



Universidad de Las Palmas de Gran Canaria

Escuela de Ingenierías Industriales y Civiles

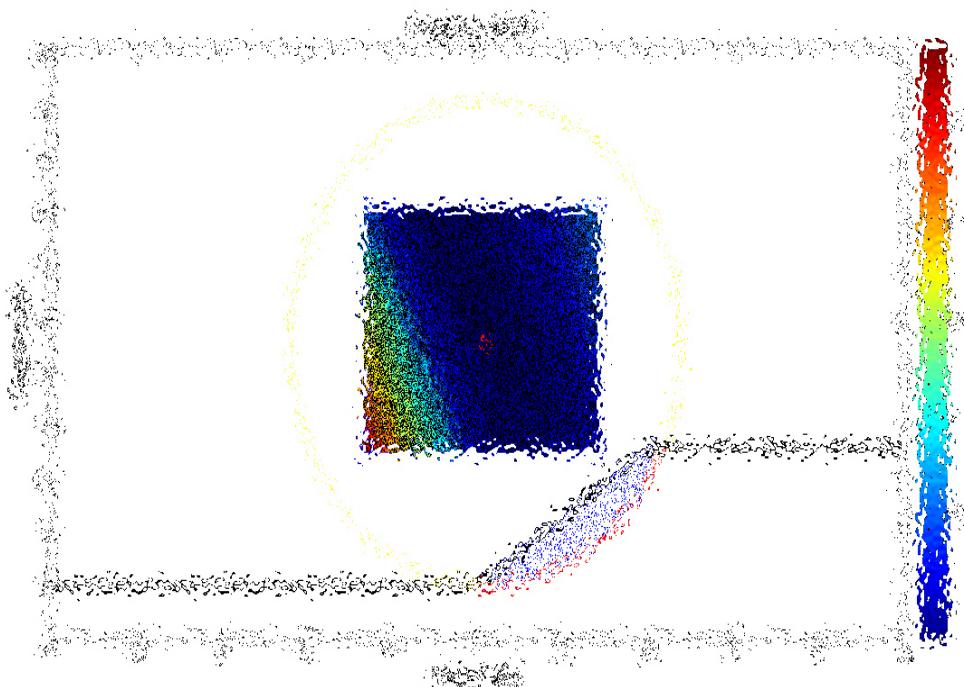
---

---

# Programa para el cálculo de estabilidad de taludes mediante el método de Morgenstern-Price

Ingeniero Industrial (Plan 2001)

---



---

Tutores: Francisco Chirino Godoy  
David Greiner Sánchez

Autor: Sergio Marrero Marrero

---

Las Palmas, Julio de 2015



---

**Programa para el cálculo de estabilidad de taludes  
mediante el método de Morgenstern-Price**

---

---

# **Programa para el cálculo de estabilidad de taludes mediante el método de Morgenstern-Price**

---

## **AUTOR:**

**Sergio Marrero Marrero**

## **RESUMEN DEL PROYECTO**

Se ha desarrollado un programa de cálculo de estabilidad de taludes de tierra, el cual permite calcular el factor de seguridad que tiene un determinado talud. Los métodos de cálculo que se han usado para tal fin han sido el método de Fellenius, el método de Bishop simplificado y finalmente el método de Morgenstern-Price. Debido a los largos tiempos de simulación se desarrollaron métodos de optimización haciendo el programa más eficiente. Finalmente se comparan los resultados de este programa con otros programas de referencia, concluyéndose que los resultados son buenos y quedando justificado el uso del programa para resolver casos reales.

## ÍNDICE DE CONTENIDOS

<b><u>1. INTRODUCCIÓN</u></b>	<b>14</b>
<b><u>2. MARCO TEÓRICO</u></b>	<b>17</b>
<b>2.1 TALUD, DESLIZAMIENTO Y SUPERFICIE DE ROTURA</b>	<b>18</b>
<b>2.2 MÉTODOS DE EQUILIBRIO LÍMITE.</b>	<b>21</b>
2.2.1 INTRODUCCIÓN	21
2.2.2 MÉTODOS DE REBANADAS	23
<b><u>3. IMPLEMENTACIÓN DE LOS MÉTODOS DE EQUILIBRIO LÍMITE</u></b>	<b>40</b>
<b>3.1 SUPERFICIE DEL TALUD</b>	<b>40</b>
<b>3.2 SUPERFICIE DE ROTURA</b>	<b>41</b>
3.2.1 INTERSECCIÓN ENTRE UNA RECTA Y UNA CIRCUNFERENCIA EN EL PLANO	42
3.2.2 INTERSECCIÓN ENTRE TRES RECTAS Y UNA CIRCUNFERENCIA EN EL PLANO	44
3.2.3 SELECCIÓN DE LOS PUNTOS DE CORTE	45
<b>3.3 DIVISIÓN DE LA SUPERFICIE DE ROTURA EN REBANADAS</b>	<b>53</b>
3.3.1 CÁLCULO DE REBANADAS	55
3.3.2 CÁLCULO DE PARÁMETROS	59
<b>3.4 VARIABLES ASOCIADAS A CADA REBANADA</b>	<b>61</b>
3.4.1 ORDENADAS DE LAS ESQUINAS SUPERIORES DE CADA REBANADA	63
3.4.2 ORDENADAS DEL SEMICÍRCULO	63
<b>3.5 MÉTODO DE FELLENIUS</b>	<b>64</b>
<b>3.6 MÉTODO DE BISHOP SIMPLIFICADO</b>	<b>66</b>
<b>3.7 MÉTODO DE MORGENSTERN-PRICE</b>	<b>68</b>
<b><u>4. IMPLEMENTACIÓN DE LOS ALGORITMOS DE BÚSQUEDA</u></b>	<b>78</b>
<b>4.1 FACTOR DE SEGURIDAD EN UNA SUPERFICIE DE DESLIZAMIENTO</b>	<b>78</b>
4.1.1 VARIABLES NECESARIAS	78
<b>4.2 FACTOR DE SEGURIDAD VARIANDO EL RADIO SOBRE UN PUNTO</b>	<b>79</b>
4.2.1 INTRODUCCIÓN	79
4.2.2 RADIO MÍNIMO	83

4.2.3 RADIO MÁXIMO PARA CENTROS MENORES A LA ALTURA DEL TALUD	87
4.2.4 IMPLEMENTACIÓN ALGORÍTMICA	88
<b>4.3 FACTOR DE SEGURIDAD EN UN MARCO DE BÚSQUEDA</b>	<b>90</b>
 <b>5. OPTIMIZACIÓN DE LOS ALGORITMOS DE BÚSQUEDA</b>	<b>92</b>
 5.1 OPTIMIZACIÓN DE LA BÚSQUEDA DEL FS AMPLIANDO EL RADIO EN UN PUNTO	92
5.1.1 CURVAS TÍPICAS $FS(R)$ .	92
5.1.2 ALGORITMO DE OPTIMIZACIÓN	94
5.1.3 ALGORITMO FINAL	97
5.1.4 CONCLUSIÓN	101
<b>5.2 OPTIMIZACIÓN DE LA BÚSQUEDA DEL FS EN UN MARCO</b>	<b>103</b>
5.2.1 CURVAS TÍPICAS $FS(A,B)$	103
5.2.2 ALGORITMO DE OPTIMIZACIÓN	105
<b>5.3 CONCLUSIÓN</b>	<b>121</b>
 <b>6. COMPARACIÓN DE RESULTADOS CON PROGRAMAS DE REFERENCIA</b>	<b>123</b>
 6.1 FACTOR DE SEGURIDAD PARA UNA SUPERFICIE DE ROTURA DETERMINADA	123
<b>6.2 FACTOR DE SEGURIDAD MÍNIMO GLOBAL</b>	<b>125</b>
 <b>7. RESOLUCIÓN DE UN CASO PRÁCTICO</b>	<b>130</b>
 <b>8. CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO</b>	<b>136</b>
8.1 CONCLUSIÓN	136
8.2 LÍNEAS FUTURAS	137
 <b>9. BIBLIOGRAFÍA</b>	<b>138</b>
 <b>10. ANEXO</b>	<b>139</b>

**ÍNDICE DE ILUSTRACIONES**

Ilustración 2-1: Talud típico de tierra (Perez, 2005).....	18
Ilustración 2-2: Talud, deslizamiento y superficie de rotura.....	19
Ilustración 2-3: Taludes artificiales (Ramirez & Alejano, 2009) .....	20
Ilustración 2-4: Talud artificial: en terraplen y en desmonte.....	20
Ilustración 2-5: Talud y variables implicadas I.....	24
Ilustración 2-6: Talud y variables implicadas II.....	25
Ilustración 2-7: Rebanadas y esfuerzos implicados.....	25
Ilustración 2-8: Polígono de fuerzas general.....	26
Ilustración 2-9: Polígono de fuerzas. Método de Fellenius.....	29
Ilustración 2-10: Equilibrio de fuerzas. Método de Fellenius .....	29
Ilustración 2-11: Polígono de fuerzas. Método de Bishop simplificado.....	31
Ilustración 2-12: Equilibrio de fuerzas verticales. Método de Bishop simplificado .....	31
Ilustración 2-13: Funcion $f(x)$ a lo largo del talud .....	34
Ilustración 2-14: Funciones $f(x)$ típicas.....	34
Ilustración 2-15: Diagrama de flujo Morgenstern-Price .....	39
Ilustración 3-1: Talud y rectas $k_1, k_2, k_3$ .....	41
Ilustración 3-2: Circunferencias en el plano.....	42
Ilustración 3-3: Definición de una circunferencia .....	42
Ilustración 3-4: Seis puntos de corte entre la circunferencia y las tres rectas .....	45
Ilustración 3-5: Reglas R1 y R2 .....	46
Ilustración 3-6: Regla R4. Eliminación de puntos con corte mayor al centro .....	48
Ilustración 3-7: Intersección de la matriz <i>puntosposibles</i> con la superficie del talud .....	49
Ilustración 3-8: Circunferencia con doble entrada en el talud.....	50
Ilustración 3-9: Puntos extremos pti,ptd .....	50
Ilustración 3-10: Puntos extremos a la misma altura .....	51
Ilustración 3-11: División en rebanadas general .....	53
Ilustración 3-12: División en rebanadas forzando los vértices.....	54
Ilustración 3-13: Variables implicadas en la división en rebanadas.....	56
Ilustración 3-14: Casos posibles en la división en rebanadas .....	56
Ilustración 3-15: Parámetros I .....	62
Ilustración 3-16: Parámetros II .....	62
Ilustración 3-17: Familia de funciones half-sine.....	71
Ilustración 4-1: Distintos FS variando el radio.....	80

Ilustración 4-2: Resultados de la simulación 4.1 .....	81
Ilustración 4-3: Resultados de la simulación 4.2 .....	82
Ilustración 4-4: Cálculo del radio mínimo I .....	83
Ilustración 4-5: Cálculo del radio mínimo II .....	86
Ilustración 4-6: Cálculo del radio máximo cuando $b < H$ .....	87
Ilustración 4-7: Definición de márco de búsqueda .....	90
Ilustración 4-8: Pseudocódigo de la función Rastreo.....	91
Ilustración 5-1: Resultados de la simulación 5.1 .....	93
Ilustración 5-2: Resultados de la simulación 5.2 .....	93
Ilustración 5-3 : Cálculo del radio máximo cuando $b > H$ .....	94
Ilustración 5-4: Etapas al aumentar el radio .....	96
Ilustración 5-5: Asignación de variables $I_i$ .....	96
Ilustración 5-6: Ejemplo desimulación usando la función FSRMinConOpt.....	98
Ilustración 5-7: Comparación de la función FSRMinConOpt y FSRMinSinOpt .....	99
Ilustración 5-8: Segunda etapa del algoritmo de optimiación.....	100
Ilustración 5-9: Resultados de la simulación 5.3 .....	102
Ilustración 5-10: Marco inicial de la simulación 5.4 .....	103
Ilustración 5-11: Marco con distribución del FS y FS crítico.....	104
Ilustración 5-12: Distribución 3D del FS .....	104
Ilustración 5-13: Zoom de la distrución del FS en 3D.....	105
Ilustración 5-14:Marco con variables de optimización .....	106
Ilustración 5-15: Marco consecutivo mínimo .....	107
Ilustración 5-16: Resultados de la simulación 5.6 .....	119
Ilustración 5-17: Resultado de la simulación 5.7 .....	120
Ilustración 5-18: Resultados de la simulación 5.8 .....	121
Ilustración 7-1: Resultados del caso práctico. Método de Fellenius.....	133
Ilustración 7-2: Resultado del caso práctico. Método de Bishop simplificado .....	133
Ilustración 7-3: Resultado del caso práctico. Método de Morgenstern-Price: .....	134
Ilustración 7-4: Resultado del caso práctico. Comparación de los métodos .....	134

**ÍNDICE DE TABLAS**

Tabla 2-1: Métodos de cálculo de estabilidad de taludes.....	22
Tabla 2-2: Número de incognitas totales .....	27
Tabla 2-3: Número de ecuaciones disponibles .....	28
Tabla 2-4: Hipótesis añadidas en el método de Fellenius.....	30
Tabla 2-5: Hipótesis en el método de Bishop simplificado .....	33
Tabla 2-6: Hipótesis en el método de Morgenstern-Price .....	35
Tabla 2-7: Incognita adicional en el método de Morgenstern-Price.....	35
Tabla 2-8: Adaptación al método de Morgenstern-Price.....	39
Tabla 3-1: Características de las rectas $k_1, k_2, k_3$ .....	45
Tabla 4-1: Valores de la simulación 4.1.....	81
Tabla 4-2: Valores de la simulación 4.2.....	82
Tabla 5-1: Valores de la simulación 5.1.....	92
Tabla 5-2: Valores de la simulación 5.2.....	93
Tabla 5-3: Valores de la simulación 5.3.....	101
Tabla 5-4: Resultados de la simulación 5.3 .....	101
Tabla 5-5: Valores de la simulación 5.4.....	103
Tabla 5-6: Resultados de la optimización.....	115
Tabla 5-7: Valores de entrada y resultados de la simulación 5.6.....	118
Tabla 5-8: Valores iniciales y resultados de la simulación 5.7 .....	119
Tabla 5-9: Valores iniciales y resultados de la simulación 5.8 .....	120
Tabla 5-10: Valores iniciales de la simulación 5.8.....	122
Tabla 5-11: Resultados sin optimizar. Simulación 5.8.....	122
Tabla 5-12: Resultados optimizando. Simulación 5.8 .....	122
Tabla 6-1: Características de taludes de referencia .....	123
Tabla 6-2: Puntos y radios para las simulaciones con el talud 1 .....	124
Tabla 6-3: Puntos y radios para las simulaciones con el talud 3 .....	124
Tabla 6-4: Resultados de la comparación del FS para una superficie de rotura .....	125
Tabla 6-5: Valores iniciales I .....	125
Tabla 6-6: Valores iniciales II .....	126
Tabla 6-7: Valores iniciales III .....	126
Tabla 6-8: FS crítico con talud 1 .....	126
Tabla 6-9: FS crítico con talud 2 .....	127
Tabla 6-10: FS crítico con talud 3 .....	127

Tabla 6-11: FS crítico con talud 4 .....	128
Tabla 6-12: FS crítico con talud 5 .....	128
Tabla 6-13: FS crítico con talud 6 .....	129
Tabla 6-14: FS crítico con talud 7 .....	129
Tabla 7-1: Características del talud de un caso práctico.....	130
Tabla 7-2: Valores iniciales.....	130
Tabla 7-3: Caso práctico; H=10.....	131
Tabla 7-4: Caso práctico; H=20.....	131
Tabla 7-5: Caso práctico; H=30.....	131
Tabla 7-6: Caso práctico; H=40.....	131
Tabla 7-7: Caso práctico; H=50.....	131
Tabla 7-8: Caso práctico; H=60.....	132
Tabla 7-9: Caso práctico; H=70.....	132
Tabla 7-10: Caso práctico; H=80.....	132
Tabla 7-11: Caso práctico; H=90.....	132
Tabla 7-12: Caso práctico; H=100.....	132
Tabla 8-1: Comparación de tiempos y simulaciones.....	136

**ÍNDICE DE ECUACIONES**

Ecuación 2-1 .....	27
Ecuación 2-2 .....	28
Ecuación 2-3 .....	30
Ecuación 2-4 .....	32
Ecuación 2-5 .....	33
Ecuación 2-6 .....	37
cuación 2-7 .....	37
Ecuación 2-8 .....	37
Ecuación 2-9 .....	37
Ecuación 2-10 .....	37
Ecuación 2-11 .....	37
Ecuación 2-12 .....	37
Ecuación 2-13 .....	37
Ecuación 2-14 .....	38
Ecuación 2-15 .....	38
Ecuación 3-1 .....	40
Ecuación 3-2 .....	43
Ecuación 3-3 .....	43
Ecuación 3-4 .....	43
Ecuación 3-5 .....	51
Ecuación 3-6 .....	65
Ecuación 3-7 .....	67
Ecuación 5-1 .....	109
Ecuación 5-2 .....	109
Ecuación 5-3 .....	109
Ecuación 5-4 .....	110
Ecuación 5-5 .....	112
Ecuación 5-6 .....	112
Ecuación 5-7 .....	113
Ecuación 5-8 .....	114
Ecuación 5-9 .....	116
Ecuación 5-10 .....	116
Ecuación 5-11 .....	116

**ÍNDICE DE ALGORITMOS**

Algoritmo 3-1: Función <i>taludgeometria</i> .....	41
Algoritmo 3-2: Función <i>raicircunferecta</i> .....	44
Algoritmo 3-3: Función <i>intersectaludcirc</i> .....	47
Algoritmo 3-4: Función <i>calculoextremos</i> .....	52
Algoritmo 3-5: Subproceso de la función <i>divisorderebanadas</i> .....	57
Algoritmo 3-6: Subproceso de la función <i>divisorderebanadas</i> .....	57
Algoritmo 3-7: Subproceso de la función <i>divisorderebanadas</i> .....	58
Algoritmo 3-8: Subprocesos de la función <i>divisorderebanadas</i> .....	58
Algoritmo 3-9: Subproceso de la función <i>divisorderebanadas</i> .....	60
Algoritmo 3-10: Subproceso de función <i>divisorderebanadas</i> .....	61
Algoritmo 3-11: Pseudocódigo de la función <i>divisorderebanadas</i> .....	61
Algoritmo 3-12: Funcion mFelle.....	66
Algoritmo 3-13: Función mBishop.....	68
Algoritmo 3-14: Subproceso de la función mMorgPri .....	70
Algoritmo 3-15: Subprocesos de la función mMorgPri.....	72
Algoritmo 3-16: Subproceso de la función ZhuLeeChen.....	74
Algoritmo 3-17: Subproceso de la función ZhuLeeChen .....	75
Algoritmo 3-18: Subprocesos de la función ZhuLeeChen .....	76
Algoritmo 4-1: Función FSP .....	79
Algoritmo 4-2: Primera propuesta para la función FSRMinSinOpt.....	80
Algoritmo 4-3: Función distminR1 .....	88
Algoritmo 4-4: Pseudocodigo de la función FSRMinSinOpt.....	89
Algoritmo 5-1: Pseudocódigo del subproceso Radiomaximo .....	95
Algoritmo 5-2: Subproceso de la función FSRMinConOpt .....	97
Algoritmo 5-3: Subproceso de la función FSRMinConOpt .....	100
Algoritmo 5-4: Pseudocódigo de la función FSRMinConOpt .....	101
Algoritmo 5-5: Función marcosoptimos.....	117



# **MEMORIA DESCRIPTIVA**

## 1. Introducción

---

En cualquier obra de ingeniería –normalmente civil o minera- en la que se requiera hacer alguna actuación sobre el suelo, es habitual que surja la necesidad de realizar desmontes o terraplenes, quedando en ambos casos la superficie del terreno inclinada un ángulo respecto de la horizontal. Dependiendo de las características del terreno y de otras condiciones como el nivel torrencial, de vegetación o sísmico, estas estructuras pueden ser construidas de forma peligrosa. Se debe hacer un estudio sobre cuál debería ser la inclinación apropiada para que la masa de suelo inclinada no siga adelante con la tendencia inherente a deslizar sobre sí misma. La teoría de estabilización de taludes es la disciplina que con su esfuerzo y desarrollo arroja luz sobre la forma de llevar a cabo este tipo de construcciones.

Por otro lado, no hay que olvidar las implicaciones económicas que tiene determinar apropiadamente el ángulo de talud. Cualquier desplazamiento de tierra de condiciones más o menos importantes, viene asociado a un gran esfuerzo humano en el que se ven implicadas grandes máquinas, sin las cuales sería imposible realizar los enormes desmontes o terraplenes que se practican hoy en día. Para quien desembolsa el dinero, lo ideal sería que el talud formara el mayor ángulo posible respecto de la horizontal, de esta forma se conseguiría el menor desplazamiento de tierra posible y por ende, menor repercusión económica. Esto deja de ser cierto cuando se tienen en cuenta los daños que podría ocasionar un deslizamiento de tierra por culpa de una inclinación demasiado pronunciada, los cuales no sólo serían caros sino que también podrían ser irreparables, si se piensa por ejemplo en las posibles víctimas que podrían estar asociadas a dicho desprendimiento.

El diseño de grandes taludes en el ámbito de la ingeniería civil y minera resulta cada día más común. El ingeniero se ve ante la tesitura de hacer un equilibrio entre seguridad y economía. Ciertamente, aumentando la inclinación de los taludes se pueden reducir los costes al disminuir el volumen del material a excavar, pero la inclinación de un talud debe tener como límite la seguridad del mismo, ya que todo el ahorro conseguido puede perderse, poniéndose además en riesgo la seguridad de las personas o cosas que encuentre en o bajo el talud. (Hustruill et al.,2000).

El problema de la estabilidad de taludes ha merecido a lo largo de la historia la atención de profesionales que gracias al esfuerzo que han realizado se han podido diseñar de forma segura.

El cálculo de la estabilidad de taludes -como muchas otras disciplinas- experimentó un gran avance con el desarrollo computacional, pudiéndose abandonar los viejos ábacos a cambio de nuevos modelos numéricos antes imposibles de resolver. Actualmente el cálculo de estabilidad de taludes se realiza a través de programas informáticos comerciales, sin embargo estos programas pueden no ser baratos y por esta razón algunos profesionales aún utilizan las viejas formas para su cálculo.

