor demasiado grande, las partículas se frenarán, pero si el valor es muy pequeño, las partículas no encontrarán resistencia y se moverán demasiado rápido. Al calibrar la simulación, los valores que mejor se ajustaron se pueden apreciar en la tabla 2:

Tabla 2**.**

*Parámetros de la simulación.*

| PARÁMETRO | CÓDIGO | VALOR | UNIDAD |
|---|---|---|---|
| **Medio** | | | |
| Gravedad | g | 9.81 | [m/s2] |
| Viscosidad | viscosity | 1.7 | [P] |
| **Muros** | | | |
| Rigidez elástica | normalK | 100 | [N/m] |
| **Partículas** | | | |
| Densidad | ρ | 1550 | [kg/m3] |
| Rigidez elástica | normalK | 1 | [N/m] |
| Coeficiente de fricción | dynamicMu | 0.6 | [kg/(m*s)] |
| Resistencia al corte | shearK | 100 | [N/m] |

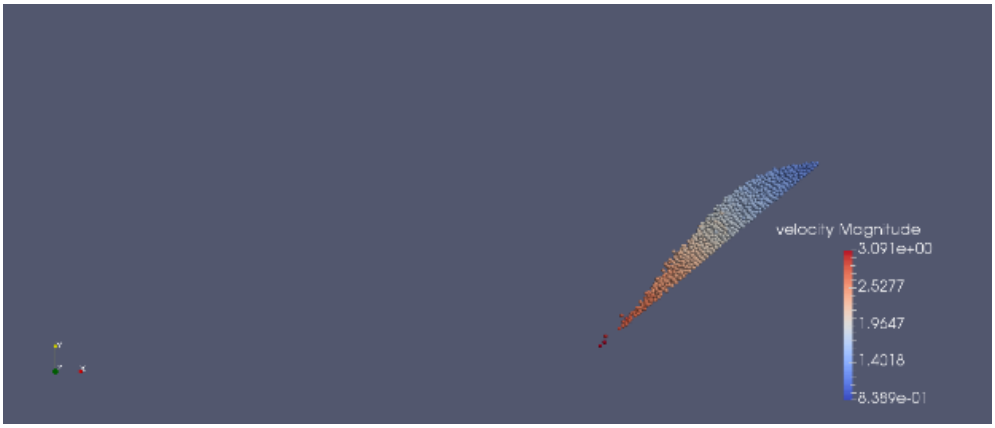
## 4. Resultados

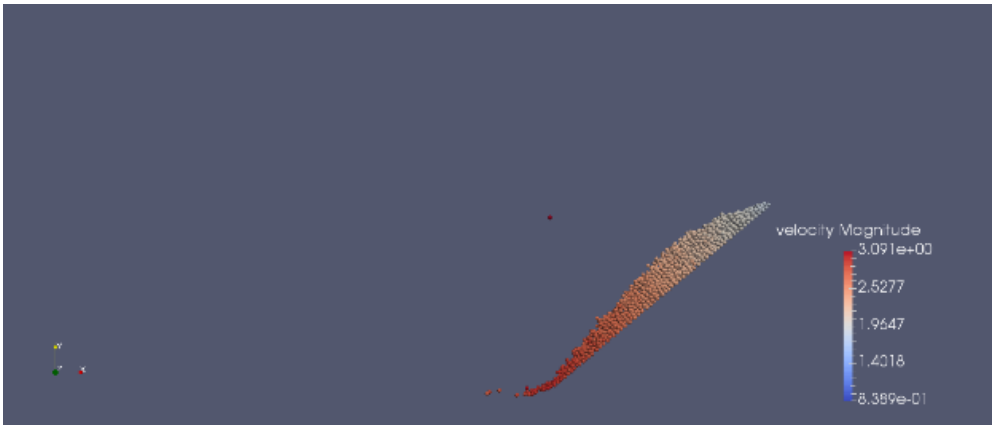
Por medio del software Paraview es posible visualizar los resultados obtenidos de la simulación, como se muestra en la Figura 7.