Por otro lado, la teoría de estabilidad de taludes ha seguido evolucionando en los últimos cincuenta años. Tanto es así que el método de Morgenstern-Price, cuya implementación algorítmica es el objetivo principal de este proyecto fue publicado en el año 1965. Y no solo esto, la adaptación que se va a seguir para poder implementar el código de forma eficiente se publicó en el año 2005. (Zhu, Lee, Qian, & Chen, 2005)

En este proyecto se desarrolla un programa informático que permite el cálculo de estabilidad de taludes a través de tres métodos clásicos de esta disciplina: El método de Fellenius, el método de Bishop simplificado y el método de Morgenstern-Price. Esta memoria se desarrolla como se viene a explicar a continuación.

El capítulo dos está dedicado a desarrollar un marco teórico que sirva de introducción a la problemática en general, recorriendo así los distintos métodos que se han estudiado y utilizado para dar con una solución aceptable al problema de la estabilidad de taludes. Se abordarán los métodos de equilibrio límite, recalmando que el más efectivo es el método de Morgenstern-Price.

Una vez se ha desarrollado el marco teórico se puede comenzar a implementar algorítmicamente los métodos. Se consideró propicio para llevar esta tarea, utilizar la herramienta de software matemático Matlab, con lo cual el lenguaje de programación que se ha utilizado es el propio de este entorno de desarrollo (*lenguaje M*)

El capítulo tres contiene todos los engranajes y piezas que permiten a los procesos que se desarrollan en capítulos posteriores realizar operaciones complejas. En resumen, en este capítulo se explican todas las funciones y procesos que permiten calcular los métodos de estabilidad de taludes. El capítulo cuatro permite orientar en un sentido u otro las distintas funciones que aparecen en el capítulo anterior. En este se explican los métodos que permiten

mover las distintas superficies de rotura a lo largo y ancho de un plano. El capítulo cinco se ha dedicado a la optimización, en cierto modo permite que las funciones del capítulo cuatro manejen a las funciones del capítulo tres con mayor velocidad.

Una vez se ha verificado que los métodos funcionan, estos deben ser testeados y comparados. El capítulo seis está dedicado a comprobar si coinciden los resultados del programa desarrollado con los resultados obtenidos en otros programas de referencia. Una vez se ha verificado que el programa funciona correctamente se da paso en el capítulo siete a la resolución de un caso práctico. Se deja el capítulo ocho para dar una serie de conclusiones sobre el trabajo realizado y además, se dejarán planteadas cuales deberían de ser las líneas futuras de investigación y desarrollo sin salir del ámbito que se desarrolla en este proyecto.

Por otro lado, haciendo referencia al estilo de redacción utilizado, hay que tener en cuenta lo siguiente. Se quiso evitar en todo momento que la densidad de las explicaciones - lejos de satisfacer el objetivo de ayudar a comprender el código - pudiera integrar en sí mismo una dificultad adicional. Por esta razón se sustituyó la explicación detallada de algunos códigos por una explicación orientada a que el lector pueda intuir el paradigma que se deseaba transmitir. Por ello, en algunas ocasiones se hace referencia al código original y en otras ocasiones se hace referencia a pseudocódigos, que sin duda alguna son más fáciles de entender.

Es posible que las distintas variables que aparecen a lo largo de la memoria puedan representar entidades diferentes en distintos capítulos. Esto es debido a que a veces se hace referencia a ellas desde el punto de vista matemático, otras veces desde el punto de vista ingenieril y otras veces desde el punto de vista del nombre que recibe en el código fuente. Sin embargo, se ha puesto empeño en que las explicaciones sean lo suficientemente claras para que no suponga ningún problema.

## 2. Marco teórico

---

Para comenzar se debe aclarar que los métodos que se desarrollan en este proyecto usan los modelos desarrollados en la disciplina de mecánica de suelos. El término suelo es bastante utilizado en el lenguaje común y en el ámbito técnico-científico adquiere un significado u otro dependiendo de la disciplina en la que sea tratado. Con ánimo de reducir la ambigüedad, pasan a citarse dos definiciones de suelo.

*El término suelo puede tener diferentes matices de significados dependiendo del contexto en el que se use. Para el geólogo, "suelo describe las capas de material suelto sin consolidar que se extienden desde la superficie hasta la roca sólida, y que se han formado por el intemperismo y la desintegración de las propias rocas. Por otra parte, para el ingeniero, el concepto de suelo está relacionado con la obra que pueda hacer sobre él, con él o en él. Por consiguiente, para la ingeniería el término suelo se refiere al material que se puede utilizar sin necesidad de perforaciones o voladuras.*

(Whitlow, 1994)

*Desde el punto de vista de la Geotécnica, los materiales de la corteza terrestre se dividen en dos grandes grupos:*

- ✓ *Rocas: Materiales duros que sólo se pueden fragmentar con grandes esfuerzos mecánicos (sílice, caliza, feldespato, granito, pizarra) y cuyo estudio es el objeto de la Mecánica de Rocas.*
- ✓ *Suelos: Conjunto de agregados minerales que pueden segregarse en elementos de dimensiones más o menos grandes, sin que haga falta para ello aplicar un gran esfuerzo mecánico y que provienen de procesos de alteración de las rocas, de tipo químico, físico o mecánico. Un terreno estará formado en general por diferentes tipos de suelos.*

(Iglesias, 1997)

## 2.1 Talud, deslizamiento y superficie de rotura

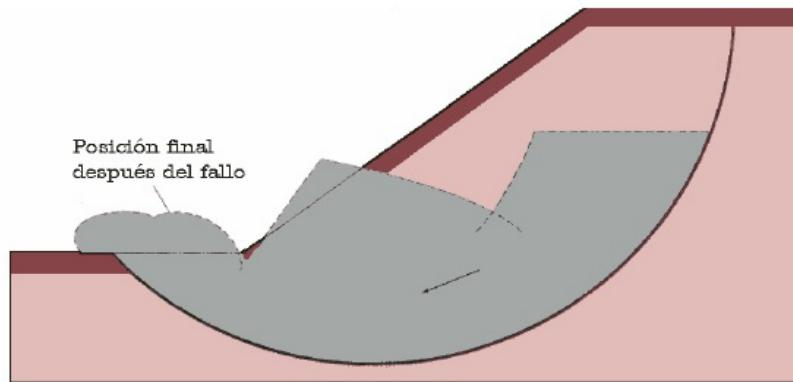
---

El concepto de *talud* hace referencia a una disposición inclinada del suelo. Cuando se dan las condiciones oportunas se produce un movimiento de talud, con los problemas que esto acarrea. El término más comúnmente usado para designar los movimientos producidos en los taludes es el *deslizamiento* a través de una *superficie de rotura*. (AYALA CARCEDO & ANDREU POSSE, 1987)

Ilustración 2-1: Talud típico de tierra (Perez, 2005)



En la *Imagen 2-1* se aprecia un talud de tierra típico. Por otro lado en la Ilustración 2-2 se aprecian los tres conceptos que se acaban de explicar. Por un lado se aprecia el talud, por otro lado se aprecia la posición final de la masa de suelo deslizada, y por otro lado se aprecia la superficie de rotura sobre la que deslizó dicha masa de suelo.

**Ilustración 2-2: Talud, deslizamiento y superficie de rotura**

De forma general, los deslizamientos de taludes, se inician debido a variaciones de condición, como por ejemplo, un cambio en las condiciones de humedad del suelo, drenado carga o estabilidad superficial (por ejemplo por eliminación de vegetación). Estos cambios pueden presentarse inmediatamente después de la construcción, o se pueden desarrollar con lentitud a lo largo de algunos años, o bien pueden presentarse de repente. En el análisis de taludes tanto cortados como construidos es necesario tener en cuenta las condiciones de estabilidad tanto inmediatas como a largo plazo.

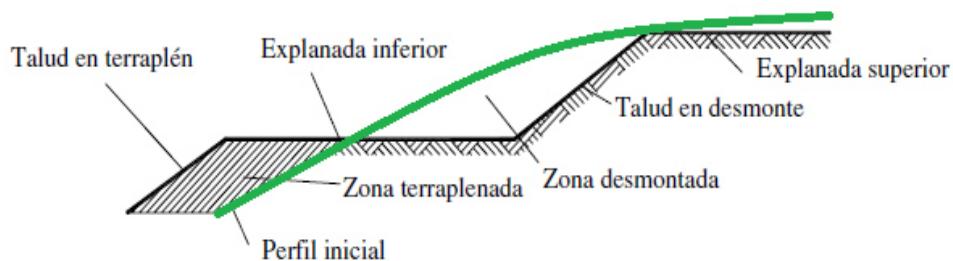
La mecánica de suelos distingue a priori dos tipos de taludes, los taludes de origen natural y los taludes artificiales. Los primeros suelen ser de dimensiones bastante grandes, de tal forma que el costo de la solución sería demasiado grande, siendo más rentable buscar una reubicación o traslado en la obra. Los taludes artificiales son bastante comunes en la obra civil o minera, constituyendo un elemento constructivo bastante habitual (ver Ilustración 2-3)

Ilustración 2-3: Taludes artificiales (Ramirez & Alejano, 2009)



Al perfil conseguido tras una excavación se le denomina *talud en desmonte* y al perfil conseguido tras depositar una masa de tierra sobre el suelo, *talud en terraplén*. (ver Ilustración 2-4)

Ilustración 2-4: Talud artificial: en terraplén y en desmonte



Los métodos que se explican en este proyecto están enfocados a los taludes artificiales, que son el principal problema de ingeniería que se quiere resolver. Sin embargo, estos métodos también pueden aplicarse a problemas de taludes naturales, siempre y cuando se hagan las hipótesis acerca de las características del suelo lo más cercanas a la realidad.

## 2.2 Métodos de equilibrio límite.

---

### 2.2.1 Introducción

El objetivo principal de estudiar y dedicar esfuerzo sobre los taludes de tierra, es el de buscar el grado de seguridad que tiene o tendría -en caso de construirse- un determinado talud. Este grado de seguridad podría indicarse de diferentes formas y una de ellas, que es la que usan los métodos de equilibrio límite que se desarrollan en este proyecto, es a través del *factor de seguridad*.

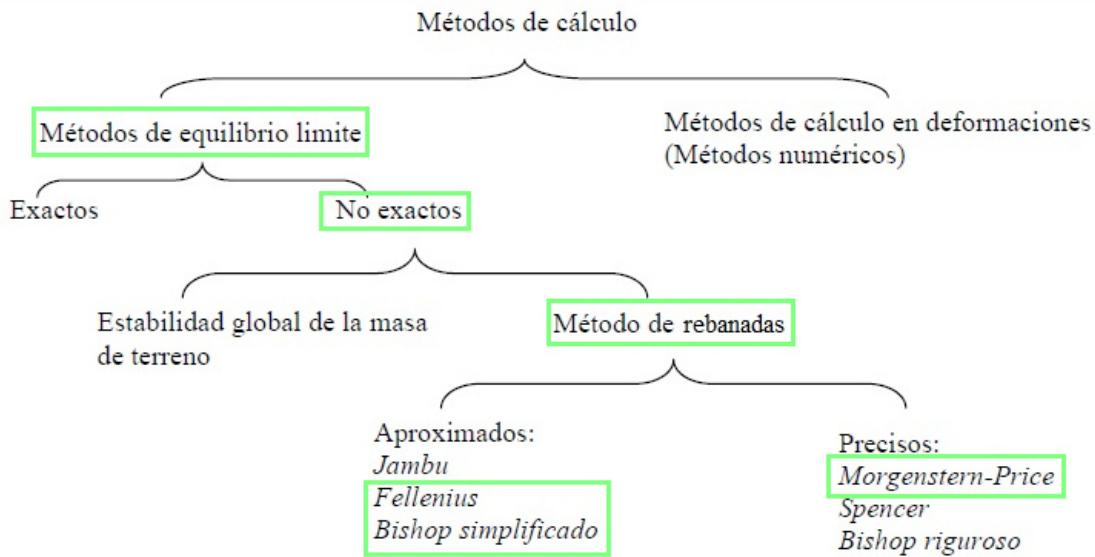
El *factor de seguridad* relaciona a través de un cociente, a las fuerzas o pares que tratan de desestabilizar la masa de suelo inclinada, con las fuerzas o pares que impiden que estas acciones desestabilizadoras puedan provocar un deslizamiento. De forma simplificada, se puede escribir este *factor de seguridad* como sigue

$$FS = \frac{\text{Acciones estabilizadoras}}{\text{Acciones desestabilizadoras}}$$

Si las acciones estabilizadoras son mayores a las desestabilizadoras, entonces el *factor de seguridad* será mayor a uno, y se podrá decir que ese talud tiene un grado de seguridad suficiente para que no se produzca un deslizamiento en las condiciones en las que está. En caso contrario, las acciones desestabilizadoras harán deslizar a la masa de suelo. Si el *factor de seguridad* es igual a la unidad, se dice que ese talud está en una situación de *falla incipiente*.

Existen diversos métodos que intentan resolver este problema. En la Tabla 2-1 se observa un esquema que muestra las distintas formas de cálculo. Se ha marcado con un recuadro los grupos a los que pertenecen los métodos que se desarrollan en este texto.

Tabla 2-1: Métodos de cálculo de estabilidad de taludes



Los *métodos de cálculo en deformaciones* consideran el cálculo las deformaciones del terreno además de las leyes de la estática. Su aplicación práctica es de gran complejidad y el problema debe estudiarse aplicando *el método de los elementos finitos*.

En los *métodos de equilibrio límite* –que son los que se desarrollan en este texto, en concreto los métodos de rebanadas - se hace uso exclusivamente de las leyes de la estática para determinar el estado de equilibrio de una masa de terreno potencialmente inestable, sin tener en cuenta las deformaciones del terreno.

Como se verá más adelante, para desarrollar los métodos de equilibrio límite es necesario hacer una hipótesis sobre el tipo de superficie de rotura. Si esta superficie de rotura, es lo suficientemente sencilla -por ejemplo, si se escoge como superficie de rotura una cuña o una superficie plana- las leyes de la estática proporcionan una solución exacta del problema. Estos métodos entran dentro del grupo de *métodos exacto*. Sin embargo, en la mayor parte de los casos, la geometría de la superficie de rotura no permite obtener una solución exacta del problema mediante la única aplicación de las ecuaciones de la estática. Esto es debido a que el problema es hiperestático y ha de hacerse alguna simplificación o hipótesis previa que permita su resolución. Con lo cual, cuando la hipótesis que se hace sobre la superficie de rotura, no permite obtener una solución exacta del problema, se denominan métodos de equilibrio límite no exactos.

Siguiendo la línea de descenso del esquema mostrado en la Tabla 2-1 - por el camino de los métodos no exactos- aparecen los siguientes métodos. Los métodos de *equilibrio global* de la masa deslizante. En estos métodos se trata la masa potencialmente deslizante como un único sólido rígido. La hiperestaticidad de las ecuaciones se resuelve normalmente introduciendo una hipótesis previa respecto a la distribución de tensiones normales en la superficie potencial de deslizamiento. Debido a que los métodos de rebanadas ofrecen resultados mucho mejores han quedado en desuso. Los *métodos de rebanadas* son los que más difusión han tenido. Esto es debido a que ofrecen resultados más que aceptables y sobre todo porque gracias al desarrollo del cálculo computacional, son relativamente sencillos de implementar. Básicamente consisten en dividir la masa potencialmente deslizante en un número determinado de rebanadas, cada rebanada se trata de forma aislada y se somete a las ecuaciones de equilibrio estático. Como el sistema de ecuaciones resultante está indeterminado habrá que añadir hipótesis para determinarlo. Estas hipótesis se hacen sobre los esfuerzos laterales que se hacen unas rebanadas a otras. Las distintas hipótesis que se hagan sobre estos esfuerzos, darán nombre a los distintos métodos.

Los *métodos aproximados* introducen hipótesis que no cumplen todas las ecuaciones de equilibrio. Como ejemplos de estos métodos se tiene el *método de Fellenius* y el *método de Bishop*. Cuando las hipótesis que se hacen para eliminar la hiperestaticidad sí cumplen con todas las ecuaciones de la estática, se tiene los *métodos precisos*. Dentro de este tipo de métodos se tiene el *método de Morgenstern-Price*, siendo su implementación informática el principal objetivo que se persigue en este proyecto.

## 2.2.2 Métodos de rebanadas

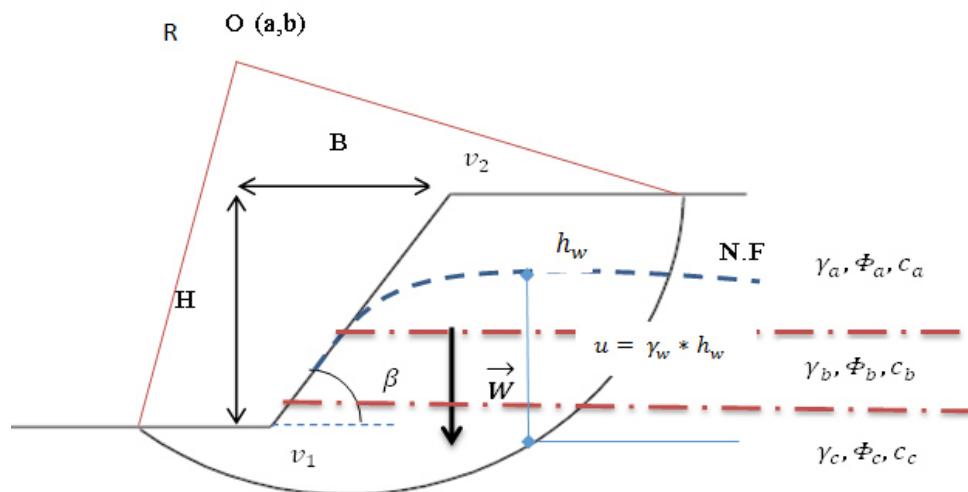
---

En el apartado anterior se intentó enmarcar y contextualizar los métodos que se desarrollan en esta memoria, haciendo un breve repaso de las distintas formas de cálculo que existen. En este apartado se desarrolla el método de rebanadas de forma general, dejando así planteado el sistema de ecuaciones indeterminado que resolverán los distintos métodos propuestos por Fellenius, Bishop y Morgenstern-Price.

### 2.2.2.1 Método de rebanadas general

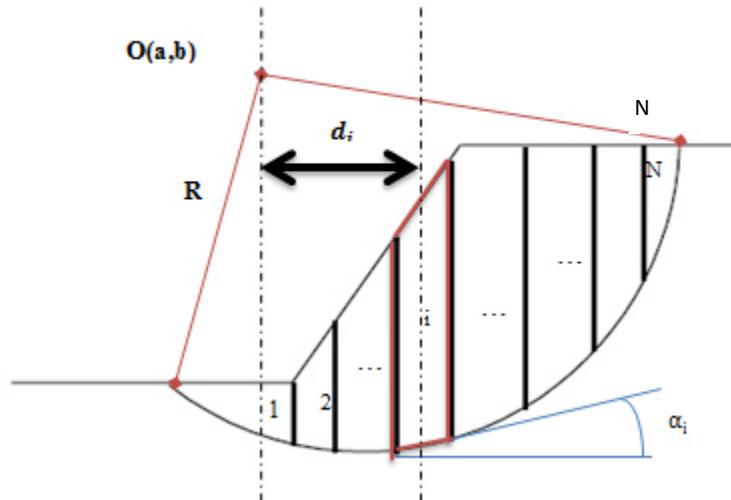
Tomando como referencia la Ilustración 2-5 se definen los siguientes objetos. Sea un talud determinado geométricamente por la distancia vertical  $H$  que hay entre los vértices  $\{v_1, v_2\}$  y su ángulo de talud  $\beta$  o su base  $B$ . El punto  $O(a,b)$  -donde  $a$  y  $b$  son las coordenadas de dicho punto desde algún sistema de referencia previamente definido- el centro de una circunferencia que determina con su perímetro la supuesta superficie de rotura. La *línea N.F.* representa el nivel freático del terreno el cual está compuesto por distintos estratos, quedando cada estrato representado por el peso específico  $\gamma_i$  medido en  $\frac{kN}{m^2}$ , la cohesión efectiva  $c'_i$  medida en  $kPa$  y  $\phi'_i$  el ángulo de rozamiento efectivo medido en  $^\circ$ , el peso  $W$  es el que ejerce la masa potencialmente deslizante que se encuentra sobre la hipotética superficie circular de fractura.

Ilustración 2-5: Talud y variables implicadas I



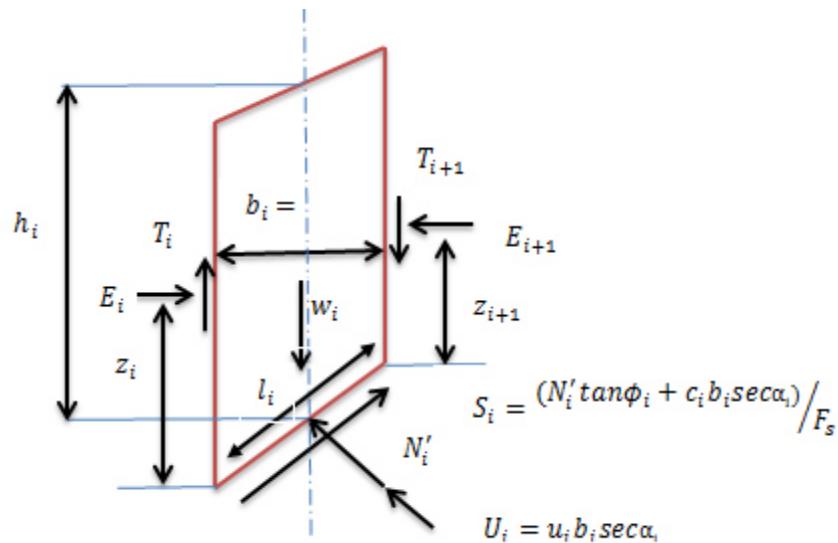
La Ilustración 2-5 representa el mismo talud bajo las mismas condiciones expuestas en las líneas anteriores, pero ahora dividido en un número finito de rebanadas enteras, no necesariamente iguales, limitadas superiormente por la superficie libre del talud e inferiormente por la hipotética superficie de fractura.

Ilustración 2-6: Talud y variables implicadas II



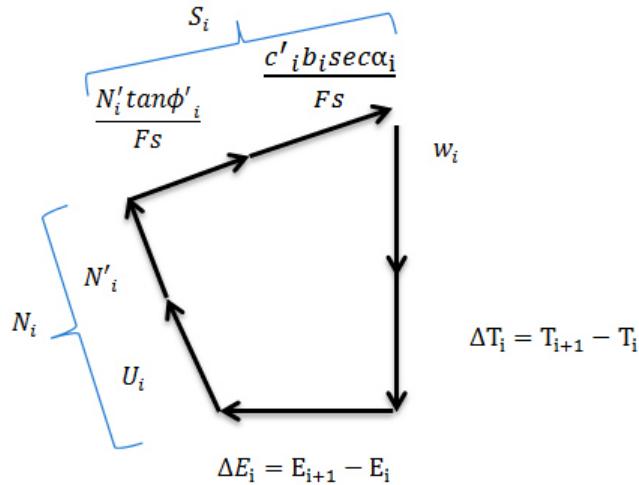
Extrayendo la rebanada  $i$  y añadiendo sobre ella todas las fuerzas que actúan se tiene

Ilustración 2-7: Rebanadas y esfuerzos implicados



Si ahora se impone el equilibrio estático, resulta el siguiente polígono de fuerzas

Ilustración 2-8: Polígono de fuerzas general



Uno de los puntos de partida del método es que el punto de aplicación de  $N_i$  se sitúa en el centro de la base de cada rebanada, esto se puede admitir debido a que si el grosor  $b_i$  de cada rebanada tiende a cero entonces  $N_i$  se puede considerar que presiona en el centro de la rebanada, pasando su línea de acción por el centro del círculo  $O(a,b_i)$ . Con lo cual  $N_i$  no genera momento respecto del centro  $O(a,b_i)$ . Por otro lado, la suma de los momentos de las fuerzas entre rebanadas a lo largo de todo el talud resulta nula. Así, tomando momentos con respecto a  $O$ , las únicas fuerzas implicadas son los pesos y las resultantes de resistencia al corte movilizado:

$$\sum_1^n W_i \cdot d_i = \sum_1^n S_i \cdot R$$

sustituyendo valores

$$\sum_1^n W_i \cdot R \sin \alpha_i = \sum_1^n \left[ \frac{c' i \cdot l_i}{F_s} + (N_i - u_i \cdot l_i) \frac{\tan \phi_i}{F_s} \right] R$$

Asumiendo ahora que el coeficiente de seguridad es único a lo largo de la superficie, se puede despejar éste de la expresión anterior, resultando:

**Ecuación 2-1**

$$FS = \frac{\sum_1^n [c'_{i \cdot} l_i + (N_i - u_{i \cdot} l_i) \tan \phi_i]}{\sum_1^n W_i \sin \alpha_i}$$

Todos los valores de esta ecuación son conocidos salvo  $N_i$  y es precisamente aquí donde, dependiendo de las hipótesis que se hagan para su cálculo, se formulan los distintos métodos.

Antes de pasar a describir los métodos, se va a hacer un balance del número de incógnitas que contiene el problema frente al número de ecuaciones del que se dispone. La Tabla 2-2 muestra el número de incógnitas existentes en el problema. Hay que tener en cuenta para la compresión de lo que se plantea, que si una rebanada incorpora una incógnita, entonces esta incógnita aparecerá tantas veces como rebanadas haya.

**Tabla 2-2: Número de incognitas totales**

<b>Variable</b>	<b>Nº de incognita</b>	<b>Observación</b>
<i>Fuerzas <math>N'_i</math></i>	$n$	<i>Una por rebanada</i>
<i>Puntos de aplicación <math>N'_i</math></i>	$n$	<i>Ídem celda inmediatamente superior</i>
<i>Resistencia al corte movilizada <math>S_{m_i}</math></i>	$n$	<i>Ídem celda inmediatamente superior</i>
<i>Fuerzas tangenciales entre rebanadas <math>T_i</math></i>	$n-1$	<i>En las rebanadas extremas solo existe la fuerza entre rebanada que esté hacia el interior del talud.</i>
<i>Fuerzas normales entre rebanadas <math>E_i</math></i>	$n-1$	<i>Ídem celda inmediatamente superior</i>
<i>Puntos de aplicación para fuerzas normales entre rebanadas <math>E_i</math></i>	$n-1$	<i>Ídem celda inmediatamente superior</i>
<i>Factor de seguridad</i>	1	<i>Solo existe un único FS</i>
suma	$3n+3(n-1)+1=6n-2$	

Las fuerzas inter-rebanadas están a los lados de cada rebanada y obviamente se anulan en los extremos del talud, ya que dichos extremos se encuentran “al aire”. Por esta razón las fuerzas entre rebanadas incorporan una incógnita menos que las demás. La Tabla 2-3 muestra el número de ecuaciones disponibles.

Tabla 2-3: Número de ecuaciones disponibles

Ecuación	Nº de ecuaciones	Observación
<i>Equilibrio de fuerzas horizontales</i>	$n$	<i>Una por rebanada</i>
<i>Equilibrio de fuerzas verticales</i>	$n$	<i>Ídem celda inmediatamente superior</i>
<i>Equilibrio de momentos</i>	$n$	<i>Ídem celda inmediatamente superior</i>
<i>Criterio de rotura de Mohr-Coulomb</i>	$n$	<i>Ídem celda inmediatamente superior</i>
suma	$4n$	

El número de hipótesis necesarias para poder resolver el problema será igual al grado de hiperestaticidad que tenga el problema

#### Ecuación 2-2

$$\text{nº hipótesis} = 6n - 2 - 4n = 2n - 2$$

A la hora de tomar momentos se formuló la primera hipótesis, y esta era afirmar que si el ancho de cada rebanada era lo suficientemente pequeño, entonces se podía considerar que el punto de aplicación de  $N'_i$  era el centro de la rebanada. Con lo cual que hay una incógnita menos. Para avanzar de manera ordenada esta hipótesis se incorporará más adelante.

#### 2.2.2.2 Métodos aproximados

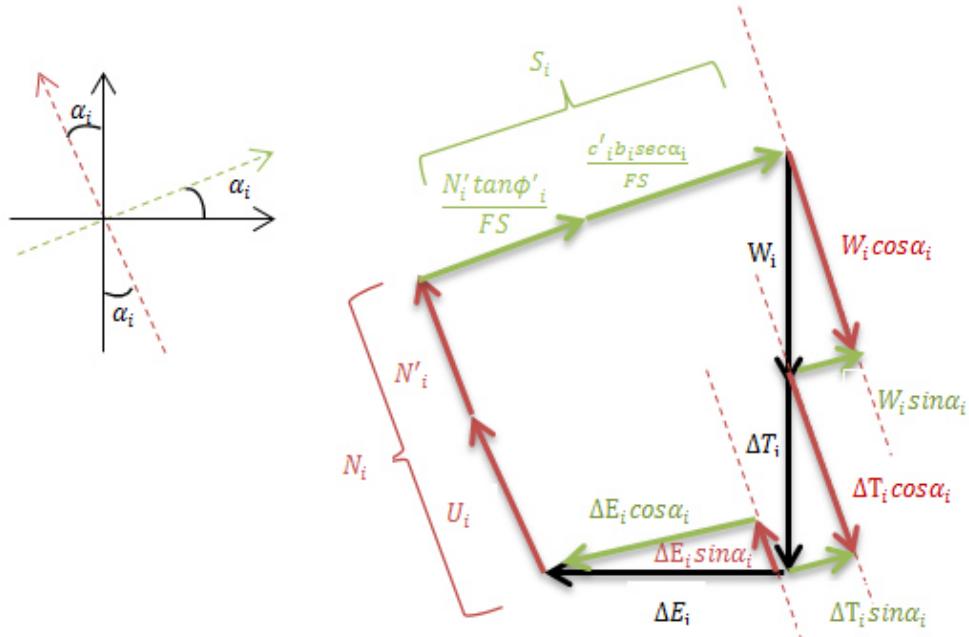
Como se dijo, estos métodos formulan hipótesis que simplifican el problema hasta convertirlo en isostático. Sin embargo, como se verá, estos métodos atentan contra el equilibrio del problema. Por esta razón se denominan métodos aproximados. Los métodos que se verán serán el método de Fellenius y el método de Bishop simplificado.

##### 2.2.2.2.1 Método de Fellenius

Se dejó planteada la Ecuación 2-1 y se dijo que lo que diferencian a los métodos es en la forma de calcular  $N_i$ . Se pasa ahora a desarrollar el método de Fellenius.

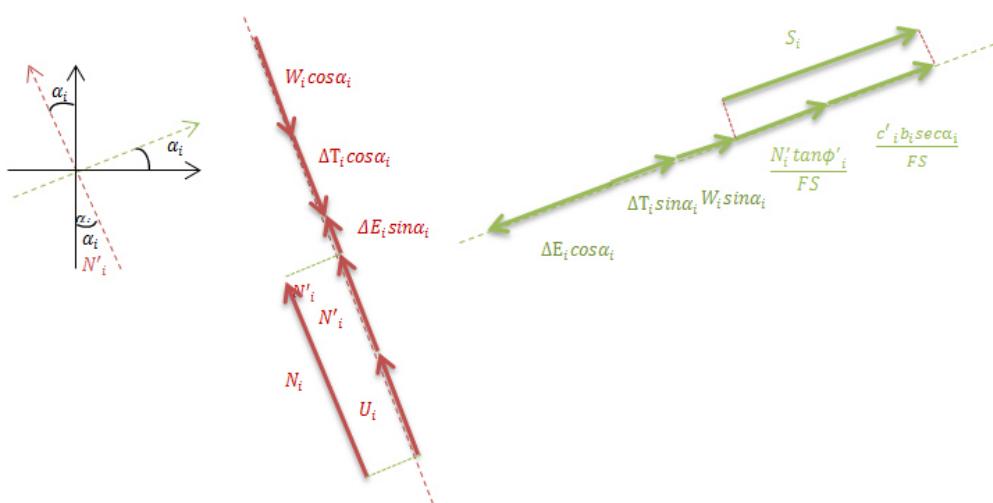
Si se elige la dirección en la que actúa  $N_i$  para calcular el equilibrio se tiene que

Ilustración 2-9: Polígono de fuerzas. Método de Fellenius



Representando ahora las distintas componentes perpendiculares por separado

Ilustración 2-10: Equilibrio de fuerzas. Método de Fellenius



Su expresión analítica sería

$$N_i + [E_i - E_{i+1}] \operatorname{sen} \alpha_i = [W_i + T_{i+1} - T_i] \cdot \cos \alpha_i$$

con lo que sustituyendo

$$FS = \frac{\sum_1^n [c'_i \cdot l_i + (W_i \cdot \cos \alpha_i - u_i \cdot l_i) \cdot \tan(\phi'_i)] + [[T_i - T_{i+1}] \cdot \cos \alpha_i - [E_i - E_{i+1}] \operatorname{sen} \alpha_i] \cdot \tan(\phi'_i)}{\sum_1^n W_i \cdot \operatorname{sen}(\alpha_i)}$$

Lo que en realidad diferencia el método de Fellenius de lo demás es que se hace la hipótesis de que las fuerzas entre rebanadas son nulas

$$\forall i \quad T_i = E_i = 0$$

Con lo cual

$$[[T_i - T_{i+1}] \cdot \cos \alpha_i - [E_i - E_{i+1}] \operatorname{sen} \alpha_i] = 0$$

Quedando el factor de seguridad de la siguiente forma

**Ecuación 2-3**

$$FS = \frac{\sum_1^n [c'_i \cdot l_i + (W_i \cdot \cos \alpha_i - u_i \cdot l_i) \cdot \tan(\phi'_i)]}{\sum_1^n W_i \cdot \operatorname{sen}(\alpha_i)}$$

Para demostrar que el problema es isostático se tiene la siguiente tabla

**Tabla 2-4: Hipótesis añadidas en el método de Fellenius**

Hipótesis	Nº de hipótesis	Observaciones
Punto de aplicación $N'_i$	$N$	Esta hipótesis se incorporó en el método general
Fuerzas tangenciales entre rebanadas nulas	$n-1$	Al no haber fuerza entre rebanadas en los extremos del talud, no genera ninguna hipótesis
Fuerzas normales entre rebanadas nulas	$n-1$	(ídem celda inmediatamente superior)
suma	$n+2(n-1)=3n-2$	

El grado de hiperestaticidad será

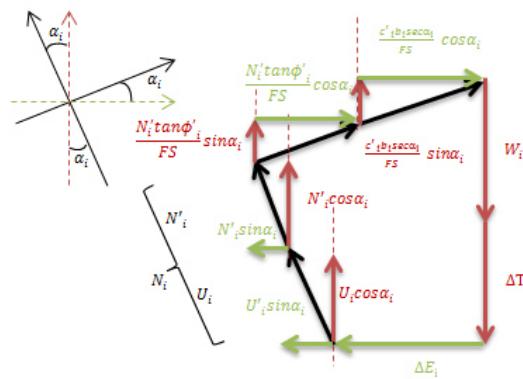
$$2n - 2 - (3n - 2) = -n$$

lo que supone haber realizado  $n$  hipótesis más de las estrictamente necesarias.

#### 2.2.2.2.2 Método de Bishop simplificado

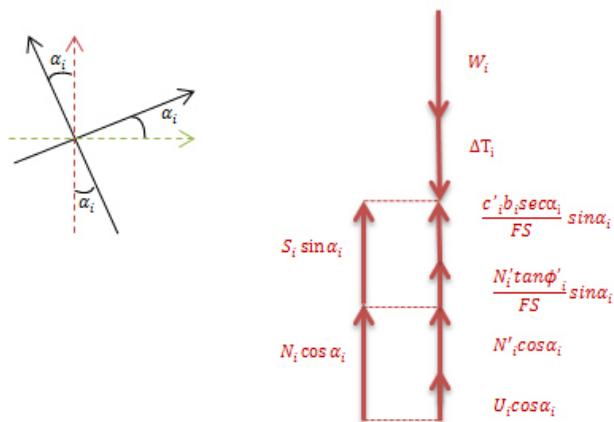
En 1954 Bishop desarrolló un procedimiento similar al de Fellenius, cambiando las hipótesis sobre las fuerzas entre rebanadas. De la misma forma que con el método de Fellenius, se parte de la Ecuación 2-1 planteada en el método general. A diferencia del método de Fellenius, el equilibrio no se hace en la dirección en la que opera  $N_i$  sino en la dirección vertical

**Ilustración 2-11: Polígono de fuerzas. Método de Bishop simplificado**



Atendiendo solo a las fuerzas verticales se tiene

**Ilustración 2-12: Equilibrio de fuerzas verticales. Método de Bishop simplificado**



$$N_i \cos \alpha_i + S_i \sin \alpha_i = W_i + \Delta T_i$$

Sustituyendo se tiene

$$(N'_i + u_i \cdot l_i) \cos \alpha_i + \left[ \frac{c'_i \cdot l_i}{FS} + N'_i \frac{\tan \phi'}{FS} \right] \sin \alpha_i = W_i + \Delta T_i$$

Despejando  $N'_i$

$$N'_i = \frac{W_i + \Delta T_i - l_i \left[ u_i \cos \alpha_i + \frac{c'_i}{FS} \sin \alpha_i \right]}{\cos \alpha_i + \frac{\tan \phi'}{FS} \sin \alpha_i}$$

Siendo  $b_i = l_i \cos \alpha_i$  y sustituyendo

$$FS = \frac{\sum_1^n \left\{ c'_i b_i + [(W_i + \Delta T_i - u_i b_i) \tan \phi'_i] \frac{\sec \alpha_i}{1 + \frac{\tan \phi'_i \tan \alpha_i}{FS}} \right\}}{\sum_1^n W_i \sin \alpha_i}$$

Bishop propone simplificar esta ecuación con la hipótesis de que solo las fuerzas verticales entre rebanadas son nulas, provocando así el no cumplimiento del equilibrio vertical:

$$\forall i \quad T_i = 0$$

Con lo cual

**Ecuación 2-4**

$$FS = \frac{\sum_1^n \left\{ c'_i b_i + [(W_i - u_i b_i) \tan \phi'_i] \frac{\sec \alpha_i}{1 + \frac{\tan \phi'_i \tan \alpha_i}{FS}} \right\}}{\sum_1^n W_i \sin \alpha_i}$$

Como se puede observar en dicha ecuación, calcular el  $FS$  por este método implicará un proceso iterativo, partiendo normalmente de  $FS=1$ .

Para demostrar que el problema es isostático se tiene que

**Tabla 2-5: Hipótesis en el método de Bishop simplificado**

Hipótesis	Nº de hipótesis	Observaciones
<i>Punto de aplicación <math>N'_i</math></i>	$n$	<i>Esta hipótesis se incorporó en el método general</i>
<i>Fuerzas tangenciales entre rebanadas nulas</i>	$n-1$	<i>Al no haber fuerza entre rebanadas en los extremos del talud, no genera ninguna hipótesis</i>
<i>suma</i>	$n+(n-1)=2n-1$	

El grado de hiperestaticidad será

$$2n - 2 - (2n - 1) = -1$$

lo que supone haber realizado una hipótesis más de las estrictamente necesarias.

### 2.2.2.3 Métodos precisos

En el apartado se vio como la forma de calcular  $N_i$  dependía de las suposiciones que se hicieran sobre las fuerzas entre rebanadas. Las suposiciones que se hicieron en los métodos anteriores no cumplían con las ecuaciones del equilibrio estático. En el método de Fellenius se supuso que ambas componentes (vertical y horizontal) de estas fuerzas eran nulas y en el caso de Bishop solo la componente vertical se consideró nula.

Los métodos precisos establecen hipótesis sobre estas fuerzas que sí que cumplen con todas las ecuaciones de equilibrio. Debido a esto, estos métodos pueden salirse de la hipótesis de que la superficie potencial de falla tenga la forma de una circunferencia, pudiéndose elegir otras formas más complejas. Sin embargo, simplifica bastante los cálculos y su implementación computacional suponer que esta superficie es generada por el perímetro de una circunferencia, y es por esta razón por la que se seguirá considerando como hipótesis de partida que la superficie potencial de rotura será la formada por una circunferencia.

#### 2.2.2.3.1 Método de Morgenstern-Price

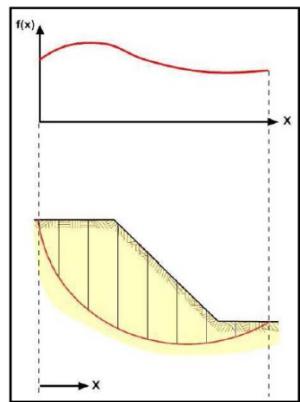
De la misma forma que Fellenius y Bishop establecieron hipótesis sobre las fuerzas entre rebanadas, Morgenstern-Price establece la siguiente hipótesis

**Ecuación 2-5**

$$\frac{T_i}{E_i} = \lambda \cdot f(x)$$

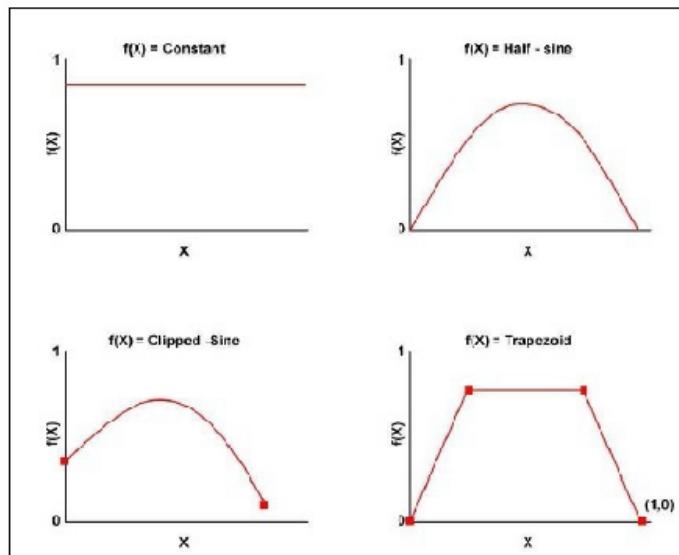
donde  $f(x)$  describe de alguna manera la forma en que la relación de fuerzas entre rebanadas tangenciales y normales, varía a lo largo del talud, y el coeficiente  $\lambda$  es un factor de corrección a determinar para que se cumplan las condiciones de equilibrio horizontal y de momentos. En la Ilustración 2-13 se puede ver la distribución de  $f(x)$  a lo largo de un talud, cumpliéndose en cada lado de cada rebanada la Ecuación 2-5

Ilustración 2-13: Función  $f(x)$  a lo largo del talud



Morgenstern y Price advirtieron que la función  $f(x)$  que se utilice no debería de influir en el resultado final (Morgenstern-Price, 1965). La Ilustración 2-14 muestra las funciones típicas que se suelen usar para  $f(x)$ .