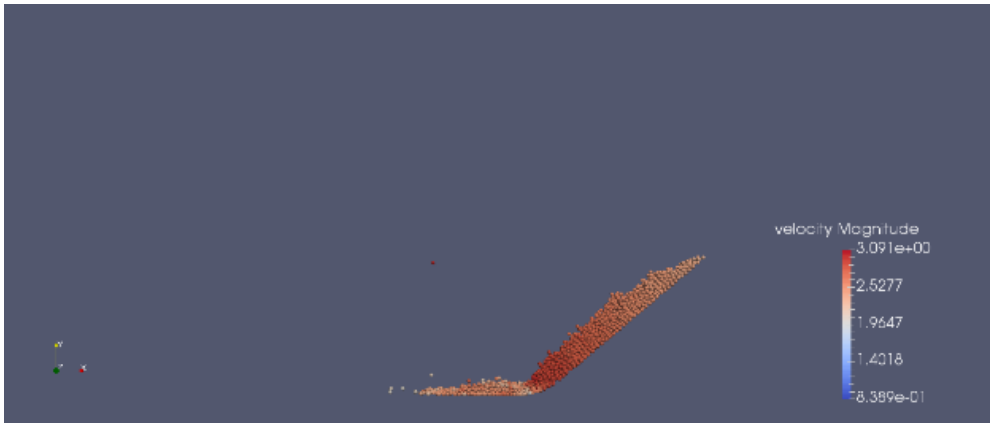
En el Apéndice  se encuentra una comparación entre el modelo Collaptes V1.0 realizado por el grupo INME y la simulación mostrada en este trabajo de investigación.

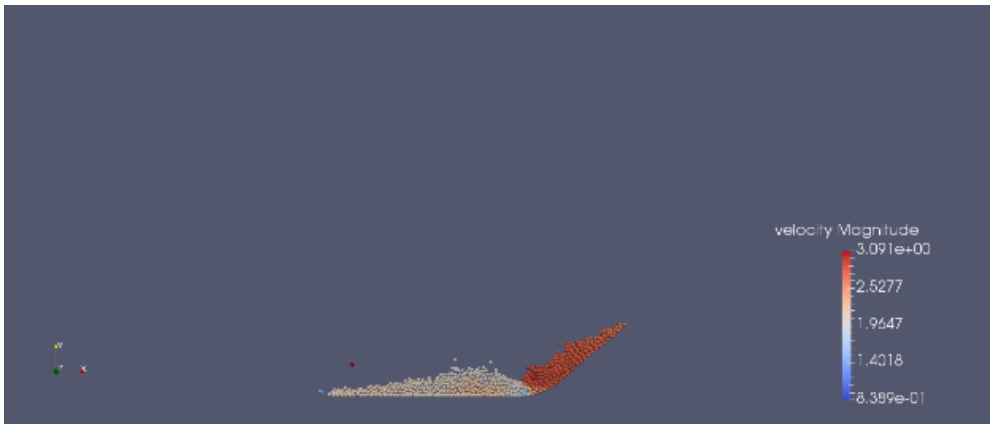

velocity Magnitude
3.091e+00
2.5277
1.9647
1.4018
8.389e-01


velocity Magnitude
3.091e+00
2.5277
1.9647
1.4018
8.389e-01


velocity Magnitude
3.091e+00
2.5277
1.9647
1.4018
8.389e-01


velocity Magnitude
3.091e+00
2.5277
1.9647
1.4018
8.389e-01


velocity Magnitude
3.091e+00
2.5277
1.9647
1.4018
8.389e-01


velocity Magnitude
3.091e+00
2.5277
1.9647
1.4018
8.389e-01

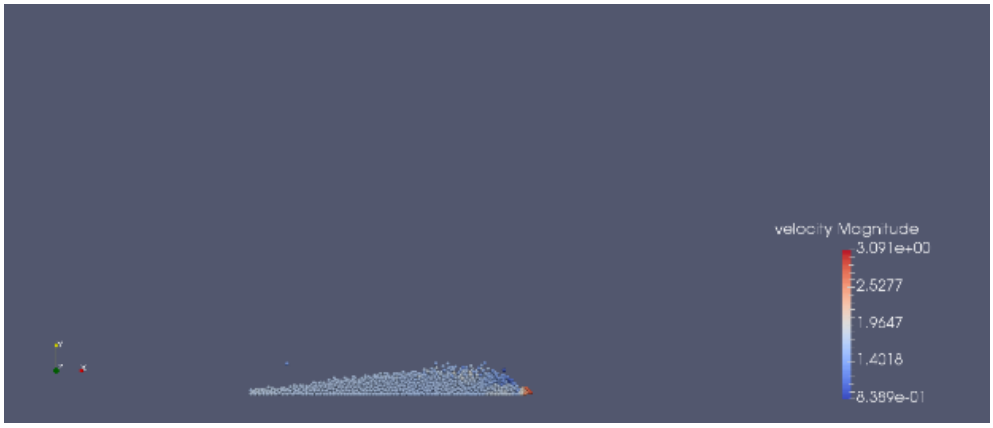



*Figura 7*.-a, b,c,d,e,f,g,h,i. Sucesión de *snapshot* tomadas cada 0.2 segundos.