Ilustración 2-14: Funciones  $f(x)$  típicas



Sustituyendo se tendrá

$$FS = \frac{\sum_1^n \left\{ c'_i b_i + [(W_i + \lambda(f_{i+1}E_{i+1} - f_iE_i) - u_i b_i) \tan \phi'_i] \frac{\sec \alpha_i}{1 + \frac{\tan \phi'_i \tan \alpha_i}{FS}} \right\}}{\sum_1^n W_i \sin \alpha_i}$$

Para demostrar que el problema es isostático se tiene que

**Tabla 2-6: Hipótesis en el método de Morgenstern-Price**

Hipótesis	Nº de hipótesis	Observaciones
Punto de aplicación $N'_i$	$n$	Esta hipótesis se incorporó en el método general
Fuerzas tangenciales entre rebanadas nulas	$n-1$	Al no haber fuerza entre rebanadas en los extremos del talud, no genera ninguna hipótesis
suma	$n+(n-1)=2n-1$	

El grado de hiperestaticidad será

$$2n - 2 - (2n - 1) = -1$$

Esto significaría que se ha realizado una hipótesis más de las estrictamente necesarias, pero no se ha tenido en cuenta que se ha incorporado una incógnita nueva, lambda  $\lambda$ . Con lo cual

**Tabla 2-7: Incognita adicional en el método de Morgenstern-Price**

Variable	Nº de incógnita	Observación
Lambda $\lambda$	1	Solo hay una variable $\lambda$ para todo el talud

El grado de hiperestaticidad será

$$-1 + 1 = 0$$

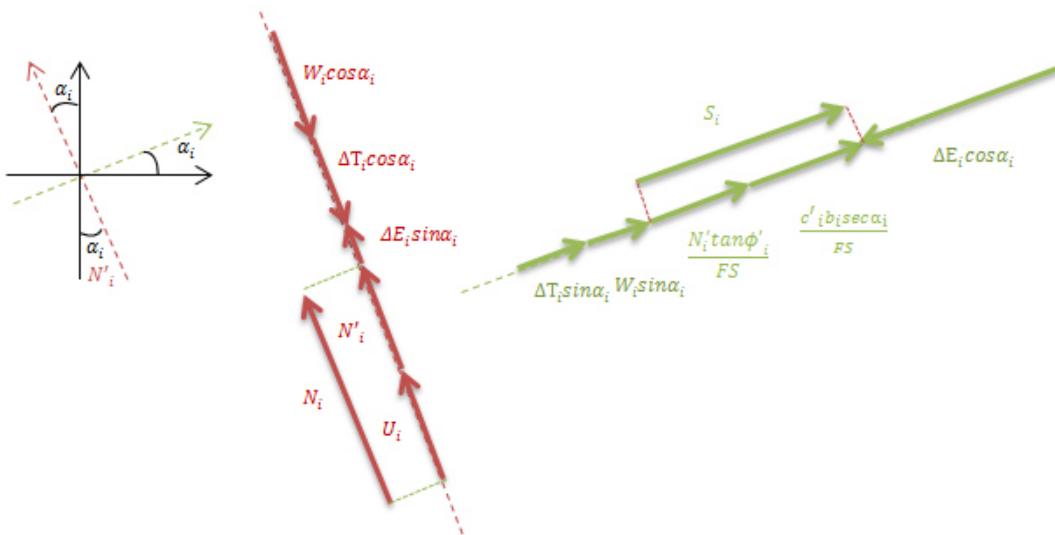
Quedando el problema determinado matemáticamente.

### 2.2.2.3.2 Adaptación del método de Morgenstern-Price (Zhu, Lee, Qian, & Chen, 2005)

Calcular el  $FS$  por este método, implica un proceso iterativo en el que el valor de  $FS$  y  $\lambda$  deberá ser calculado para cada iteración. El desarrollo de un método que solucione este problema ha generado bastantes discusiones en el ámbito científico, ya que las ecuaciones que se obtienen son complejas de manejar e implementar computacionalmente. Desde que este método fue formulado en el año 1965, diversos autores incluidos Morgenstern y Price, han diseñado diversos algoritmos para poder implementarlo computacionalmente (*Morgenstern and Price 1967; Fredlund and Krahn 1977; Zhu et al 2001, etc.*). Aunque estos procedimientos convergen no son cómodos de implementar, y además los tiempos de cálculo han demostrado ser bastante largos debido a la dificultad de las ecuaciones. En este proyecto se usará la adaptación del método formulada en el artículo “*A concise algorithm for computing the factor of safety using the Morgenstern–Price method*”. *Canadian Geotechnical Journal, Autores: D.Y. Zhu, C.F. Lee, Q.H. Qian, and G.R. Chen (2005)*.

#### 2.2.2.3.2.1 Explicación del método

Partiendo del diagrama de la **iError! No se encuentra el origen de la referencia.** y escribiendo las ecuaciones que se desprenden del mismo se tiene



$$N_i + [E_i - E_{i+1}] \operatorname{sen} \alpha_i = [W_i + T_{i+1} - T_i] \cdot \cos \alpha_i$$

$$S_i + [T_i - T_{i-1}] \operatorname{sen} \alpha_i = [W_i + E_i - E_{i-1}] \cdot \cos \alpha_i$$

Desarrollando ambas ecuaciones se tiene

$$N'_i = (W_i + \lambda f_{i-1} E_{i-1} - \lambda f_i E_i) \cos \alpha_i + (E_i - E_{i-1}) \sin \alpha_i - U_i$$

$$(N'_i \tan \phi'_i + c'_i b_i \sec \alpha_i) / FS = (W_i + \lambda f_{i-1} E_{i-1} - \lambda f_i E_i) \sin \alpha_i - (E_i - E_{i-1}) \cos \alpha_i$$

Sustituyendo la primera ecuación en la segunda se obtiene

**Ecuación 2-6**

$$E_i [(\sin \alpha_i - \lambda f_i \cos \alpha_i) \tan \phi'_i + (\cos \alpha_i + \lambda f_i \sin \alpha_i) FS] = E_{i-1} [(\sin \alpha_i - \lambda f_{i-1} \cos \alpha_i) \tan \phi'_i + (\cos \alpha_i + \lambda f_{i-1} \sin \alpha_i) FS] + FS T_i - R_i$$

en donde

**Ecuación 2-8**

$$R_i = [W_i \cos \alpha_i - U_i] \tan \phi'_i + c'_i b_i \sec(\alpha_i)$$

**Ecuación 2-9**

$$T_i = W_i \sin \alpha_i$$

Se puede decir que  $R_i$  es la suma de las fuerzas que contribuyen a la estabilidad y  $T_i$  es la suma de las fuerzas que contribuyen a la inestabilidad. El resto de la ecuación son términos que están relacionados con las fuerzas entre rebanadas. Por otro lado, agrupando y haciendo algún arreglo, la Ecuación 2-6 se puede enunciar como sigue

**Ecuación 2-10**

$$E_i \Phi_i = \Psi_{i-1} E_{i-1} \Phi_{i-1} + FS T_i - R_i$$

En donde

**Ecuación 2-11**

$$\Phi_i = [(\sin \alpha_i - \lambda f_i \cos \alpha_i) \tan \phi'_i + (\cos \alpha_i + \lambda f_i \sin \alpha_i) FS]$$

**Ecuación 2-12**

$$\Phi_{i-1} = [(\sin \alpha_{i-1} - \lambda f_{i-1} \cos \alpha_{i-1}) \tan \phi'_{i-1} + (\cos \alpha_{i-1} + \lambda f_{i-1} \sin \alpha_{i-1}) FS]$$

**Ecuación 2-13**

$$\Psi_{i-1} = [(\sin \alpha_i - \lambda f_{i-1} \cos \alpha_i) \tan \phi'_i + (\cos \alpha_i + \lambda f_{i-1} \sin \alpha_i) FS] / \Phi_{i-1}$$

Con la condición  $E_0 = 0$  y  $E_n = 0$  siendo estas las fuerzas normales interrebanadas al principio y al final de la superficie de deslizamiento, la Ecuación 2-10 es derivada a la siguiente expresión

**Ecuación 2-14**

$$FS = \frac{\sum_{i=1}^{n-1} (R_i \prod_{j=i}^{n-1} \Psi_j) + R_n}{\sum_{i=1}^{n-1} (T_i \prod_{j=i}^{n-1} \Psi_j) + T_n}$$

Por otro lado, considerando el equilibrio de momentos de la *rebanada i* entorno al centro de la base

$$E_i \left( z_i - \frac{b_i}{2} \tan \alpha_i \right) = E_{i-1} \left( z_{i-1} - \frac{b_i}{2} \tan \alpha_i \right) - \lambda \frac{b_i}{2} (f_i E_i + f_{i-1} E_{i-1})$$

Asumiendo que

$$M_i = E_i z_i \quad y \quad M_{i-1} = E_{i-1} z_{i-1}$$

Se tiene que

$$M_i = M_{i-1} - \lambda \frac{b_i}{2} (f_i E_i + f_{i-1} E_{i-1}) + \frac{b_i}{2} (E_i - E_{i-1}) \tan \alpha_i$$

Y finalmente como  $M_0 = 0$  y  $M_n = 0$  se llega a la Ecuación 2-15

**Ecuación 2-15**

$$\lambda = \frac{\sum_{i=1}^n [b_i (E_i + E_{i-1}) \tan \alpha_i]}{\sum_{i=1}^n [b_i (f_i E_i + f_{i-1} E_{i-1})]}$$

En conclusión, se llega a la Ecuación 2-14 y la Ecuación 2-15 que son sencillas de implementar.

### 2.2.2.3.2.2 Algoritmo de implementación

Se propone el siguiente algoritmo que facilita la implementación del método. (Zhu, Lee, Qian, & Chen, 2005)

Tabla 2-8: Adaptación al método de Morgenstern-Price

**MP-1.** Dividir la superficie de deslizamiento en un número  $N$  de rebanadas

**MP-2.** Calcular  $R_i$  y  $T_i$  usando la Ecuación 2-8 y la Ecuación 2-9 respectivamente para cada rebanada

**MP-3.** Especificar la función  $f(x)$

**MP4.** Dar valores iniciales a  $FS$  y  $\lambda$ .

**MP5.** Calcular  $\Phi_i$  y  $\Psi_{i-1}$  usando la Ecuación 2-11 y la Ecuación 2-13 para todas las rebanadas.

**MP6.** Calcular  $FS$  usando la Ecuación 2-14

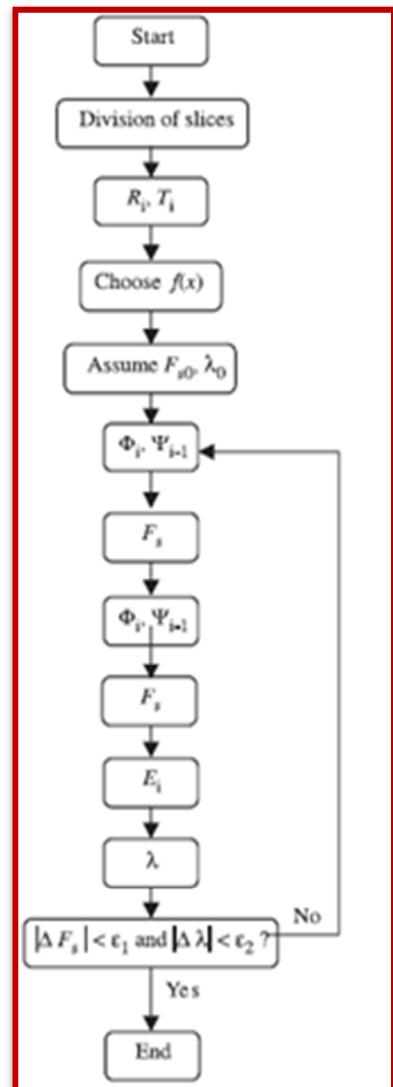
**MP7.** Repetir el paso MP5 y MP6 para mejorar los valores  $FS$ ,  $\Phi_i$  y  $\Psi_{i-1}$

**MP8.** Calcular  $E_i$  usando la Ecuación 2-10 para cada rebanada

**MP9.** Calcular  $\lambda$  usando la Ecuación 2-15

**MP10.** Repetir el algoritmo hasta que las diferencias de  $FS$  y  $\lambda$  entre dos valores consecutivos sean más pequeñas que las tolerancias  $\varepsilon_1$  y  $\varepsilon_2$  respectivamente.

Ilustración 2-15: Diagrama de flujo  
Morgenstern-Price



## 3. Implementación de los métodos de equilibrio límite

---

Antes de comenzar a implementar los métodos de equilibrio límite, es necesario construir algunos programas que permitan calcular las variables necesarias para poder diseñar dichos métodos. En general, la estrategia será la de crear un conjunto de funciones o subprocessos que sean fácilmente utilizables por cualquiera de los métodos de equilibrio límite.

En el apartado uno, se explica cómo se construye la superficie del talud. Una vez se tiene la superficie del talud, se da paso al siguiente apartado, en el cual se explica un algoritmo que permite introducir una circunferencia y calcular los puntos de corte con dicho talud. En el tercer apartado se da paso a la división en un número finito de rebanadas, luego se calculan las variables asociadas a cada rebanada y finalmente, se implementan los métodos propios de equilibrio límite.

### 3.1 Superficie del talud

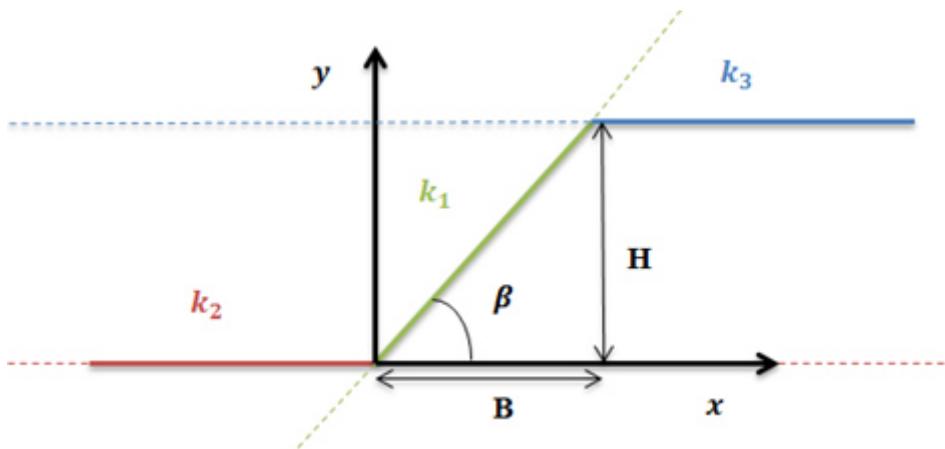
---

Para definir la superficie del talud se tomará la función a trozos siguiente

Ecuación 3-1

$$f(x) = \begin{cases} y_1 = 0 & x \leq 0 \\ y_2 = \frac{H}{B}x & 0 \leq x \leq B \\ y_3 = H & x > B \end{cases}$$

La primera recta  $k_1$  representa la porción de suelo inclinado, la segunda recta  $k_2$  representa la cota inferior del talud y la tercera recta  $k_3$  representa la cota superior. Los valores característicos del talud serán su altura  $H$  su base  $B$ , y su ángulo de inclinación  $\beta$ .

Ilustración 3-1: Talud y rectas  $k_1$ ,  $k_2$ ,  $k_3$ 

Por otra parte, se tomará como origen de coordenadas la intersección entre la primera recta  $k_1$  con la segunda  $k_2$ . Las distancias positivas en el eje horizontal se tomarán hacia la derecha y en el eje vertical hacia arriba, tal y como muestra la Ilustración 3-1

Se implementó la función *taludgeometría* para realizar esta operación. El código fuente se puede ver en el Algoritmo 3-1

Algoritmo 3-1: Función *taludgeometría*

```

1  function      y_talud = taludgeometria( B,x,H )
2  % Devuelve la geometría del talud
3  %
4  m=H/B; % Calculamos pendiente de la recta inclinada
5  y1=m.*x;
6  y2=0; % El talud siempre arranca en y=0;x=0.
7  y3=H;
8  y_talud=y2.* (x<=0)+y1.* ( (x>0) & (x<B) )+y3.* (x>=B) ;
9  end

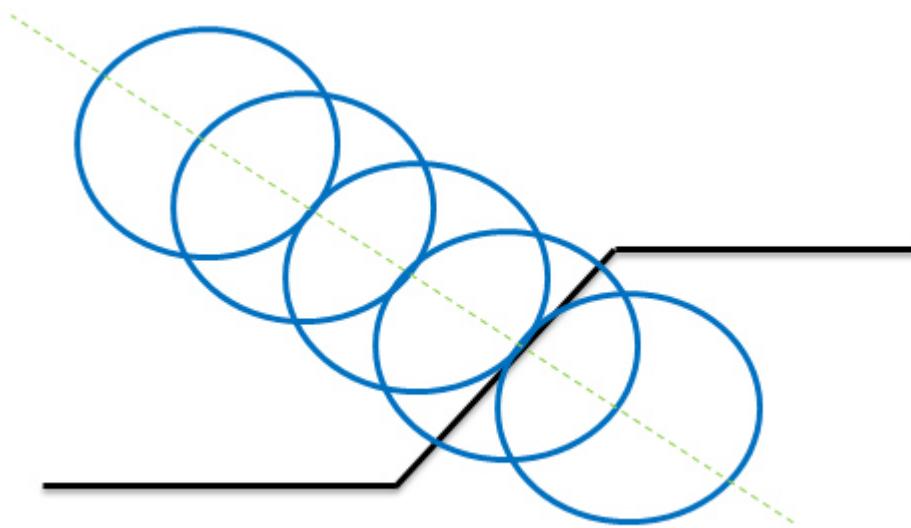
```

## 3.2 Superficie de rotura

Como ya se ha explicado, los métodos de equilibrio límite utilizan una circunferencia para representar la superficie de rotura. En realidad, solo se utilizan la semicircunferencia inferior, ya que la superior no puede formar superficie de rotura alguna como se verá más adelante.

El objetivo de este programa será poder probar tantas superficies como sea necesario o dicho de otra manera, se deberá de poder mover cualquier circunferencia en todo el espacio y tomar distintas decisiones al respecto. En la ilustración inferior se trata de llamar la atención sobre este aspecto, mostrando un conjunto de circunferencias que se van acercando hasta atravesar el talud. Cada una de estas circunferencias está en una situación particular con respecto al talud.

Ilustración 3-2: Circunferencias en el plano



Se hace necesario implementar un algoritmo que diferencie lo antes posible entre las circunferencias que no engendran superficie de rotura y las que sí.

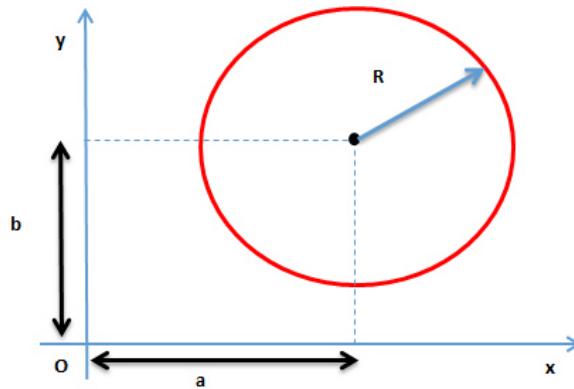
Para poder resolver el problema planteado, lo primero será estudiar analíticamente el problema que hay entre la circunferencia y el talud. Esto implica estudiar cada recta que conforma el talud con respecto a la circunferencia.

### 3.2.1 Intersección entre una recta y una circunferencia en el plano

El primer paso será estudiar la situación de cada recta por separado respecto de una circunferencia. Usando la ecuación general de una circunferencia de radio  $R$  y centro  $C(a,b)$ .

$$(x - a)^2 + (y - b)^2 = R^2$$

Ilustración 3-3: Definición de una circunferencia



Siendo la ecuación de una recta

$$y = mx + n$$

Se tiene el siguiente sistema de ecuaciones

**Ecuación 3-2**

$$\begin{cases} (x - a)^2 + (y - b)^2 = R^2 \\ y = mx + n \end{cases}$$

Sumando  $b$  a ambos lados de la ecuación de la recta y elevando al cuadrado se tiene

$$(y + b)^2 = (mx + n + b)^2$$

Desarrollando el cuadrado de la suma

$$(y + b)^2 = (mx)^2 + n^2 + b^2 + 2nb + 2mxn + 2mxb$$

Sustituyendo esto en la ecuación de la circunferencia y desarrollando

$$(m^2 + 1)x^2 + 2m(n - b) - 2a)x + (a^2 - R^2 + (n - b)^2) = 0$$

Que es una ecuación de segundo grado. Agrupando términos

**Ecuación 3-3**

$$\begin{aligned} A &= m^2 + 1 \\ B &= 2m(n - b) - 2a \\ C &= a^2 - R^2 + (n - b)^2 \end{aligned}$$

Quedando finalmente

**Ecuación 3-4**

$$Ax^2 + Bx + C = 0$$

Con lo cual, la situación de cualquier circunferencia respecto de una recta viene descrita por la *ecuación 3-4*. La función *raicircunfrecta* calcula las raíces de esta función.

**Algoritmo 3-2: Función *raicircunfereceta***

```

1  [-function [ X ] = raicircunfereceta( a,b,m,n,R )
2   [+ % Calcula los puntos de corte entre una circunferencia y una y una recta
13  - A=m^2+1;
14  - B=2*m*(n-b)-2*a;
15  - C=(a^2)-(R^2)+(n-b)^2;
16  - coeficientes=[A B C];
17  - [X] = segundo grado( coeficientes );
18  - end

```

La variable de salida de esta función es el vector X, el cual contiene las dos raíces obtenidas al resolver la ecuación de segundo grado, ordenadas de mayor a menor.

$$X = [x_1 \ x_2] / Ax^2 + Bx + C = 0 \ / \ x_1 \leq x_2$$

La situación de una circunferencia respecto de una recta puede ser de exterioridad, de tangencia o secante. Esto vendrá reflejado en las raíces  $X = [x_1 \ x_2]$  de la ecuación de segundo grado resuelta. Si la circunferencia es exterior, la recta no intersecta con la circunferencia en ningún punto, y las raíces son números complejos

$$x_1, x_2 \in C$$

Si la circunferencia es tangente, las raíces son reales e iguales.

$$x_1 = x_2 / x_1, x_2 \in R$$

Y si es secante, la recta intersecta con la circunferencia en dos puntos, uno de entrada y otro de salida, viene definida porque ambas raíces son reales y distintas.

$$x_1 \neq x_2 / x_1, x_2 \in R$$

Obviamente, se deberán excluir los casos de tangencia y exterioridad.

### 3.2.2 Intersección entre tres rectas y una circunferencia en el plano

El siguiente paso será aplicar las conclusiones alcanzadas en el apartado anterior a cada una de las rectas que forman el talud. Es decir, la función *raicircunfereceta* se aplicará sucesivamente sobre la recta  $k_1, k_2$  y  $k_3$  hasta obtener todos los puntos de corte con una circunferencia prefijada.