En la Gráfica 1, se muestra el desplazamiento promedio de las partículas, en donde se logra observar en el segundo 4 estas se detienen. Se obtiene de igual manera la gráfica de la velocidad promedio, y se encuentra un comportamiento esperado, en donde aumenta su velocidad desde el comienzo del deslizamiento, alcanzando su velocidad máxima en el pie del talud, y luego disminuye hasta llegar al estado de reposo, como se muestra en la Gráfica 2.

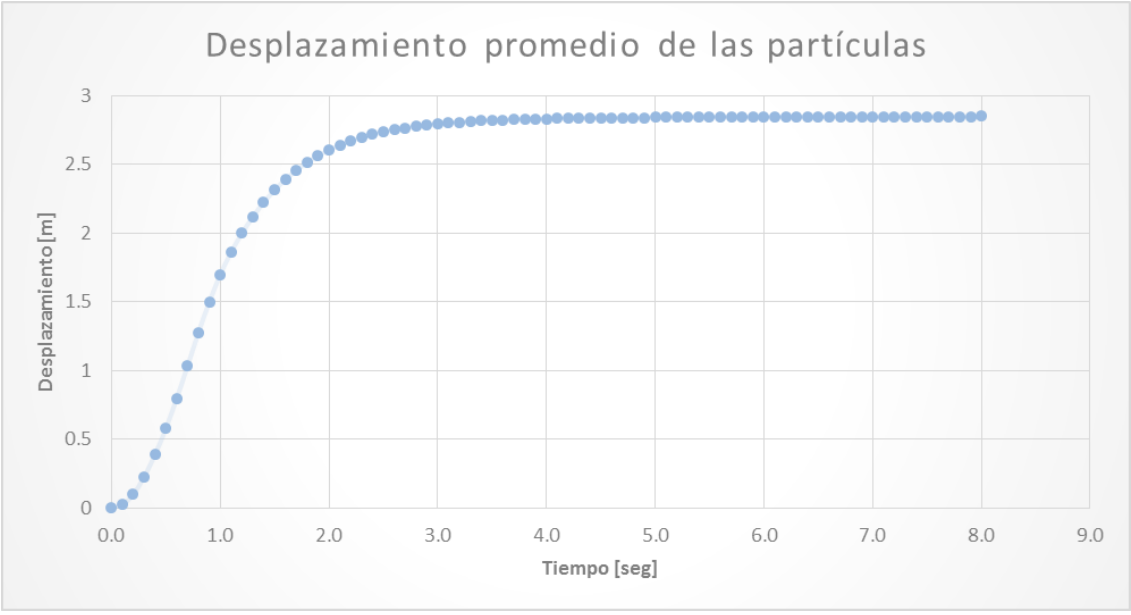



*Figura 8.* Desplazamiento promedio de las partículas.

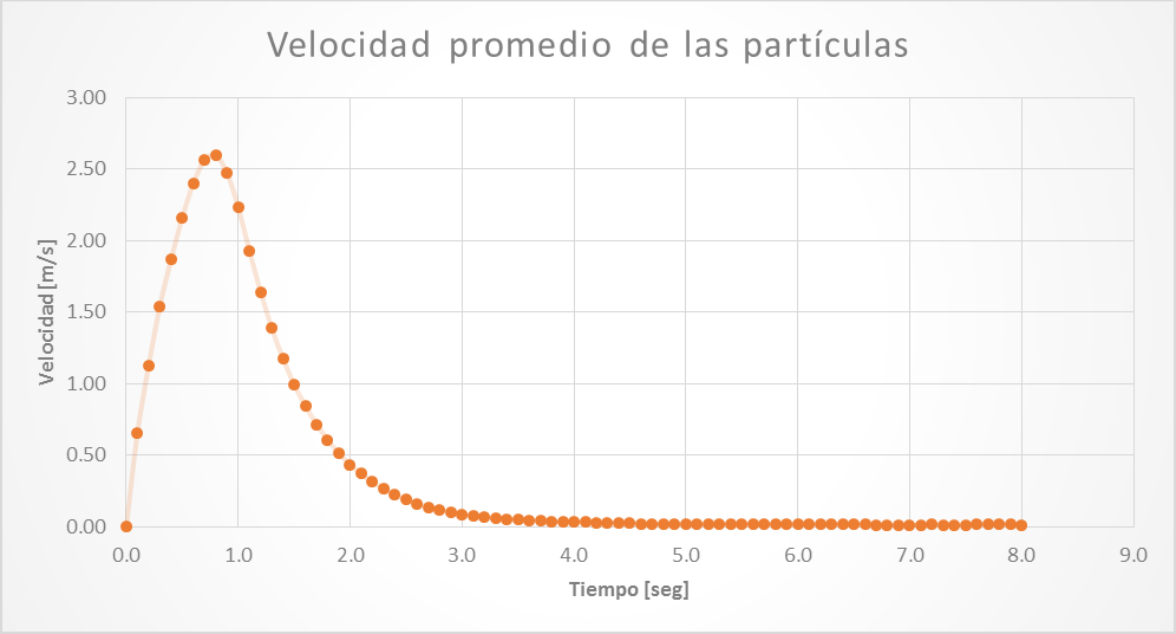



*Figura 9.* Velocidad promedio de las partículas.

Además, teniendo los datos de las velocidades en intervalos de 0.1 segundos, se encuentra la aceleración promedio de las partículas, y teniendo en cuenta la masa, se puede hallar de igual manera la fuerza promedio de cada una de ellas, obteniendo estas variables en función del tiempo, representadas en las Gráficas 3 y 4, respectivamente, los cuales fueron tomados desde el pie del talud.

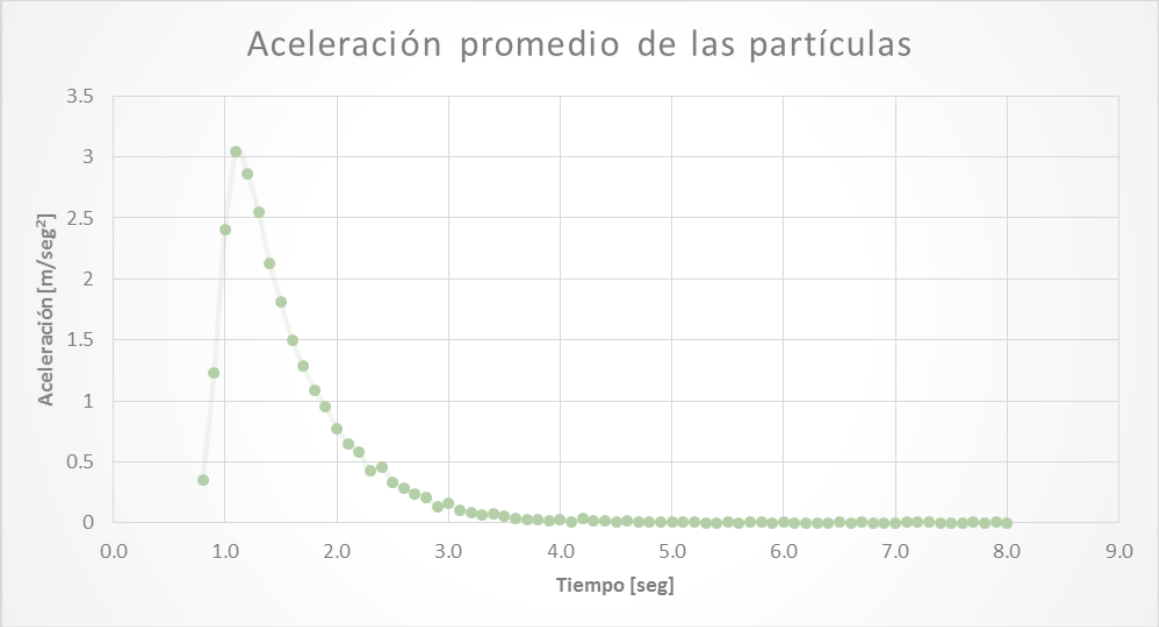



*Figura 10***.** Aceleración promedio de las partículas.