Cada recta se define por su pendiente y por su ordenada en el origen. La *tabla 3-1* muestra estos valores para cada una de las rectas que conforman el talud. En la primera columna se muestran las distintas rectas, en la segunda columna se muestra la pendiente de cada recta y en la tercera columna se muestra la ordenada en el origen de cada una de las rectas. La cuarta

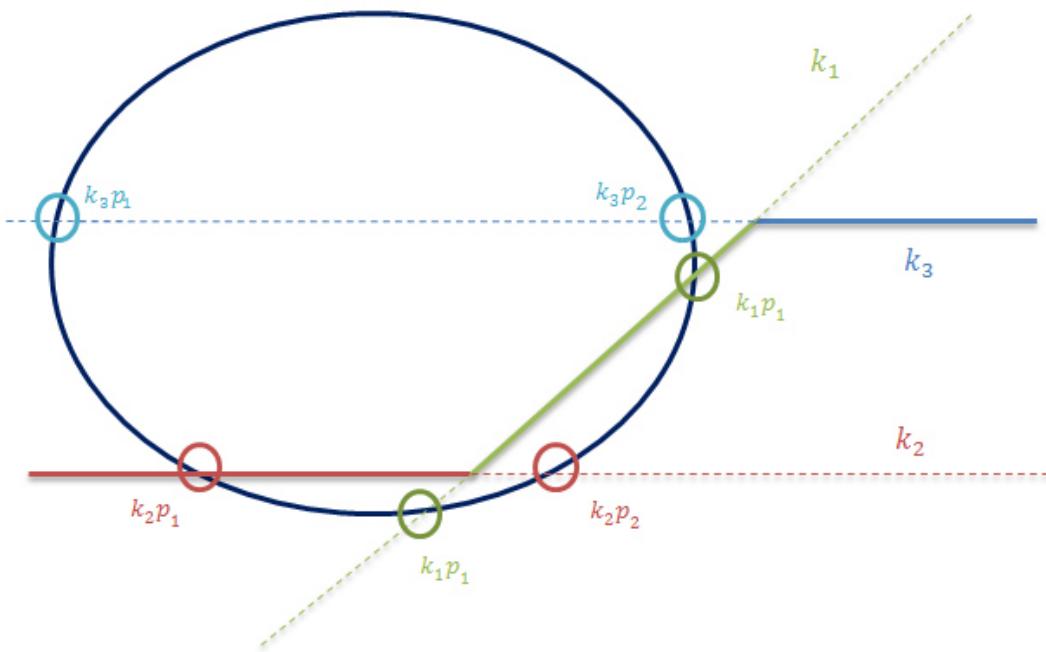
y quinta columna almacenan el nombre que adquieren las soluciones a dicho problema en el eje x y el eje y respectivamente.

**Tabla 3-1: Características de las rectas  $k_1, k_2, k_3$**

Recta	$m$	$n$	Raíces	Ordenadas
$k_1$	$\tan\beta$	0	$[k_1x_1 \ k_1x_2]$	$[k_1y_1 \ k_1y_2]$
$k_2$	0	0	$[k_2x_1 \ k_2x_2]$	$[k_2y_1 \ k_2y_2]$
$k_3$	0	$H$	$[k_3x_1 \ k_3x_2]$	$[k_3y_1 \ k_3y_2]$

En total se obtienen seis puntos de corte que deberán ser valorados. En la *ilustración 3-4* se observa a la circunferencia cortando con las tres rectas generadoras del talud y los seis puntos de corte.

**Ilustración 3-4: Seis puntos de corte entre la circunferencia y las tres rectas**



### 3.2.3 Selección de los puntos de corte

El siguiente paso es establecer una serie de reglas que permitan transformar el conjunto de seis soluciones en la solución final del problema. Lo más cómodo es explicar este apartado de forma pormenorizada de tal forma que se explicarán un conjunto de reglas que permiten encontrar la solución.

*R 1. Eliminación de los puntos  $k_2p_2$  y  $k_3p_1$*

Se puede ver en la figura () que los puntos  $k_2p_2$  y  $k_3p_1$  nunca podrán optar a ser solución del problema. No se hará demostración matemática de esto porque se considera una evidencia.

El conjunto de puntos se reduce a cuatro, los cuales se introducen dentro de la matriz *puntosposibles*

$$\text{puntosposibles} = \begin{pmatrix} k_1x_1 & k_1y_1 \\ k_1x_2 & k_1y_2 \\ k_2x_1 & k_2y_1 \\ k_3x_2 & k_3y_2 \end{pmatrix}$$

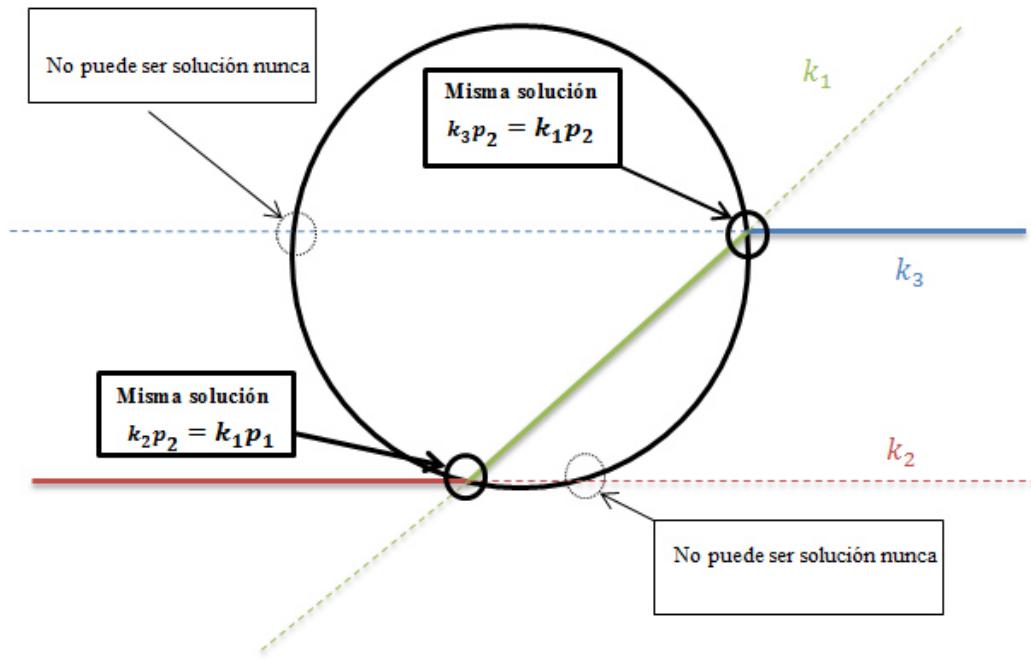
Utilizando la notación del Matlab

$$\text{puntosposibles} = [k_1p_1; k_1p_2; k_2p_1; k_3p_2]$$

#### R 2. Eliminación de puntos repetidos en la matriz *puntosposibles*

Esto sucede cuando la intersección de la circunferencia con las rectas que forman el talud se da en los vértices.

Ilustración 3-5: Reglas R1 y R2



Lógicamente la recta  $k_1$  siempre estará involucrada. La regla será simplemente la de eliminar el punto de corte que no sea con la recta  $k_1$ . Es decir, se creará la siguiente regla

$$\text{si } k_1p_1 = k_2p_2 \text{ entonces } k_2p_2 = \text{nan}$$

si  $k_1 p_2 = k_3 p_1$  entonces  $k_3 p_1 = \text{nan}$

Antes de dar paso a las siguientes reglas se va a mostrar en pseudocódigo lo que se ha hecho hasta ahora.

**Algoritmo 3-3: Función intersectaludcirc**

```

1  Funcion [puntosposibles]=intersectaludcirc(a,b,H,R,B)
2      //Recta k1.
3      m1=H/B;n1=0;
4      // Se invoca a la función raicircunferecta//
5      [X1]=raicircunferecta(a,b,m1,n1,R)
6      k1x1=X1(1);k1x2=X1(2);
7      // Se eliminan casos de tangencia o exterioridad//
8      Si k1x1 es complejo o k1x1=k1x2 Entonces
9          k1x1=nan;k1x2=nan
10     FinSi
11     // Se invoca a la función taludgeometria
12     k1y1=taludgeometria(B,k1x1,H);
13     k1y2=taludgeometria(B,k1x2,H);
14     // Se repite este bloque para X2 y X3//
15
16     // Se crea la matriz puntos posibles.
17     puntosposibles=[k1p1; k1p2;k2p1;k2p1 ] R.1
18     //Se eliminan los puntos si son iguales.
19     Si k1p1=k2p1 Entonces
20         k2p1=nan;
21         FinSi
22         Si k1p2=k3p2 Entonces
23             k3p2=nan;
24             FinSi
25     FinFuncion

```

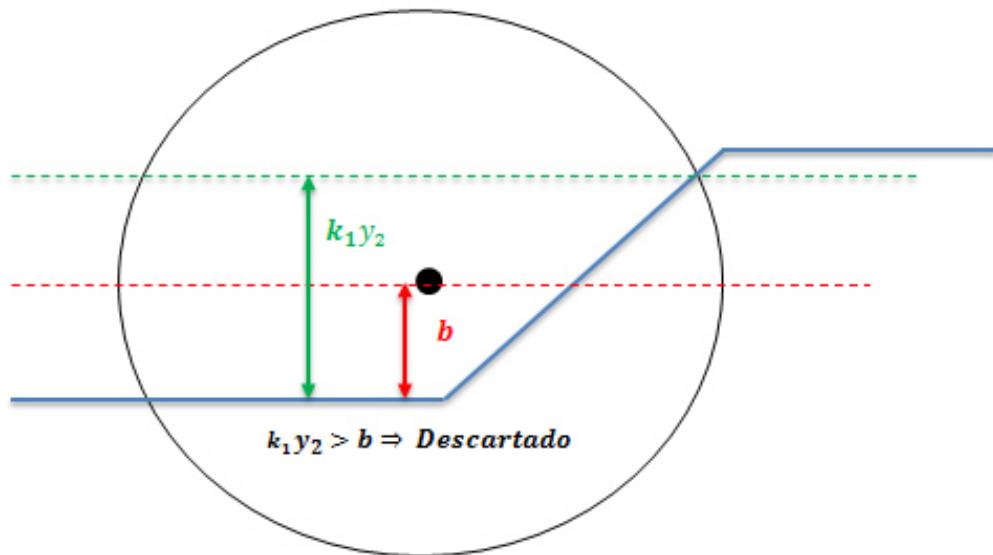
#### R 3. Corte con la recta $k_1$ . Variable $Q_1$

Siempre que una circunferencia forma una superficie de rotura con el talud, necesariamente está cortando con la recta  $k_1$ . La variable  $Q_1$  indicará si esta condición es verdadera o no, de tal forma que si  $Q_1=0$ , entonces se descarta esta circunferencia.

#### R 4. Eliminación de los punto de corte con ordenada mayor al centro b

Aquellos puntos de corte que estén situados por encima del centro  $b$  serán eliminados, tal y como muestra lo que se expone a continuación.

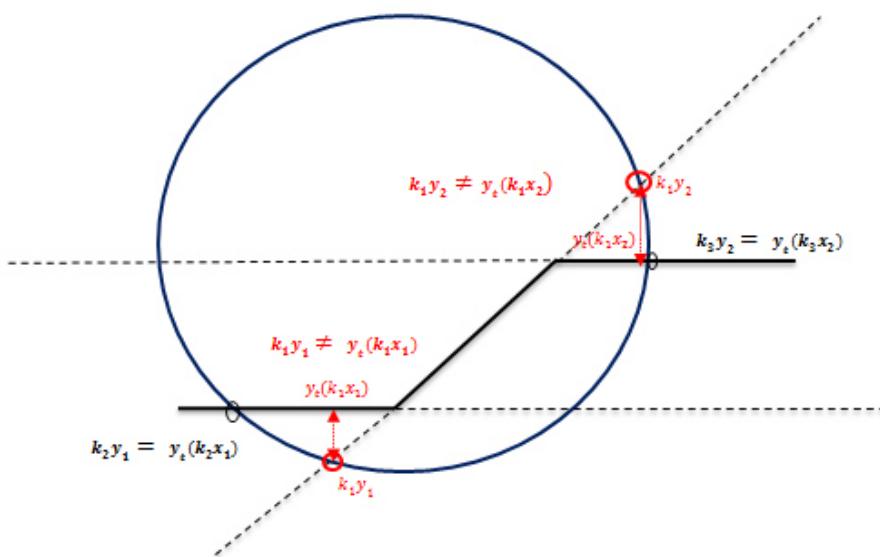
Ilustración 3-6: Regla R4. Eliminación de puntos con corte mayor al centro



### R 5 Comparación de puntos posibles con la superficie del talud.

La matriz *puntosposibles* contiene los puntos de corte de la circunferencia con el talud. Si se entra en la función *taludgeometría* con las abscisas de la matriz *puntosposibles* se obtendrá las posiciones que tendrían esas abscisas si fueran puntos del talud. Con lo cual, si se comparan estos puntos con las ordenadas de la matriz *puntosposibles*, se podrá determinar si pertenece a la superficie del talud o no.

Con lo cual, aquellas ordenadas que estén en la matriz *puntosposibles*, pero que tras introducir su abscisa correspondiente en la función *taludgeometría* se obtiene una ordenada distinta de la que hay en *puntosposibles*, serán descartados. Sin embargo, aquellos puntos que estén en el caso contrario se introducirán en una matriz llamada *puntosextremos*. (ver Ilustración 3-7)

Ilustración 3-7: Intersección de la matriz *puntosextremos* con la superficie del talud

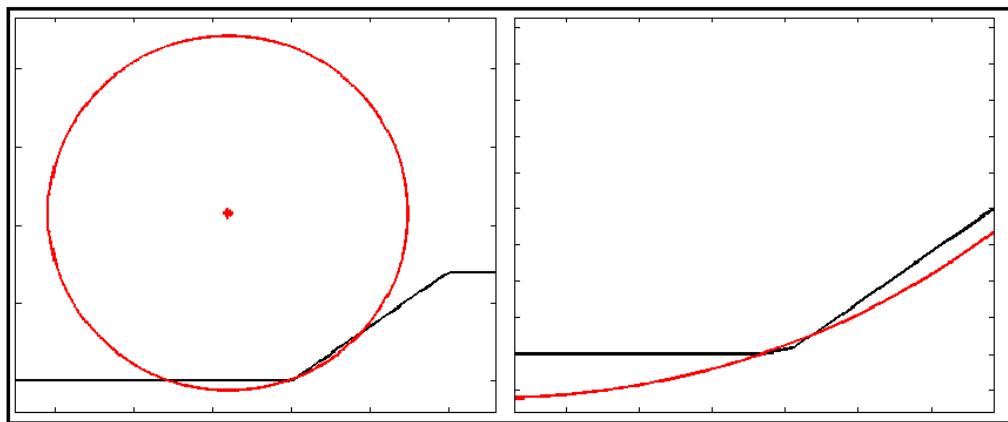
#### R 6. Filas de puntosextremos

Como la solución que se está buscando contiene como mínimo dos puntos (uno de entrada y otro de salida), si el número de filas de la matriz *puntosextremos* es menor a dos, entonces se descarta.

#### R 7 Circunferencia con doble entrada en el talud

En la *ilustracion 3-8* se ve un caso especial en el que la circunferencia tiene doble entrada dentro del talud. Como esta situación solo sucede de izquierda a derecha y el arco formado a la izquierda no formaría de ninguna forma una superficie de rotura, el problema se soluciona simplemente ordenando la matriz *puntosextremos* de menor a mayor (atendiendo a las raíces), y seleccionando los puntos situados a la derecha. Esta norma no afecta a los casos distintos de este, y este, lo arregla.

Ilustración 3-8: Circunferencia con doble entrada en el talud

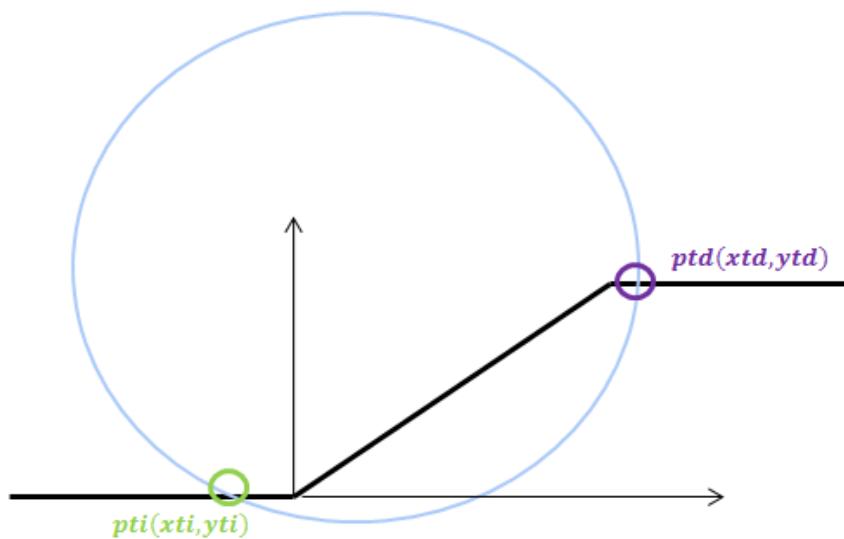


Como ya se está cerca de conseguir el objetivo propuesto, se puede aprovechar esta regla para asignar los valores a la posible solución. Llamando  $pti$  al punto extremo izquierdo y  $ptd$  al punto extremo derecho

$$\text{puntosextremos} = \text{ordenar}(\text{puntosextremos})$$

$$pti = \text{puntosextremos}(end - 1)$$

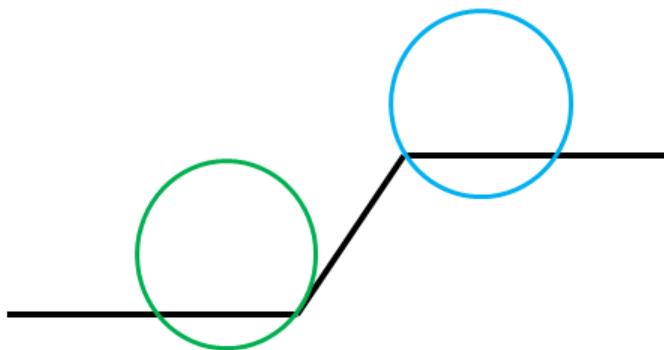
$$ptd = \text{puntosextremos}(end)$$

Ilustración 3-9: Puntos extremos  $pti, ptd$ 

*R 8 Puntos a la misma altura*

Los casos que se muestran en la Ilustración 3-10 se caracterizan porque su ordenada está a la misma altura y han conseguido llegar a la solución final. Esto solo puede suceder si la circunferencia corta sobre los vértices del talud. Con lo cual aquellas soluciones  $pti, ptd$  que estén a la misma altura serán eliminadas.

Ilustración 3-10: Puntos extremos a la misma altura



Con esto se termina el problema planteado. Ahora se presenta el pseudocódigo que reúne todos estos pasos, pero antes se presenta a las variables que se desprenden de realizar estos procesos. Los puntos extremos  $pti, ptd$  son las soluciones del problema planteado. Estas soluciones se caracterizan por sus abscisas  $xti, xtd$ , ya que su ordenada al estar necesariamente encima del talud se obtendrán cuando se necesiten entrando con las abscisas en la función *taludgeometría*. Por otro lado, se crea la variable  $K_4$  que informa con un uno lógico si la circunferencia forma superficie de rotura y con un cero cuando no. Finalmente, la variable  $r$  es un vector de tres dimensiones, donde cada posición le corresponde a cada recta, informando con un uno lógico las rectas a las que pertenece la solución.

Ecuación 3-5

$$r = \begin{bmatrix} (pti \text{ o } ptd) \in k_1 \\ (pti \text{ o } ptd) \in k_2 \\ (pti \text{ o } ptd) \in k_3 \end{bmatrix}$$

Si se cumplen los predicados del vector  $r$  se les asigna un uno lógico.

Por último, cada vez que en las reglas expuestas anteriormente se identificaba que esa circunferencia no podía formar una superficie de rotura, a nivel formal se estaba diciendo

$$k_4 = 0; xti = \text{nan}; xtd = \text{nan}; r = [0; 0; 0];$$

El pseudocódigo que se presenta a continuación se denomina *calculoextremos* y es el que reúne todos los conceptos analizados en este capítulo.

#### Algoritmo 3-4: Función calculoextremos

```

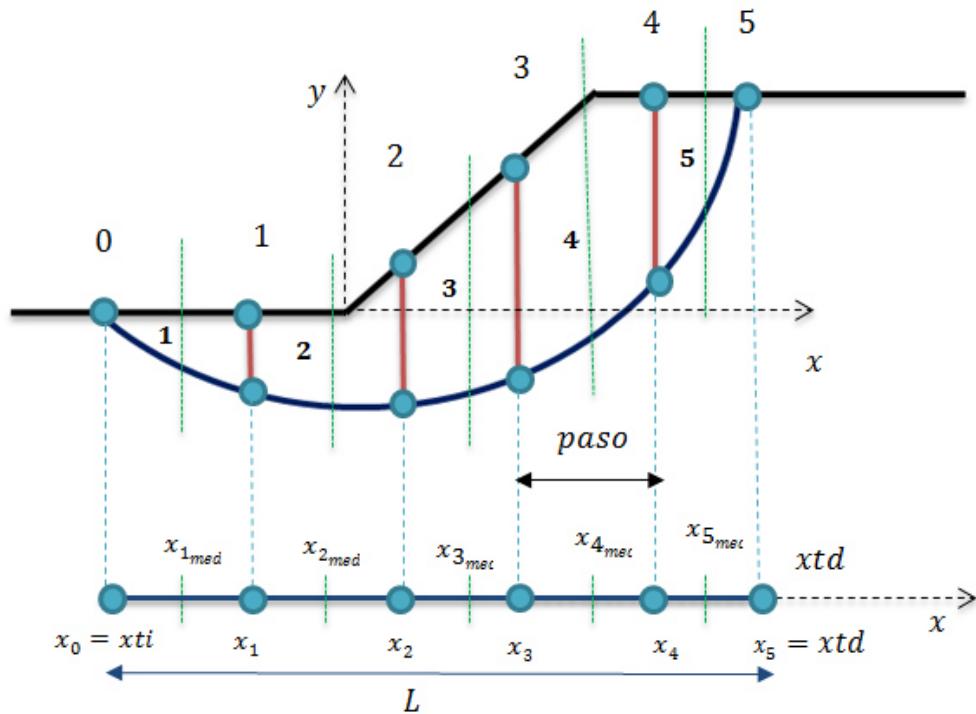
1  Funcion [ xti,xtd,k4,r]=calculoextremos(a,b,B,H,R)
2    //Se invoca a la función intersectaludcirc/
3    [puntosposibles] = intersectaludcirc( a,b,H,R,B );
4    // Regla R.3//
5    // Q1=1 si existe corte con k1//
6    Si Q1=1 Entonces R.3
7      // Regla R.4//
8      Si b<puntosposibles(:,2) Entonces R.4
9        puntosposibles(:,2)<-nan;
10       FinSi
11      // Regla R.5// // Se debe de invocar a la función taludgeometria
12      puntosextremos = (puntosposibles INTERSECCION y<-taludgeometria(puntosposibles(:,1))) R.5
13
14      // Regla R.6//
15      si numfilas(puntosextremos)>2 R.6
16
17      // Regla R.7//
18      puntosextremos=ordenar(puntosextremos)
19      xti=puntosextremos(1,end-1);
20      xtd=puntosextremos(1,end); R.7
21
22      // Regla R.8//
23      Si yti=ytd Entonces R.8
24        xti=nan;xtd=nan;k4=0;r=[0 0 0 ];
25        FinSi
26        xti=nan;xtd=nan;k4=0;r=[0 0 0 ]; R.8
27      FinSi
28
29      Sino
30        xti=nan;xtd=nan;k4=0;r=[0 0 0 ];
31      FinSi
32
33  FinFuncion

```

### 3.3 División de la superficie de rotura en rebanadas

El siguiente paso es dividir el área la superficie que queda por encima de la línea de rotura en  $n$  rebanadas, siendo  $n$  un número finito y entero positivo.

Ilustración 3-11: División en rebanadas general



En la imagen se aprecia una superficie dividida en cinco rebanadas. Estas rebanadas no deben de ser necesariamente iguales, pero en este caso lo son. Realmente lo que determina cada rebanada son los puntos coloreados, ya que delimitan sus extremos. Con lo cual, dividir una superficie en rebanadas es un problema idéntico a dividir una línea de largo  $L$  en un número  $n$  de divisiones. Como resultado, obtendrá una recta con  $m$  marcas, tal que  $m=n+1$ . Cada punto o marca, vendrá representado por su coordenada  $x$ , así la coordenada del punto  $i$  será  $x_i$ .

Si se quiere que el punto  $x_0$  y  $x_n$  formen parte de las divisiones y además se quiere que las rebanadas estén equiespaciadas se hará

$$x = \text{repartir}(xti, xtd, n + 1)$$

Esta operación genera el vector  $x$  el cual contiene la coordenada de todos estos puntos. Para conseguir el ancho de cada rebanada, simplemente se restarán dos elementos consecutivos cualesquiera del vector  $x$

$$paso = x_{i+1} - x_i$$

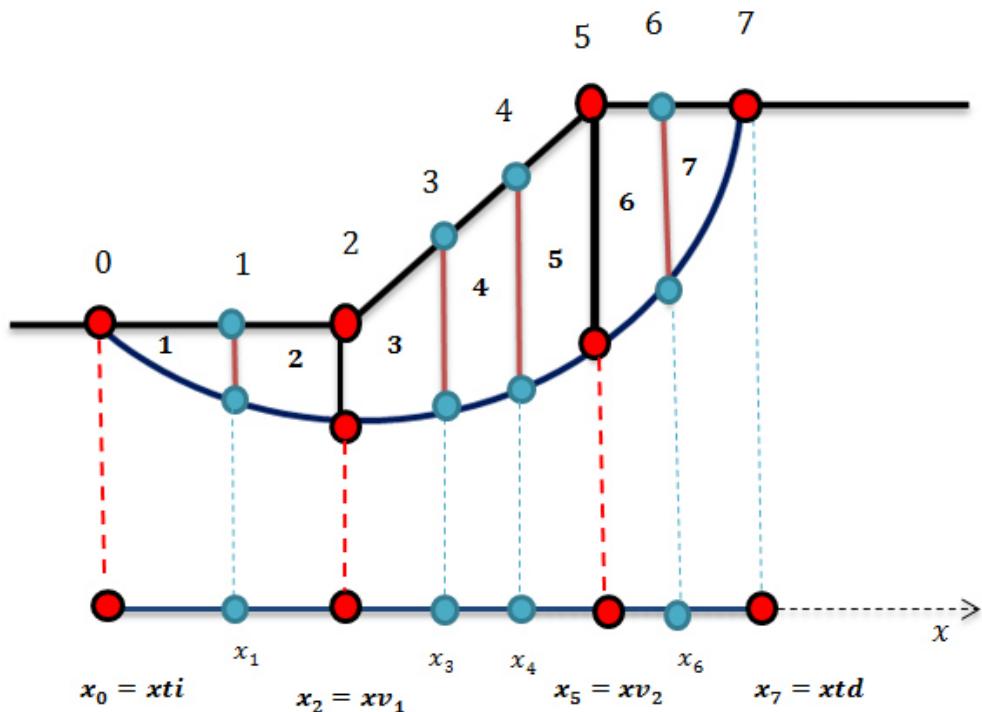
Debido a que también interesa un vector que contenga la coordenada del centro de cada rebanada, se creará el vector  $x_{medio}$ <sup>1</sup>

$$x_{medio} = x + \frac{paso}{2}$$

El problema se hace más complejo cuando se quiere imponer las normas que se explicarán en el desarrollo de este apartado. Se dividirá lo que queda de apartado en dos partes

- Cálculo de rebanadas
- Cálculo de parámetros

Ilustración 3-12: División en rebanadas forzando los vértices



<sup>1</sup> Matlab puede sumar vectores con constantes. A cada elemento del vector le suma la constante.

### 3.3.1 Cálculo de rebanadas

---

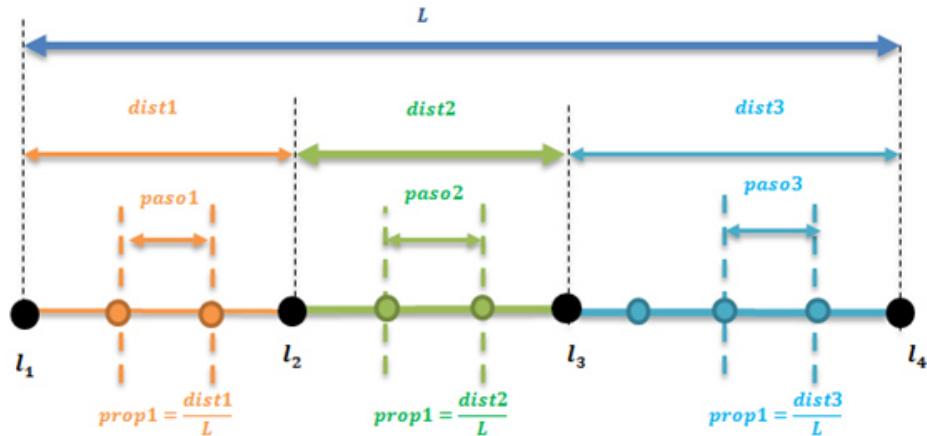
Como se aprecia en la *ilustración 3-12* se deberán de repartir  $m$  puntos a lo largo de  $L$  con la condición de que los vértices estén incluidos en estos  $m$  puntos. Esto significa que la abscisa de cada vértice debe de estar incluida dentro del vector  $x$ .

Para no escribir un párrafo engorroso, se van a dictar las condiciones con las que se quiere dividir el largo del talud

- 1) La distancia  $L$  representa el espacio que hay entre  $xti$  y  $xtd$ . Esta a su vez está dividida por tramos. Estos tramos son los formados por los puntos inicial, final y vértices.
- 2) La abscisa de los puntos inicial, final y vértice uno y dos - en caso pertenecer al espacio comprendido entre  $xti$  y  $xtd$  -deberán formar parte del conjunto de valores correspondientes al vector  $x$ .
- 3) Dentro de un mismo tramo los puntos están equiespaciados.
- 4) El reparto de puntos se hará de forma proporcional al tamaño del tramo.
- 5) Cada tramo deberá de estar representado como mínimo con una rebanada
- 6) El número de rebanadas después del reparto deberá de ser igual al número de rebanadas con el que se pidió resolver el problema.
- 7) Los distintos tramos pueden tener rebanadas de distinto ancho, sin embargo se intentará que este ancho sea lo más parecido posible.

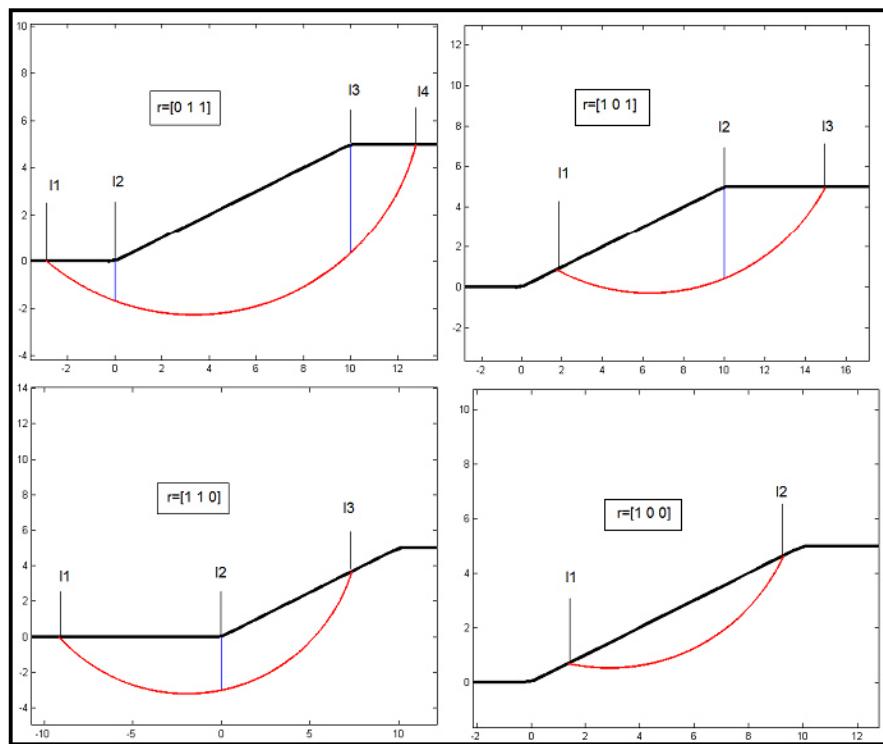
Estas condiciones se apoyan con la siguiente imagen

Ilustración 3-13: Variables implicadas en la división en rebanadas



Por otro lado, los vértices no siempre quedan en el interior de la superficie de rotura. Esto se aprecia en la imagen siguiente

Ilustración 3-14: Casos posibles en la división en rebanadas



Por esta razón se define la variable  $l_i$ . Esta variable permite resolver el problema para cualquiera de los casos. La variable  $l_i$  delimita los tramos y se enumera de izquierda a derecha según van apareciendo los puntos inicial, final y vértices.

Lo que se deberá hacer para poder dividir la masa del terreno en  $n$  rebanadas será identificar el caso en el que se encuentra la superficie de deslizamiento respecto del talud. Para ello se hará uso del vector  $r$ , el cual informa de las rectas que están siendo cortadas por la circunferencia. Se aplicará entonces la siguiente regla

El siguiente código muestra cómo se realiza la asignación de la variable  $l_i$  en función del vector  $r$ .

**Algoritmo 3-5: Subproceso de la función *divisorderebanadas***

```

3      % Subprocesos tramos
4      if isequal(r,[0 1 1])
5          l1=xti;l2=0;l3=B;l4=xtd;
6      elseif isequal(r,[1 1 0])
7          l1=xti;l2=0;l3=xtd;l4=[];
8      elseif isequal(r,[1 0 1])
9          l1=xti;l2=B; l3=xtd;l4=[];
10     elseif isequal(r,[ 1 0 0])
11        l1=xti;l2=xtd; l3=[];l4=[];
12     end
13

```

El siguiente paso será determinar el número de rebanadas que deberá de tener cada tramo.

**Algoritmo 3-6: Subproceso de la función *divisorderebanadas***

```

% Distancias y proporciones por tramo
distanciatotal=xtd-xti;
dist1=l2-l1;
dist2=l3-l2;
dist3=l4-l3;
% Proporciones por tramo
prop1=dist1/distanciatotal;
prop2=dist2/distanciatotal;
prop3=dist3/distanciatotal;
% Rebanadas enteras por tramo
reb1=round(prop1*rebanadas);
reb2=round(prop2*rebanadas);
reb3=round(prop3*rebanadas);

```

Como el número de rebanadas por tramo son el resultado de algunos cálculos, será necesario redondearlas. Esta operación puede provocar que, si el tramo no es lo suficientemente grande, el número entero tras el redondeo será igual a cero y como ya se dijo, es necesario que cualquier tramo esté representado con al menos una rebanada. El algoritmo Ilustración 3-6 resuelve este conflicto.

**Algoritmo 3-7: Subproceso de la función *divisorderebanadas***

```

% Tramos no representados
if reb1==0;
    reb1=reb1+1;
end;
if reb2==0;
    reb2=reb2+1;
end;
if reb3==0;
    reb3=reb3+1;
end

% Generamos el vector reb
reb=[reb1 reb2 reb3];
dimensionreb=length(reb);

```

Una vez se ha hecho esto, se almacenan las rebanadas en el siguiente vector

$$reb = [reb_1 \ reb_2 \ reb_3]$$

Por otra parte, al redondearse el número de tramos por tramo, es posible que el número de rebanadas inicial se vea alterado.

Es posible que en los pasos anteriores, ya sea por el redondeo o bien por forzar la representación total, el número de rebanadas totales haya variado. Es decir  $reb_1 + reb_2 + reb_3 \neq N$ . Por esta razón, se deberán añadir o sustraer rebanadas hasta que el número de rebanadas coincida con las iniciales. Debido a que todas las rebanadas deberán tener el ancho lo más parecido posible, la forma de sustraer o añadir rebanadas que altera menos el ancho de las mismas, es hacerlo sobre el tramo que tenga mayor número de estas.

Siendo  $reb_{mayor}$  el tramo que tiene más rebanadas, se plantea el siguiente algoritmo (Algoritmo 3-8)

**Algoritmo 3-8: Subprocesos de la función *divisorderebanadas***

```

Reajuste de rebanadas
while rebanadas < sum(reb)
    [rebmayor,pos]=max(reb);
    reb(pos)=reb(pos)-1;
end

while rebanadas > sum(reb)
    [rebmayor,pos]=max(reb);
    reb(pos)=reb(pos)+1;
end

```

Ya se dispone del número de rebanadas que se asignará a cada tramo, siendo estas  $reb_1, reb_2, reb_3$ .

El conjunto de algoritmos que se han expuesto hasta ahora se realizan uno tras otro, y forman parte de un único bloque, el cual será invocado como si fuera una función llamada *repartorebanadas* y que proporciona las variables  $reb_1, reb_2, reb_3, l1, l2, l3, l4$ .

### 3.3.2 Cálculo de parámetros

---

Ya se sabe el número de rebanadas que se incorporarán en cada tramo. Ahora se dará paso al cálculo de los puntos por cada tramo y el ancho que estos tendrán. Cada tramo tiene la siguiente cantidad de puntos

$$m_{total} = rebanadas + 1$$

$$m_1 = reb_1 + 1$$

$$m_2 = reb_2 + 1$$

$$m_3 = reb_3 + 1$$

Ahora se distribuyen los puntos por cada tramo.

$$x_i = linspace(l_i, l_{i+1}, M_i)$$

Se calcula el paso de cada tramo

$$paso_i = (x_{i+1} - x_i)$$

Y finalmente se calcula el vector de abscisas en el punto medio de cada rebanada

$$x_{medio_i} = x_i + \frac{paso_i}{2}$$

Por otro lado, por razones que se comprenderán más adelante, interesa tener un vector de dimensión igual al número de rebanadas totales que contenga el ancho de cada rebanada. Este vector será

$$pasovector_i = paso_i * ones(1, reb_i)$$

En el Algoritmo 3-9 se aprecia lo que se acaba de explicar.

Por otro lado, se aprecia en el código mostrado a continuación, una serie de condiciones que no se han explicado. Estas condiciones permiten saber si el tramo existe o no. No se dirá más al respecto para no hacer más engorroso el texto.

**Algoritmo 3-9: Subproceso de la función *divisorderebanadas***

```

% Tramo 1
x1=linspace(11,12,M1); % Puntos M1
paso1=x1(2)-x1(1);      % Paso1
x1_medio=x1+paso1/2;x1_medio(end)=[]; % Puntos M1 medios
paso1vector=paso1*ones(1,reb1); % Vector de pasos por rebanada

% Tramo 2
if length(dist2)==1
    x2=linspace(12,13,M2); % Puntos M2
    paso2=x2(2)-x2(1);      % Paso2
    x2_medio=x2+paso2/2;x2_medio(end)=[]; % Puntos M2 medios
    paso2vector=paso2*ones(1,reb2); % Vector de pasos por rebanada
else
    x2=[];paso2vector=[];x2_medio=[];
end

% Tramo 3
if length(dist3)==1
    x3=linspace(13,14,M3); % Puntos M3
    paso3=x3(2)-x3(1);      % Paso3
    x3_medio=x3+paso3/2;x3_medio(end)=[]; % Puntos M3 medios
    paso3vector=paso3*ones(1,reb3); % Vector de pasos por rebanada
else
    x3=[];paso3vector=[];x3_medio=[];
end

```

Solo queda juntar las variables que se han creado para cada tramo por separado

$$\text{pasovector} = [\text{pasovector}_1 \text{ pasovector}_2 \text{ pasovector}_3]$$

$$x = [x_1 \ x_2 \ x_3]$$

$$x_{medio} = [x_{medio_1} \ x_{medio_2} \ x_{medio_3}]$$

Debido a que la estrategia que se ha usado es la de tratar a cada tramo como un tramo independiente, es decir, el punto final de un tramo coincidirá con el punto inicial del siguiente tramo, se genera el problema de que a la hora de concatenar todos los puntos para obtener la distribución total de puntos, los puntos final de tramo o comienzo de tramo estarán repetidos. Es decir

$$m < \text{length}(x_1) + \text{length}(x_2) + \text{length}(x_3)$$

Para quitar este problema se creará un pequeño algoritmo que analiza dicha distribución y elimina los puntos repetidos.

**Algoritmo 3-10: Subproceso de función *divisorderebanadas***

```
% Juntamos todos los resultados

pasovector=[pasolvector paso2vector paso3vector];%Pasos vector
x_medio=[x1_medio x2_medio x3_medio];%Puntos M medio
x= [x1 x2 x3 ];%Puntos M. Este vector tiene valores repetidos

% Eliminamos los valores repetidos de x.
countrep=0;
repetidos=0;
for s=2:length(x)
    if x(s)== x(s-1)
        countrep=countrep+1;
        repetidos(countrep)=s; %#ok<*AGROW>
    end
end

if repetidos~=0
x(repetidos)=[];
end
```

Y finalmente se consigue el objetivo deseado. Este subprocesso se denominará *vectoresrebanada*. La función que integra los dos subprocessos que se han explicado en este apartado se denomina *divisorderebanadas* y su funcionamiento es el mostrado en el siguiente algoritmo.

**Algoritmo 3-11: Pseudocódigo de la función *divisorderebanadas***

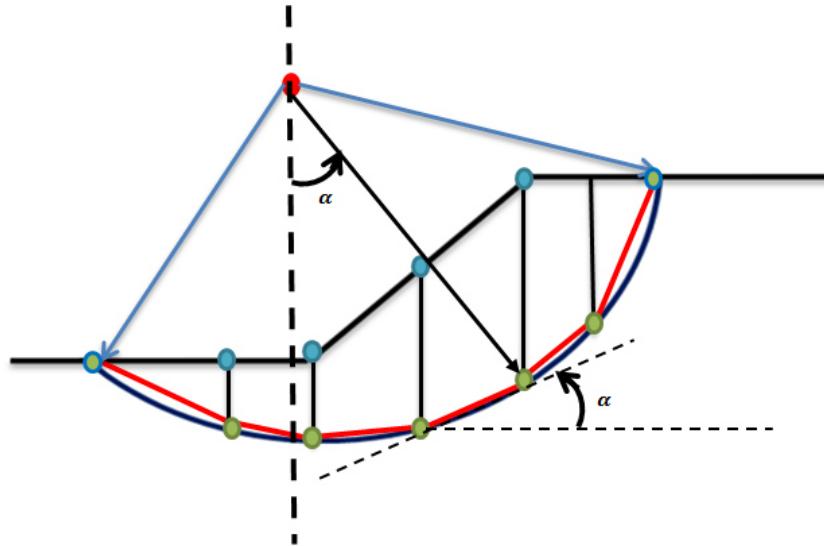
```
1 Funcion [ x,x_medio,pasovector ] = divisorderebanadas( xti,xtd,rebanadas,r,B)
2 // Se invoca al subprocesso repartoderebanadas//
3 [reb1,reb2,reb3,11,12,13,14]<-repartoderebanadas(xti,xtd,rebanadas,r,B)
4 // Se invoca al subprocesso vectoresrebanada//
5 [x,x_medio,pasovector]<-repartoderebanadas(reb1,reb2,reb3,11,12,13,14,rebanadas
6 FinFuncion
7
```

### 3.4 Variables asociadas a cada rebanada

En la siguiente ilustración se muestra un talud, con una supuesta rotura circular, dividido en  $n$  rebanadas. El ángulo  $\alpha$  se comienza a medir desde el cuarto cuadrante de un eje cartesiano, tomándolo como positivo en sentido anti-horario. Este ángulo  $\alpha$  coincide con el ángulo que forma la parte inferior de cada rebanada respecto de la horizontal. Este ángulo coincide solamente en los puntos que están en el vector  $x$ , ya que los puntos que están en el vector  $x_{medio}$  tienen una inclinación diferente, esto es debido a que la superficie circular es ahora eliminada y cambiada por una serie de rectas que van de punto a punto, por esta razón, el

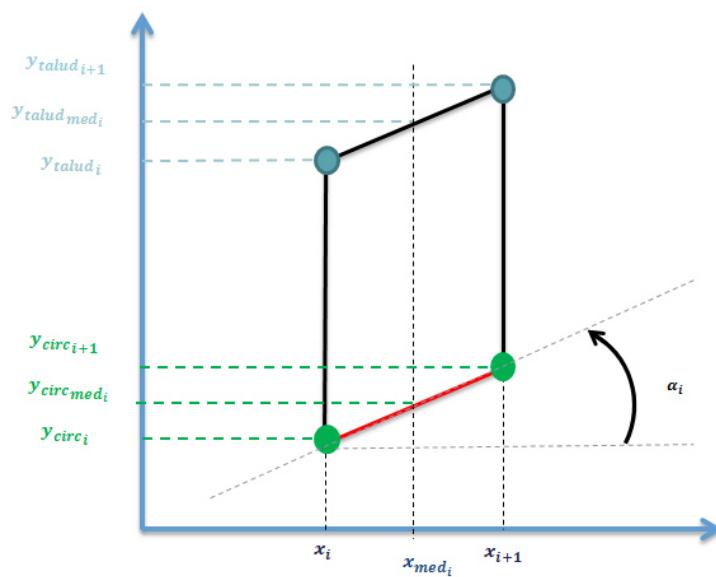
ángulo de inclinación  $\alpha$  de los puntos medios de la rebanada, se relaciona con la pendiente de la recta que pasa por cada rebanada.