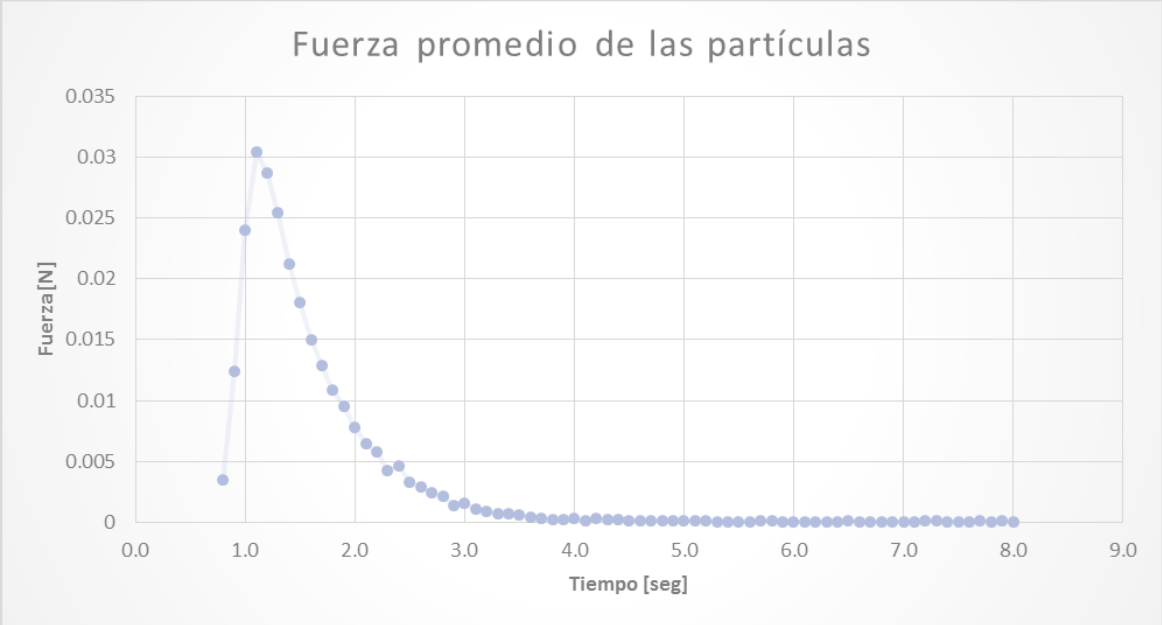



*Figura 11***.** Fuerza promedio de las partículas.

# 5. Conclusiones

El presente proyecto de investigación realiza una representación, por medio del Método de Elementos Discretos (DEM) el comportamiento del flujo de material desagregado en aras de la estimación del alcance y la velocidad de una masa deslizante, con aplicación al fenómeno de remoción de masas para deslizamientos clasificados entre rápidos y extremadamente rápidos.

A través de la programación en el software ESyS-Particle se realizaron varias simulaciones del comportamiento dinámico del fenómeno de remoción de masas a escala de laboratorio (con un volumen de 3 litros de material desagregado) por medio del método de elementos discretos, obteniendo las variables de desplazamiento y velocidad a lo largo de todo su recorrido.

Se calibró el modelo de forma cualitativa, a través del modelo experimental Collaptes V1.0 modificando los parámetros de repulsión elástica y resistencia del medio, logrando así llegar a unos desplazamientos muy cercanos a los obtenidos por el grupo INME. Además se logró medir la fuerza promedio que tiene cada una de las partículas a lo largo de su trayectoria.

Haciendo unos ajustes a la simulación mostrada, se puede representar un caso de la vida real, por medio del cual se podrían tomar acciones preventivas en pro de las viviendas y vías aledañas a los taludes, ya que se pudo comprobar que se puede hallar la velocidad y el alcance. No obstante, es necesario tener en cuenta la discretización de la masa y el tamaño de las partículas, ya que de la cantidad de estas depende el tiempo necesario para realizar la simulación.

## Referencias Bibliográficas

Bobet, A. (2009). *Numerical Methods in Geomechanics.*

Brabb, & Harrod. (1989). In *Landslides: Extent and Economic Significance.*

Chapra, S. C., & Canale, R. P. (1996). *Métodos numéricos para ingenieros con aplicaciones en computadoras personales.* México: McGraw-Hill.

Charris, I. F. (1999). *Métodos Numéricos: Un Primer Curso.* Medellín.

Cruden, D. M. (1991). A Simple Definition of a Landslide. *Department of Civil Engineering of Alberta*.

ESyS-Particle. (2004). *ESyS-Particle*. Retrieved from https://launchpad.net/esys-particle

Román, D. (2017). *Análisis experimental del movimiento del fenómeno de remoción de masas en materiales granulares.* Bucaramanga.

Suarez, J. (2012). Deslizamentos: Análisis Geotécnico. In J. Suarez, *Deslizamentos: Análisis Geotécnico.*

Vergara, J., & Franco, K. (2012). Uso del Método de Elementos Discretos para el Modelamiento Geomecánico a Escala de Pozo.

Weatherly, D., Hancock, W., & Boros, V. (2014). *ESyS-Particle Tutorial and User's Guide Version 2.3.1*.

Zhao, T. (2014). *Investigation of Landslide-Induced Debris Flows by the DEM and CFD.* Oxford: University of Oxford.