Ilustración 3-15: Parámetros I



Se extrae una rebanada  $i$  y se examina qué variables se quiere saber de ella. De forma pormenorizada se tiene (ver Ilustración 3-16)

Ilustración 3-16: Parámetros II



De forma pormenorizada se tiene:

### 3.4.1 Ordenadas de las esquinas superiores de cada rebanada

---

Por encima del eje de abscisas, la rebanada queda delimitada por la superficie del talud., para calcularla simplemente habrá que entrar en la función *taludgeometría* con los vectores  $x, x_{medio}$ . Los vectores obtenidos serán  $y_{talud}, y_{talud\_med}$ .

### 3.4.2 Ordenadas del semicírculo

---

#### a) Vector: $y_{circ}$

Por debajo del eje de abscisas, la rebanada queda delimitada por la superficie inferior de la circunferencia, el vector  $y_{circ_i}$  contiene la ordenada de todos los puntos que pertenecen al vector  $x$

$$y_{circ_i} = -\sqrt{R^2 - (x_i - a)^2} + b$$

#### b) Vector: $y_{circ\_med}$

Como ya se mencionó, cada dos puntos consecutivos formados por  $(x_i, y_{circ_i})$  y  $(x_{i+1}, y_{circ_{i+1}})$  están conectados por un arco de circunferencia. Sin embargo, para llevar a cabo los métodos de equilibrio límite es necesario simplificar el problema imponiendo que están conectados por una línea recta.

Por esta razón la ordenada del vector  $x_{med}$  no estará sobre la propia curva de la circunferencia, sino sobre la recta que une ambos puntos. Para calcular dicha ordenada se hará lo siguiente. Tomando esta recta que une ambos puntos como una recta de largo  $l$  siendo  $x$  igual a  $(x_{i+1} - x_i)$  y siendo  $y$  la cantidad  $(y_{circ_{i+1}} - y_{circ_i})$ , se tiene un triángulo rectángulo tal que

$$l^2 = x^2 + y^2$$

lo que se quiere averiguar es si la mitad de la distancia de  $x$ , que viene , que es la que almacena el vector  $x_{medio}$  coincide con la mitad de la distancia  $y$ . Para ello

$$\frac{l^2}{2^2} = \frac{x^2 + y^2}{2^2}$$

Es decir

$$(l/2)^2 = (x/2)^2 + (y/2)^2$$

que viene a decir que el punto medio en la abscisa viene a coincidir con el punto medio de la recta y con el punto medio de la ordenada. Por esta razón cada elemento del vector  $y_{circmed}$  viene como sigue

$$y_{circmed} = \frac{y_{circ_i} + y_{circ_{i+1}}}{2}$$

c) *Angulo de inclinación  $\alpha$*

Para calcular el ángulo  $\alpha$  se hará lo que sigue

$$m_i = \frac{y_{circ_i} - y_{circ_{i+1}}}{x_i - x_{i+1}}$$
$$\alpha_i = \text{atan}(m_i)$$

La función que realiza esta operación se denomina *parámetros*. Esta función recibe como variables de entrada las variables geométricas del problema y las que se han calculado en la función *calculoextremos* y *divisorderebanadas*. Como variables de salida se tienen todas las que se han explicado en este apartado.

### 3.5 Método de Fellenius

---

El factor de seguridad en el método de Fellenius es

$$FS = \frac{\sum_1^n [c'_i \cdot l_i + (W_i \cdot \cos\alpha_i - u_i \cdot l_i) \cdot \tan\phi'_i]}{\sum_1^n W_i \cdot \sin\alpha_i}$$

Si se supone que la línea del nivel freático está lo suficientemente lejos o bien que la presión de agua intersticial es inexistente, entonces se puede anular el término correspondiente en la ecuación. Si además se supone que el terreno es homogéneo en todas las direcciones, la ecuación queda como sigue

**Ecuación 3-6**

$$\begin{aligned} FS &= \frac{\sum_1^n [c' l_i + W_i \cdot \cos\alpha_i \tan\phi']}{\sum_1^n W_i \sin\alpha_i} \\ &= \frac{c' \sum_1^n l_i + \tan\phi' \sum_1^n W_i \cos\alpha_i}{\sum_1^n W_i \sin\alpha_i} \end{aligned}$$

Teniendo en cuenta que

$$l_i = \frac{\text{paso}_i}{\cos\alpha_i}$$

El peso será el producto del área  $A_i$  de cada rebanada por el peso específico ( $\gamma$ ) del terreno, con lo cual

$$h_i = \gamma_{\text{talud}}_{med_i} - \gamma_{\text{circ}}_{med_i}$$

$$A_i = \text{paso vector}_i * h_i$$

$$W_i = \gamma * A_i$$

La descomposición perpendicular del peso es

$$N_i = W_i * \cos\alpha_i$$

Si a cada término  $i$  del numerador, correspondiente a cada rebanada, se denomina como  $num_i$  y se denomina con  $num$  a la suma de todos estos términos

$$num_i = c' * l_i + N_i * \tan\phi'$$

$$num = \sum_{i=1}^N num_i$$

Por otro lado, calcular el denominador es idéntico a calcular la proyección tangencial del peso, de la misma manera que se hizo para el numerador, se crea la variable  $den_i$

$$den_i = W_i \sin(\alpha_i)$$

Sumando todos los elementos  $den_i$  se obtiene la variable  $den$ .

$$den = \sum_{i=1}^N den_i$$

Finalmente

$$FS = \frac{num}{den}$$

La función que resuelve esto se denomina *mFelle*.

**Algoritmo 3-12: Funcion mFelle**

```

1  function [ FS,num,den ] = mFelle( y_talud_med,y_circ_med,alfa ,pasovector,gd,C,fi )
2  % Esta función calcula el FS por el método de Fellenius
16 -  l=pasovector./cosd(alfa);
17 -  % 2) Peso de cada rebanada
18 -  altura=y_talud_med-y_circ_med;
19 -  area=pasovector.*altura;
20 -  W=gd*area;
21 -  % 3) Calculamos Denominador (fuerzas desestabilizadoras (Stress))
22 -  T=W.*sind(alfa);      % descomposicion del peso(W) en la tangencial
23 -  den=sum(T);
24 -  % 4) Calculamos Numerador (fuerzas estabilizadoras (Strength))
25 -  cohesion=C*l;
26 -  N=W.*cosd(alfa);      % descomposicion del peso(W) en la normal
27 -  rozamiento=N*tand(fi);
28 -  num=sum(cohesion+rozamiento);
29 -  % 5) Calculamos el factor de seguridad (FS)
30 -  FS=abs(num/den);
31 - end

```

### 3.6 Método de Bishop simplificado

El factor de seguridad en el método de Bishop simplificado

$$FS = \frac{\sum_1^n \left\{ c'b_i + [(W_i - u_i b_i) \tan \phi'_i] \frac{\sec \alpha_i}{1 + \frac{\tan \phi'_i \tan \alpha_i}{FS}} \right\}}{\sum_1^n W_i \sin \alpha_i}$$

Se supondrán las mismas condiciones que en el método de Fellenius acerca del nivel freático y de la composición del suelo.

$$FS = \frac{\sum_1^n \left\{ c'b_i + \tan \phi' W_i \frac{\sec \alpha_i}{1 + \frac{\tan \phi'_i \tan \alpha_i}{FS}} \right\}}{\sum_1^n W_i \sin \alpha_i}$$

Teniendo en cuenta las siguientes relaciones trigonométricas

$$\sec \alpha_i = \frac{1}{\cos \alpha_i}; \sin \alpha_i = \tan \alpha_i \cdot \cos \alpha_i$$

Se llega a la siguiente ecuación

**Ecuación 3-7**

$$FS = \frac{c' \sum_1^n b_i + \tan \phi' \sum_1^n \frac{W_i}{\cos \alpha_i + \frac{\tan \phi' \cdot \sin \alpha_i}{FS}}}{\sum_1^n W_i \sin \alpha_i}$$

Como el factor de seguridad  $FS$  se encuentra también dentro de la ecuación, obliga a elaborar un método iterativo. Si el factor de seguridad se va almacenan en una variable  $FS$ , cuando se hayan realizado  $n$  iteraciones se tendrá

$$FS = [FS_0 \ FS_1 \ FS_2 \ \dots \ FS_j \ \dots \ FS_n]$$

Dada una tolerancia predefinida, el método finalizará cuando

$$abs(FS_n - FS_{n-1}) \leq \text{tolerancia}$$

El peso de cada rebanada y el denominador se calculan de forma idéntica que en el método de Fellenius.

Por otra parte, como el numerador de dicha ecuación es demasiado complejo para ser procesado del tirón, se hacen las siguientes sustituciones iniciales

$$B1_i = c'_i \cdot b_i; B2_i = W_i \cdot \tan \phi'_i; B3_i = \cos \alpha_i; B4_i = \tan \phi'_i \cdot \sin \alpha_i;$$

$$B5_i = B3_i + \frac{B4_i}{FS(i)}; B6_i = \frac{1}{B5_i};$$

Con lo cual se tiene que

$$num_i = (B1_i + B2_i) * B6_i$$

La variable  $num$  será

$$num = \sum_{i=1}^N num_i$$

Antes de proseguir hay que resaltar que el sumatorio de numerador alcanzado depende del FS que se haya utilizado en ese momento, es decir

$$num = num(FS_j)$$

con lo cual

$$FS_{j+1} = \frac{num(FS_j)}{den}$$

Un vector denominado FS irá almacenando todos los valores  $FS_j$  que se vayan calculando. El recuadro que aparece en el *algoritmo 3-11* muestra cómo se terminó de resolver el bucle iterativo que determina la convergencia de estas iteraciones

**Algoritmo 3-13: Función mBishop**

```

1  function [FS,num,den] = mBishop( y_talud_med,y_circ_med,alfa,pasovector,gd,C,fi )
2  % Esta función calcula el FS por el método de Bishop simplificado...
17 - altura=y_talud_med-y_circ_med;
18 - area=pasovector.*altura;
19 - W=gd*area;
20 - % 2) Calculamos Denominador (fuerzas desestabilizadoras (Stress))
21 - T=W.*sind(alfa);
22 - den=sum(T);
23 - % 3) Calculamos Numerador (fuerzas estabilizadoras (Strength)) y FSB
24 - B1=C*pasovector;B2=W*tand(fi);B3=cosd(alfa);B4=tand(fi)*sind(alfa);
25 - % Contador i=0 % Primer FS=1 % definimos toleranciaB1=1>0.001 para entrar
26 - i=0;FS(1)=1;toleranciaB=1;
27 - while toleranciaB>0.0001
28 -     i=i+1 ;
29 -     B5=B3+B4./FS(i) ;B6=1./B5;
30 -     num=sum((B1+B2).*B6);
31 -     FSB=abs(num/den);
32 -     % 4) Guardamos el FSB en un vector y calculamos el error
33 -     FS(i+1)=FSB; %#ok<AGROW> % Guardamos cada valor de FSB en el vector FS(i)
34 -     toleranciaB=abs(FS(i+1)-FS(i)); % Calculamos el error
35 - end
36 - % 5) Escogemos el último puesto como representante
37 - FS=FS(end); % Nos quedamos con el último FS
38 - end

```

La función que resuelve este método se denomina *mBishop*. (Algoritmo 3-13)

### 3.7 Método de Morgenstern-Price

En el marco teórico se desarrolló un método para implementar el método de Morgenstern-Price (Zhu, Lee, Qian, & Chen, 2005). La ecuación para calcular el *FS* viene expresada por la Ecuación 2-144

$$FS = \frac{\sum_{i=1}^{n-1} (R_i \prod_{j=i}^{n-1} \Psi_j) + R_n}{\sum_{i=1}^{n-1} (T_i \prod_{j=i}^{n-1} \Psi_j) + T_n}$$

La expresión para calcular el factor de corrección  $\lambda$  viene expresado por la Ecuación 2-155

$$\lambda = \frac{\sum_{i=1}^n [b_i(E_i + E_{i-1}) \tan \alpha_i]}{\sum_{i=1}^n [b_i(f_i E_i + f_{i-1} E_{i-1})]}$$

Además, se tiene también la Ecuación 2-1010

$$E_i \Phi_i = \Psi_{i-1} E_{i-1} \Phi_{i-1} + FS T_i - R_i$$

Los valores de la ecuación anterior vienen dados por la Ecuación 2-111, la Ecuación 2-122 y la Ecuación 2-133

$$\Phi_i = [(\sin \alpha_i - \lambda f_i \cos \alpha_i) \tan \phi'_i + (\cos \alpha_i + \lambda f_i \sin \alpha_i) FS]$$

$$\Phi_{i-1} = [(\sin \alpha_{i-1} - \lambda f_{i-1} \cos \alpha_{i-1}) \tan \phi'_{i-1} + (\cos \alpha_{i-1} + \lambda f_{i-1} \sin \alpha_{i-1}) FS]$$

$$\Psi_{i-1} = [(\sin \alpha_i - \lambda f_{i-1} \cos \alpha_i) \tan \phi'_i + (\cos \alpha_i + \lambda f_{i-1} \sin \alpha_i) FS] / \Phi_{i-1}$$

Además se tiene la Ecuación 2-88 y la Ecuación 2-99

$$R_i = [W_i \cos \alpha_i - U_i] \tan \phi'_i + c'_i b_i \sec(\alpha_i)$$

$$T_i = W_i \sin \alpha_i$$

Por otra parte, en el marco teórico se propuso el siguiente método para implementar dichas ecuaciones

**MP1.** Dividir la superficie de deslizamiento en un número  $N$  de rebanadas

**MP2.** Calcular  $R_i$  y  $T_i$  usando la Ecuación 2-88 y la Ecuación 2-99 respectivamente para cada rebanada

**MP3.** Especificar la función  $f(x)$

**MP4.** Dar valores iniciales a  $FS$  y  $\lambda$ .

**MP5.** Calcular  $\Phi_i$  y  $\Psi_{i-1}$  usando la Ecuación 2-111 y la Ecuación 2-123 para todas las rebanadas.

**MP6.** Calcular FS usando la Ecuación 2-144.

**MP7.** Repetir el paso MP5 y MP6 para mejorar los valores  $FS, \Phi_i$  y  $\Psi_{i-1}$ .

**MP8.** Calcular  $E_i$  usando la Ecuación 2-1010 para cada rebanada.

**MP9.** Calcular  $\lambda$  usando la Ecuación 2-155.

**MP10.** Con los nuevos valores de  $FS$  y  $\lambda$  volver al paso tres y repetir el algoritmo hasta que las diferencias de  $FS$  y  $\lambda$  entre dos valores consecutivos sean más pequeñas que las tolerancias  $\varepsilon_1$  y  $\varepsilon_2$  respectivamente.

Se procede ahora a la implementación del método de MP usando de apoyo las reglas propuestas.

**MP1.** Dividir la superficie de deslizamiento en un número  $N$  de rebanadas. Esta parte se llevará a cabo con las funciones *calculoextremos* y la función *parámetros*.

**MP2.** Calcular  $R_i$  y  $T_i$  usando usando la Ecuación 2-88 y la Ecuación 2-99 respectivamente para cada rebanada

Se calculan estos dos valores usando sus ecuaciones respectivas

$$R_i = W_i \cos \alpha_i \tan \phi'_i + c'_i b_i \sec(\alpha_i) \quad i = [1, 2, \dots, N - 1, N]$$

$$T_i = W_i \sin \alpha_i \quad i = [1, 2, \dots, N - 1, N]$$

El código que resuelve este apartado es el mostrado en el algoritmo 3-14

**Algoritmo 3-14: Subproceso de la función mMorgPri**

```
%% Calculo de R,T
altura=y_talud_med-y_circ_med;
area=pasovector.*altura;
W=gd*area;
% R
N=W.*cosd(alfa);
rozamiento=N*tand(fi);
cohesion=C*pasovector.*secd(alfa);
R=rozamiento+cohesion;
% T
T=W.*sind(alfa);
```

Se obtendrán dos vectores tal que

$$R = [R_1 \ R_2 \dots \ R_i \dots \ R_{n-1} \ R_n]$$

$$T = [T_1 \ T_2 \dots \ T_i \dots \ T_{n-1} \ T_n]$$

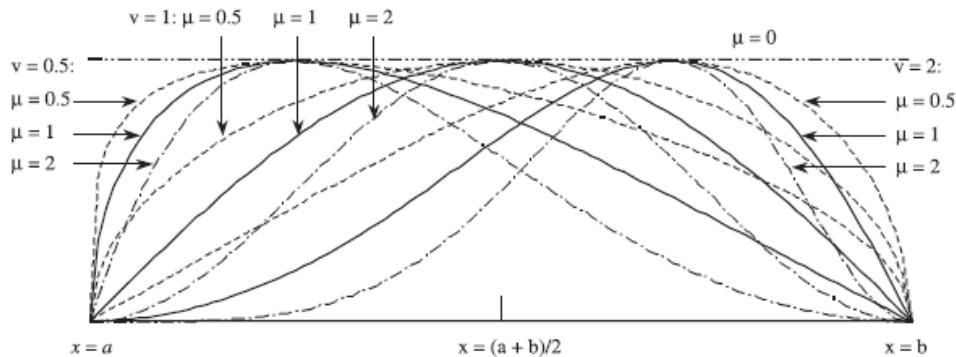
**MP3.** Especificar la función  $f(x)$

La función que se propone en el texto que se está utilizando como referencia (Zhu, Lee, Qian, & Chen, 2005) es

$$f(x) = \sin^\mu \left[ \pi \left( \frac{x-a}{b-a} \right)^\nu \right]$$

En donde  $a$  y  $b$  son las abscisas extremas de la superficie de deslizamiento que cortan con el talud;  $\mu$  y  $\nu$  son dos valores no negativos, tal que  $\mu = 0 - 5.0; \nu = 0.5 - 2$

Ilustración 3-17: Familia de funciones half-sine



De esta familia de funciones, habrá que escoger una de ellas para aplicar el método. Esta función se utiliza para definir las fuerzas inter-rebanadas, estará definida sobre los extremos de las rebanadas y por ello su dimensión será  $n+1$ . Es decir, siendo  $f_i$  el valor de la función en cada punto  $i$  inter-rebanada, se tendrá un vector  $f$ , que contiene a todos los elementos  $f_i$  tal que

$$f = [f_0 \ f_1 \ f_2 \dots \ f_i \dots \ f_{n-1} \ f_n]$$

El algoritmo 3-15 muestra la implementación de esta función

**Algoritmo 3-15: Subprocesos de la función mMorgPri**

```

14 %% Elección de f(x)
15 - mu=1;      %0-5
16 - uve=1;     %0.5-2
17 - aa=x(1);
18 - bb=x(end);
19 - argumento=pi*((x-aa)/(bb-aa)).^uve;
20 - medioseno=(sin(argumento)).^mu;
21 - f=medioseno; %f(1)=f0,f(2)=f1,f(i)=fi-1,f(n)=f(n-1)

```

**MP4.** Dar valores iniciales a  $FS$  y  $\lambda$ .

La elección de los valores iniciales tendrán efectos sobre el número de iteraciones necesarios para converger, pero no tendrá efecto sobre el valor en el cual converge (*Morgenstern and Price ,1965*). En general se asumirán los siguiente valores

$$FS = 1 \text{ y } \lambda = 0;$$

**MP5.** Calcular  $\Phi_i$  y  $\Psi_{i-1}$  usando la Ecuación 2-111y la Ecuación 2-123 para todas las rebanadas  $f_i(\Phi_i)$

Para implementar estas dos magnitudes hay que tener especial cuidado con las dimensiones de los vectores. Observando la Ecuación 2-111

$$\Phi_i = [(\sin \alpha_i - \lambda f_i \cos \alpha_i) \tan \phi + (\cos \alpha_i + \lambda f_i \sin \alpha_i) FS]$$

se observa que  $\Phi_i = \Phi_i(\alpha_i, f_i, \phi', \lambda, FS)$ . Los parámetros  $\phi, \lambda, FS$  son constantes a lo largo del talud. Las magnitudes  $\alpha_i, f_i$  varían para cada rebanada,  $f_i$  está definido para cada nodo M, y por ello el primer elemento del vector  $f$  no es  $f_1$  sino  $f_0$ . Por esta razón se creará vector  $fsinf_0$  tal que

$$fsinf_0 = [f_1 \ f_2 \ \dots \ f_i \ \dots \ f_n]$$

siendo ahora coherente la expresión

$$\Phi_i = \Phi_i(\alpha_i, fsinf_0, \phi', \lambda, FS).$$

finalmente se obtiene la expresión que se busca

$$FI = [\Phi_1 \ \Phi_2 \ \dots \ \Phi_i \ \dots \ \Phi_{n-1} \ \Phi_n]$$

### 1. Psi ( $\Psi_{i-1}$ )

Observando la Ecuación 2-12

$$\Psi_{i-1} = [(\sin \alpha_i - \lambda f_{i-1} \cos \alpha_i) \tan \phi' + (\cos \alpha_i + \lambda f_{i-1} \sin \alpha_i) FS] / \Phi_{i-1}$$

se tiene que  $\Psi_{i-1} = \Psi_{i-1}(\alpha_i, f_{i-1}, \Phi_{i-1}, \phi', \lambda, FS)$ . Como antes,  $\phi', \lambda, FS$  son constantes en todas las rebanadas y los valores que varían son  $\alpha_i, f_{i-1}, \Phi_{i-1}$ . Para  $i=1$  se tendría  $\Psi_0 = (\alpha_1, f_0, \Phi_0, \phi', \lambda, FS)$ . Sin embargo, como ya se ha explicado, el vector  $\Phi$  comienza en  $\Phi_1$ . Con lo cual, el vector  $\Psi_{i-1}$  debe de comenzar en  $i=2$ . Por esta razón se hace el siguiente arreglo en el vector `teta_med`.

$$tmsint_1 = [\alpha_2 \alpha_3 \dots \alpha_i \dots \alpha_n]$$

Por otro lado, para  $i=n$  se tiene  $\Psi_{n-1} = (\alpha_n, f_{n-1}, \Phi_{n-1}, \phi', \lambda, FS)$ . Sin embargo,

$$f = [f_0 \ f_1 \ \dots \ f_n]; \Phi = [\Phi_1 \ \dots \ \Phi_n]$$

Para poder continuar con el método habrá que crear los siguientes vectores

$$fsinfn = [f_1 \ \dots \ f_{n-1}]; FIsinFIn = [\Phi_1 \ \dots \ \Phi_{n-1}]$$

Con lo cual se tiene que

$$\Psi_{i-1} = \Psi_{i-1}(\alpha_i, fsinfn_{n-1}, FIsinFIn_{i-1}, \phi', \lambda, FS).$$

Obteniendo finalmente un vector tal que

$$PSI = [\Psi_1 \ \Psi_2 \ \dots \ \Psi_{n-1}]$$

**MP6.** Calcular FS usando la Ecuación 2-144.

$$FS = \frac{\sum_{i=1}^{n-1} (R_i \prod_{j=i}^{n-1} \Psi_j) + R_n}{\sum_{i=1}^{n-1} (T_i \prod_{j=i}^{n-1} \Psi_j) + T_n}$$

desarrollando esta se tiene que

$$FS = \frac{R_1[\Psi_1\Psi_2\Psi_3\Psi_4 \dots \Psi_{n-1}] + R_2[\Psi_2\Psi_3 \dots \Psi_{n-1}] + \dots + R_{n-1}[\Psi_{n-1}] + R_n}{T_1[\Psi_1\Psi_2\Psi_3 \dots \Psi_{n-1}] + T_2[\Psi_2\Psi_3 \dots \Psi_{n-1}] + \dots + T_{n-1}[\Psi_{n-1}] + T_n}$$

**MP7.** Repetir el paso MP5 y MP6 para mejorar los valores  $FS$ ,  $\Phi_i$  y  $\Psi_{i-1}$

El Algoritmo 3-166 muestra lo que se ha explicado.