# Apéndices

## Apéndice A Código de la simulación

```
#import the appropriate ESyS-Particle modules:
from esys.lsm import *
from esys.lsm.util import *
from esys.lsm.geometry import *
from WallLoader import WallLoaderRunnable

#La primera línea de código importa ciertas subrutinas y comandos requeridos en todas las
#simulaciones de ESyS-Particle. La segunda importa una serie de comandos que permiten
#especificar parámetros tales como la velocidad, posición, aceleración, entre otros. La
#tercera línea de código permite definir la geometría de los parámetros creados. La cuarta
#línea de comando importa un código de programación escrito en otro documento del cual se
#hablará más adelante.

k = 1.000e+02
#instantiate a simulation object and
#initialise the neighbour search algorithm:
sim = LsmMpi (numWorkerProcesses = 1, mpiDimList = [1,1,1])
sim.initNeighbourSearch (
   particleType = "NRotSphere",
   gridSpacing = 2.5000,
   verletDist = 0.1000
)

#specify the number of timesteps and the timestep increment:
sim.setNumTimeSteps (100000)
sim.setTimeStepSize (1.0000e-04)

#specify the spatial domain for the simulation:
domain = BoundingBox(Vec3(-20,-20,-1), Vec3(20,20,1))
sim.setSpatialDomain(domain)

#construct a block of particles with radii in range [0.2,0.5]:
geoRandomBlock = RandomBoxPacker (
   minRadius = 0.01000,
   maxRadius = 0.01000,
   cubicPackRadius = 2.2000,
   maxInsertFails = 1000,
   bBox = BoundingBox(
      Vec3(1.188, 1.350,-0.0500),
      Vec3(1.708, 1.850, 0.0500)
   ),
   circDimList = [False, False, False],
   tolerance = 1.0000e-05
)

geoRandomBlock.generate()
geoRandomBlock_particles = geoRandomBlock.getSimpleSphereCollection()

#add particles to simulation one at a time, tagging those in layers of 2m
```

```
for pp in geoRandomBlock_particles:
   centre = pp.getPosn()
   Y = centre[1]
   if (int(Y%4) < 2):
      pp.setTag(1) # layer 1
   else:
      pp.setTag(2) # layer 2
   sim.createParticle(pp) # add the particle to the simulation object
   sim.setParticleDensity(
    tag = 0,
    mask = -1,
    Density = 1550
    )
#add a wall as a floor for the model, wall 1:
sim.readMesh(
   fileName = "01floor.msh",
   meshName = "Wall1"
)
#add a wall as a floor for the model, wall 2:
sim.readMesh(
   fileName = "02floor.msh",
   meshName = "Wall2"
)
#add a wall as a floor for the model, wall 3:
sim.readMesh(
   fileName = "03floor.msh",
   meshName = "Wall3"
)
#add a wall as a floor for the model, wall 4:
sim.readMesh(
   fileName = "04floor.msh",
   meshName = "Wall4"
)
#add a wall as a floor for the model, wall 5:
sim.readMesh(
   fileName = "05floor.msh",
   meshName = "Wall5"
)
#add a wall as a floor for the model, wall 6:
sim.readMesh(
   fileName = "06floor.msh",
   meshName = "Wall6"
)
#add a wall as a floor for the model, wall 7:
sim.readMesh(
   fileName = "07floor.msh",
   meshName = "Wall7"
)
#add a wall as a floor for the model, wall 8:
sim.readMesh(
   fileName = "08floor.msh",
   meshName = "Wall8"
)
#add a wall as a floor for the model, wall 9:
sim.readMesh(
   fileName = "09floor.msh",
   meshName = "Wall9"
)
#add a wall as a floor for the model, wall 10:
sim.readMesh(
   fileName = "10floor.msh",
```

```
   meshName = "Wall10"
)
#add a wall as a floor for the model:
sim.readMesh(
   fileName = "11floor.msh",
   meshName = "slope"
)
#add a left side wall to the model:
sim.createWall (
   name = "left_wall",
   posn = Vec3(1.18700, 0.0000, 0.0000),
   normal = Vec3(1.0000, 0.0000, 0.0000)
)

#add a floor wall to the model:
sim.createWall (
   name = "piso",
   posn = Vec3(0.0000, 0.0000, 0.0000),
   normal = Vec3(0.0000, 1.0000, 0.0000)
)

#add a right side wall to the model:
sim.createWall (
   name = "right_wall",
   posn = Vec3(1.7100, 0.0000, 0.0000),
   normal = Vec3(-1.0000, 0.0000, 0.0000)
)

#add a back wall to the model:
sim.createWall (
   name = "back_wall",
   posn = Vec3(0.0000, 0.0000, -0.0600),
   normal = Vec3(0.0000, 0.0000, 1.0000)
)

#add a front wall to the model:
sim.createWall (
   name = "front_wall",
   posn = Vec3(0.0000, 0.0000, 0.0600),
   normal = Vec3(0.0000, 0.0000, -1.0000)
)

#specify that particles undergo frictional interactions:
sim.createInteractionGroup (
   NRotFrictionPrms (
      name = "friction",
      normalK = 1.0,
      dynamicMu = 0.6,
      shearK = 100.0,
      scaling = True
   )
)

#El comando NRotFrictionPrms especifica tanto la repulsión elástica como la resistencia a
#la fricción entre #partículas. Para la repulsión elástica, es necesario especificar la
#rigidez elástica del material (NormalK) en N/m. En cuanto a la fricción, es necesario
#establecer dos parámetros. El primer parámetro (dynamicMu) es el coeficiente de fricción
#entre las partículas mientras que el segundo parámetro (shearK) es la resistencia al
#corte en el punto de contacto.

#Para la interacción entre las partículas y la pared, se establece sólo la repulsión
```

```
#elástica, puesto que el programa aún no es capaz de simular paredes con fricción. La línea
#de código es muy parecida a la usada para simular la interacción entre partículas:

#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "slopeWall_repell",
      meshName = "slope",
      normalK = k
   )
)

#El parámetro normalK denota la rigidez elástica de la interacción entre la partícula y la
#pared en N/m. Es importante saber que, si la rigidez elástica es muy pequeña, la pared no
#podrá soportar el peso de la partícula y la atravesará; por otro lado, si es demasiado
#grande, la partícula rebotará de forma irreal (Weatherly, Hancock, & Boros, 2014).

#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "01floor_repell",
      meshName = "Wall1",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "02floor_repell",
      meshName = "Wall2",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "03floor_repell",
      meshName = "Wall3",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "04floor_repell",
      meshName = "Wall4",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "05floor_repell",
      meshName = "Wall5",
      normalK = k
```

```
   )
)
#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "06floor_repell",
      meshName = "Wall6",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "07floor_repell",
      meshName = "Wall7",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "08floor_repell",
      meshName = "Wall8",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "09floor_repell",
      meshName = "Wall9",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#with the floor mesh wall:
sim.createInteractionGroup (
   NRotElasticTriMeshPrms (
      name = "10floor_repell",
      meshName = "Wall10",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#with the piso wall:
sim.createInteractionGroup (
   NRotElasticWallPrms (
      name = "piso_repell",
      wallName = "piso",
      normalK = k
   )
)

#specify that particles undergo elastic repulsion
#from the left side wall:
sim.createInteractionGroup (
   NRotElasticWallPrms (
```

```
      name = "lw_repell",
      wallName = "left_wall",
      normalK = k
   )
)
#specify that particles undergo elastic repulsion
#from the right side wall:
sim.createInteractionGroup (
   NRotElasticWallPrms (
      name = "rw_repell",
      wallName = "right_wall",
      normalK = k
   )
)

#specify that particles undergo elastic repulsion
#from the back wall:
sim.createInteractionGroup (
   NRotElasticWallPrms (
      name = "bw_repell",
      wallName = "back_wall",
      normalK = k
   )
)

#specify that particles undergo elastic repulsion
#from the front wall:
sim.createInteractionGroup (
   NRotElasticWallPrms (
      name = "fw_repell",
      wallName = "front_wall",
      normalK = k
   )
)

#specify the direction and magnitude of gravity:
sim.createInteractionGroup (
   GravityPrms (
      name = "gravity",
      acceleration = Vec3(0.0000, -9.8100, 0.0000)
   )
)

#add viscosity to damp particle oscillations:
sim.createInteractionGroup (
   LinDampingPrms (
      name = "viscosity",
      viscosity = 1.7000,
      maxIterations = 100
   )
)
#add a wall loader to move the top wall:
wall_loader1 = WallLoaderRunnable(
LsmMpi = sim,
wallName = "left_wall",
vPlate = Vec3 (-500.0, 0.0, 0.0),
startTime = 20000,
rampTime = 11000
)
sim.addPreTimeStepRunnable (wall_loader1)
```

```
#add a CheckPointer to store simulation data:
sim.createCheckPointer (
   CheckPointPrms (
      fileNamePrefix = "flow_data",
      beginTimeStep = 0,
      endTimeStep = 100000,
      timeStepIncr = 1000
   )
)

#Con esta línea de código, se generan unos archivos de texto que contienen toda la
#información necesaria de #cada partícula tales como el desplazamiento y la velocidad.
#Luego, para poder visualizarlos en el programa Paraview, fue necesario convertirlos en
#archivos vtk. Este proceso se realiza a través de una herramienta de postprocesos de
#ESyS-Particle llamada dump2vtk, la cual es usada en la terminal de Ubuntu. La línea de
#código es la siguiente:

#execute the simulation:
sim.run()
```

## Apéndice B Muros finitos

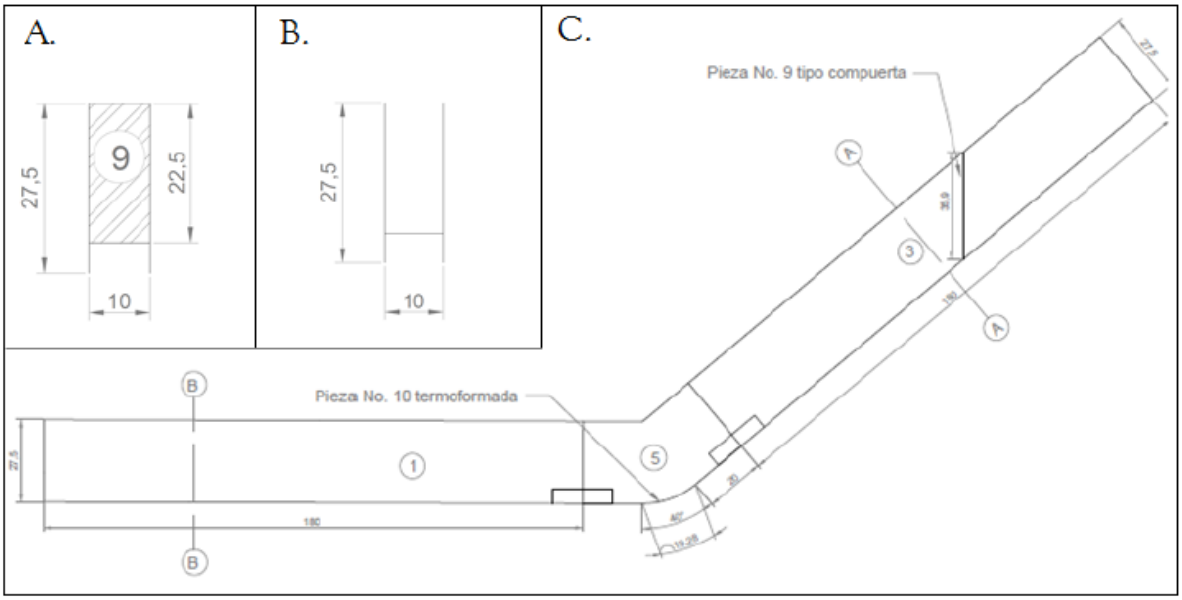
```
Triangle
3D-Nodes 5
 0  0 0  0.000 0.000 -1.0
 1  1 0  0.019 0.001 -1.0
 2  2 0  0.019 0.001  1.0
 3  3 0  0.000 0.000  1.0
 4  4 0  0.0095 0.0005 0.0

Tri3 4
 0  0  0  1  4
 1  0  1  2  4
 2  0  2  3  4
 3  0  0  4  3

#El número de nodos es especificado después de ‘3D-Notes’, mientras que el número después #de
‘Tri3’ especifica cuántos triángulos existen en la pared. Para crear este tipo de #paredes, es
necesario realizarlo en un archivo aparte con extensión ‘msh’, ubicado en la #misma carpeta
del código principal, y leerlo con la siguiente línea de código:

sim.readMesh(
        fileName = "floorMesh.msh",
        meshName = "floor_mesh_wall"
)
```

## Apéndice C Medidas del canal Collaptes V1.0




Montaje general del canal Collaptes V1.0. Vista lateral. (Román, 2017)

## Apéndice D Cómo correr la simulación

1. Abrir el terminal y buscar la carpeta en donde se encuentra guardado el archivo de simulación .py, así como los archivos .smh.

Ejemplo: cd Desktop/simulación

2. Correr la simulación por medio de la siguiente línea de código en la terminal:

mpirun -np 2 esysparticle Deslizamiento.py

3. Esperar a que se generen los archivos .txt del anterior paso y luego ingresar el siguiente comando:

dump2vtk -i "flow_data" -o vtk_snaps_ -t 0 101 1000

Las palabras escritas entre comillas deben ser las mismas con las que se nombró en el CheckPointer, mientras que los números corresponden al incio de la toma de datos, el número de archivos que debe generar y el incremento del paso, respectivamente.

4. Abrir el archivo .vtu generado, por medio del visualizador Paraview.

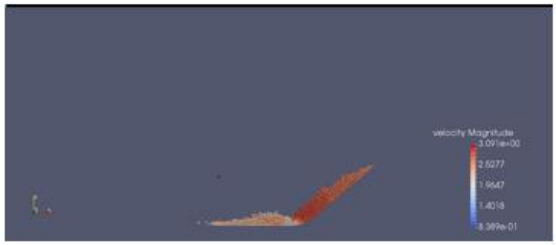
## Apéndice E Comparación entre el modelo Collaptes V1.0 realizado por el grupo INME y la simulación



Segundo 0.0



Segundo 0.4


Segundo 0.9



Segundo 1.3