**Algoritmo 3-16: Subproceso de la función ZhuLeeChen**

```

for j=1:2
    % FI (FI1....Fin)
    fsinfo=f;fsinfo(1)=[];
    FI=(sind(alfa)-lambda(end)*fsinfo.*cosd(alfa))...
        *tand(fi)+(cosd(alfa)+lambda(end)*fsinfo...
        .*sind(alfa))*FS(end);
    % PSI (PSI1....PSI(n-1))
    fsinfofn=fsinfo; fsinfofn(end)=[];
    asint1=alfa;asint1(1)=[];
    FIisinfIn=FI;FIisinfIn(end)=[];

    PSI=((sind(asint1)-lambda(end)*fsinfofn.*cosd(asint1))...
        *tand(fi)+(cosd(asint1)+lambda(end)...
        .*fsinfofn.*sind(asint1))*FS(end)./FIisinfIn;
    % FS
    N=length(alfa); %N rebanadas;
    numi=zeros(1,N-1);deni=zeros(1,N-1);
    for i=1:N-1
        prodPSI=prod(PSI(i:end));
        % numerador
        numi(i)=R(i)*prodPSI;
        % denominador
        deni(i)=T(i)*prodPSI;
    end
    num=sum(numi)+R(end);den=sum(deni)+T(end);
    FS(contadorFS)=num/den;
end

```

**MP8.** Calcular  $E_i$  usando la Ecuación 2-1010 para cada rebanada.

La Ecuación 2-1010 permite calcular los valores de  $E_i$

$$E_i = \frac{\Psi_{i-1}E_{i-1}\Phi_{i-1} + FS T_i - R_i}{\Phi_i}$$

se puede ver que

$$E_i = E(\Psi_{i-1}, E_{i-1}, \Phi_{i-1}, \Phi_i, FS, T_i, R_i).$$

para  $i=1$ ,

$$E_1 = \frac{\Psi_0 E_0 \Phi_0 + FS T_1 - R_1}{\Phi_1}$$

en principio no hay valores para  $\Psi_0, \Phi_0$  pero si se tiene en cuenta que  $E_0 = 0$ , entonces se tendrá que

$$E_1 = \frac{FS T_1 - R_1}{\Phi_1}$$

con lo cual

$$E_1 = E(\Phi_1, FS, T_1, R_1).$$

Que como se ve no depende de  $\Psi_0, \Phi_0$ . Por otro lado, para  $i=n$

$$E_n = 0$$

después de esto se tendrá

$$E = [E_0 \ E_1 \ E_2 \ \dots \ E_i \ \dots \ E_{n-1} \ E_n] = [0 \ E_1 \ E_2 \ \dots \ E_i \ \dots \ E_{n-1} \ 0]$$

sin embargo, por comodidad el vector E estará definido como sigue

$$E = [E_1 \ E_2 \ \dots \ E_i \ \dots \ E_{n-1}]$$

#### Algoritmo 3-17: Subproceso de la función ZhuLeeChen

```
% E (E1...En-1)
E=zeros(1,N-1);
E(1)=FS(end)*T(1)-R(1)/FI(1);

for i=2:N-1
    E(i)=(PSI(i-1)*E(i-1)*FIsinFIN(i-1)+FS(end)*T(i)-R(i))/FI(i);
end
```

**MP9.** Calcular  $\lambda$  usando la Ecuación 2-155.

$$\lambda = \frac{\sum_{i=1}^n [b_i(E_i + E_{i-1}) \tan \alpha_i]}{\sum_{i=1}^n [b_i(f_i E_i + f_{i-1} E_{i-1})]}$$

desarrollando para cada término se tiene que

$$\lambda = \frac{b_1(E_1 + E_0) \tan \alpha_1 + \dots + b_i(E_i + E_{i-1}) \tan \alpha_i + \dots + b_n(E_n + E_{n-1}) \tan \alpha_n}{b_1(f_1 E_1 + f_0 E_0) \tan \alpha_1 + b_i(f_i E_i + f_{i-1} E_{i-1}) \tan \alpha_{i-1} + \dots + b_n(f_n E_n + f_{n-1} E_{n-1}) \tan \alpha_n}$$

Para calcular este valor de manera rápida y concisa se crearán los siguientes vectores para el numerador

$$Eimenes1 = [0 \ E] = [0 \ E_1 \ E_2 \ \dots \ E_{n-1}]$$

$$Ei = [ \ E \ 0] = [ \ E_1 \ E_2 \ \dots \ E_{n-1} \ 0]$$

$$Enum = [Ei + Eimenes1] = [E_1 + E_0, E_2 + E_1, \dots, E_i + E_{i-1}, \dots, E_n + E_{n-1}]$$

Y para el denominador

$$fsinfnEimenes1 = [0 \ fsinfnEimenes1] = [0 \ f_1E_1 \ f_2E_2 \ \dots \ f_{n-1}E_{n-1}]$$

$$fsinf0Ei = [ \ Ei \ fsinf0 \ 0] = [ \ f_1E_1 \ f_2E_2 \ \dots \ f_{n-1}E_{n-1} \ 0]$$

$$\begin{aligned} fEden &= [fsinfnEimenes1 + fsinf0Ei] \\ &= [f_1E_1 + f_0E_0, f_2E_2 + f_1E_1, \dots, f_iE_i + f_{i-1}E_{i-1}, \dots, f_nE_n + f_{n-1}E_{n-1}] \end{aligned}$$

Con estos vectores se puede implementar fácilmente la ecuación 2-15 tal y como muestra el *algoritmo 3-18*

**Algoritmo 3-18: Subprocesos de la función ZhuLeeChen**

```
%Lambda
%Adaptamos E a la ecuación de lambda. Eo y En =0.
fsinfn=f;
fsinfn(end)=[];
Eimenes1=[0 \ E \ 0];
Enum=Ei+Eimenes1;
fsinfnEimenes1=Eimenes1.*fsinfn;fsinfoEi=Ei.*fsinfo;
fEden=fsinfoEi+fsinfnEimenes1;

contadorlambda=contadorlambda+1;

lambdanum=sum(pasovector.* (Enum).*tand(teta_med));
lambdad=+sum(pasovector.*fEden);
lambda(contadorlambda)=lambdanum/lambdad;
```

**MP10.** Con los nuevos valores de  $FS$  y  $\lambda$  volver al paso tres y repetir el algoritmo hasta que las diferencias de  $FS$  y  $\lambda$  entre dos valores consecutivos sean más pequeñas que las tolerancias  $\varepsilon_1$  y  $\varepsilon_2$  respectivamente.

Se usarán los mismos valores que los propuestos en el artículo que se está siguiendo (Zhu, Lee, Qian, & Chen, 2005)

$$\varepsilon_1 = 0.0001 \text{ y } \varepsilon_2 = 0.0001$$

Todos los valores de  $Fs$  y  $\lambda$  se irán incorporando en un vector tal que  $Fs_0 = 1$  y  $\lambda_0 = 0$ .

$$Fs = [1 \ Fs_1 \ Fs_2 \ ... \ Fs_{n-1} \ Fs_n]$$

$$\lambda = [0 \ \lambda_1 \ \lambda_2 \ ... \ \lambda_{n-1} \ \lambda_n]$$

cumpliéndose que

$$Fs_n - Fs_{n-1} \leq \varepsilon_1 \quad y \quad \lambda_n - \lambda_{n-1} \leq \varepsilon_2$$

En el método original propuesto por Morgenstern y Price (*Morgenstern and Price, 1965*) se advierte que algunos puntos no terminan de converger, ocasionando problemas serios a la hora de buscar el mínimo factor de seguridad en un marco. La recomendación que dieron fue la de probar el método con otra función distinta a la usada buscando así la convergencia. Por otra parte, en el mismo documento citado se menciona que, el 80% de los valores que tomaron de prueba convergían en menos de 10 iteraciones.

En base a esto se llevarán a cabo las distintas acciones.

- 1) Si el método no converge en menos de 20 iteraciones se comenzará de nuevo cambiando los parámetros de la función
- 2) La acción primera se repetirá un número determinado de veces, si después de esto no se ha encontrado convergencia, entonces se dirá que esa circunferencia no tiene asociado un FS.

Se concluye de esta forma la implementación del método de Morgernsten-Price. Para implementar con éxito estas reglas han hecho falta dos funciones. La función con menos jerarquía es `[FS,lambda,num,den] = ZhuLeeChen(alfa,pasovector,fi,R,T,f)` que se encarga de llevar a cabo los pasos **MP5, MP6, MP7, MP8, MP9, MP10**. La función `[FS,lambda] = mMorgPri( x,y_talud_med,y_circ_med,alfa ,pasovector,gd,C,fi )` lleva a cabo los pasos **MP1, MP2, MP3, MP4** y además introduce dentro de un bucle `for` a la función `ZhuLeeChen`.

## 4. Implementación de los algoritmos de búsqueda

---

### 4.1 Factor de seguridad en una superficie de deslizamiento

---

#### 4.1.1 Variables necesarias

---

Se han creado distintas funciones que hasta ahora son inconexas entre sí. Se calculan los extremos del talud con la función *calculoextremos*, luego se hace una división en  $N$  rebanadas y finalmente se aplica el método que se desea a través de las funciones *mFelle*, *mBishop*, *mMorgPri*. Para que esto tenga sentido es necesario la creación de una función que permita el flujo de información entre estas funciones. Esta función será *FSP*. La explicación de esta función será conjuntamente con la aplicación del programa al caso más sencillo, que es el cálculo del factor de seguridad en un punto determinado del espacio y con un radio determinado. En principio las variables necesarias, externas a los cálculos internos, es decir, las que debe de facilitar el usuario serán:<sup>2</sup>

- Variables geométricas: *Altura del talud (H)* y *Base del talud (B)*.
- Variables características del terreno: *Cohesión (C)*, *peso específico (gd)* y *ángulo de rozamiento (fi)*.
- Circunferencia de deslizamiento: *Coordinada del centro en x (a)*, *coordenada del centro en y (b)* y *Radio (R)*.
- Métodos: Número de rebanadas a usar (rebanadas) y el método a usar

Una vez se han recogido estos datos y se dispone de las variables mencionadas se crean dos nuevas variables de caracteres. Estas son *Metodo* y *Met*. La primera es una cadena que puede contener lo siguiente

$$\text{Metodo} = \{ 'mFelle', 'mBishop', 'mMorgpri' \}$$

como se observa contiene los nombres de las distintas funciones de los distintos métodos, según se haya elegido en el paso cuatro. La variable *Met* contiene el resto del título de la

---

<sup>2</sup> Se han colocado entre paréntesis cómo aparecen estas variables en el código fuente

función (esto dependerá del método que se haya seleccionado). De tal forma que combinando las variables *método* y *Met* el código selecciona la función de equilibrio límite requerida por el usuario. A continuación se muestra en pseudocódigo cómo funcionan en conjunto, todas las funciones que se han explicado. Cada vez que se calcule el FS sobre un punto, necesariamente tendrán que pasar por esta función.

**Algoritmo 4-1: Función FSP**

```
1  function [ FS,xti,xtd,k4,lambda] = FSP( a,b,R,C,gd,fi,B,H,rebanadas,Met,Metodo)
2  %
3  % [ xti,xtd,k4,r] = calculoextremos(a,b,B,H,R);
4  %
5  % if k4==0 % No hay superficie de deslizamiento válida
6  %    FS=nan;
7  %    lambda=nan;
8  %
9  % else
10 %    [ x,x_medio,y_talud_med ,y_circ_med,alfa, ...
11 %      pasovector,y_talud,y_circ ] = parametros( rebanadas, ...
12 %      xti,xtd,B,H,a,b,R,r); %#ok<*ASGLU,*NASGU>
13 %    switch Metodo
14 %      case 'mMorgPri'
15 %        eval(Met);
16 %      case {'mFelle','mBishop'}
17 %        eval(Met);lambda=nan;
18 %    end
19 % end
20 % end
```

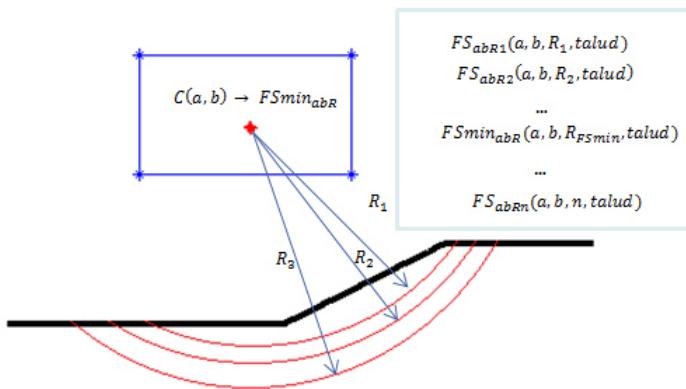
## 4.2 Factor de seguridad variando el radio sobre un punto

---

### 4.2.1 Introducción

El primer paso para poder hacer la búsqueda de un *FS mínimo* en un entorno de búsqueda es situarnos sobre un punto e ir calculando los distintos *FS* para distintos radios sobre ese punto. Todos los *FS* que se van obteniendo, y sus respectivos radios se irán guardando en un vector llamado *FSR*. Una vez hecho esto se buscará sobre este vector, el mínimo *FS* almacenándose en el vector *FSminR*, con su respectivo radio.

Ilustración 4-1: Distintos FS variando el radio



En la ilustración 4-1 se observa un talud con un marco azul sobre el que representa la zona de búsqueda. El punto marcado en rojo representa el punto sobre el que se están realizando simulaciones. Relacionados con este punto hay tantos factores de seguridad como radios se puedan simular. Como se ha dicho, de todos estos factores de seguridad, se desea obtener aquel que es mínimo. Cada punto del espacio de búsqueda está relacionado con un único FS, el cual es el mínimo entre todos los FS que se le pueden calcular.

La forma más básica de realizar este proceso sería iniciar la búsqueda en un radio  $R_1$  determinado e ir haciendo crecer el radio con un paso  $pr$  hasta llegar a un radio  $R_2$  previamente fijados (ver Algoritmo 4-2)

Algoritmo 4-2: Primera propuesta para la función FSRMinSinOpt

```

1  Funcion FSRmin <- FSRMinSinOpt_1(Rinicial,Rfinal,pr,a,b,otras)
2      j=0; //contador de radios/
3      Para R<-Rinicial Hasta Rfinal Con Paso pr Hacer
4          j=j+1
5          // Se invoca a la función FSP para calcular el FS en a,b,R
6          // y se guarda en el vector FSR(j)
7          FSR(j)<-FSP(R,a,b,otras variables)
8      FinPara
9      // Se busca el menor FS del vector FSR(j)
10     FSRmin <- min(FS,FSR(j))
11  FinFuncion
12

```

Esta forma de hacer las iteraciones no es robusta, en el sentido de que no se puede saber si  $R_2$  es lo suficientemente grande como para haber pasado por encima del mínimo. No es cómodo porque el usuario tiene que elegir los radios mínimo y máximo cada vez que quiera iterar sobre un punto o peor aún, elegir un radio inicial y final igual para todos los puntos. Y tampoco es eficiente, ya que es posible que  $R_1$  no sea lo suficientemente grande como para atravesar el talud, y se hagan simulaciones que están forzadas por lógica a dar un resultado que de antemano se sabe no tiene solución factible.

Por esta razón se ha dedicado esfuerzo en mejorar las condiciones algorítmicas de este problema. A continuación se expondrán dos simulaciones en las que tratará de ilustrar lo explicado.

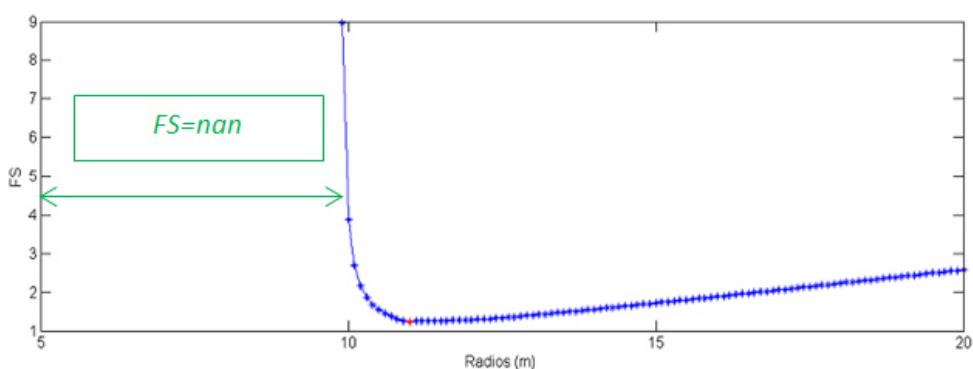
Se escogen dos puntos  $PA$  y  $PB$  y se aumentará el radio a un paso  $pr$  desde  $R1$  hasta  $R2$ . Luego se mostrarán los resultados en una gráfica y se analizarán.

#### *Simulación 4.1*

Tabla 4-1: Valores de la simulación 4.1

Talud 1				
Centro(m)		Radios (m)		
$a$	$b$	$R_1$	$R_2$	$pr$
0	1	5	20	0.5

Ilustración 4-2: Resultados de la simulación 4.1



En esta *simulación* se puede observar que los valores de  $FS$  para los radios que van desde  $R=5$  hasta  $R=10$  (aprox.) son  $FS=nan$ . Por esta razón no hay valores para ellos en la gráfica.

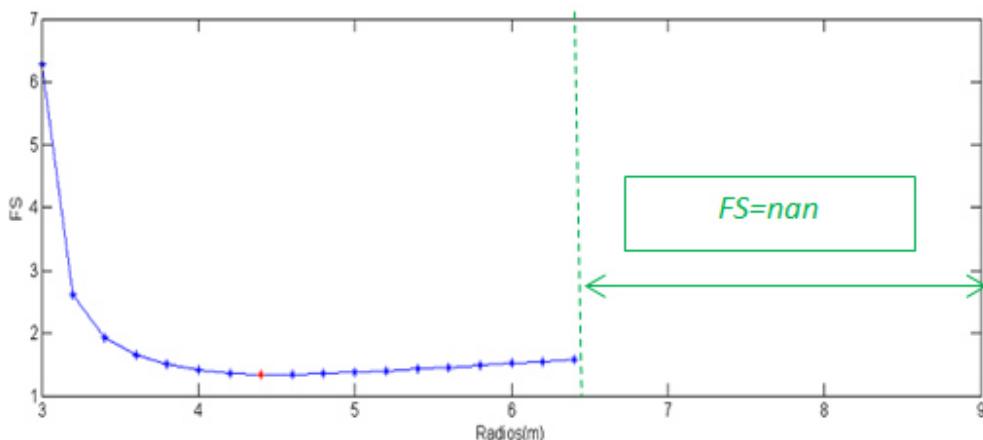
Con lo cual todas las iteraciones realizadas entre estos dos valores son innecesarias. Surge la necesidad de calcular la distancia mínima que debe de tener el radio para comenzar a simular sin incurrir en este problema.

### Simulación 4.2

**Tabla 4-2: Valores de la simulación 4.2**

<b>Talud 1</b>				
<b>Centro(m)</b>		<b>Radios (m)</b>		
<i>a</i>	<i>b</i>	$R_1$	$R_2$	<i>pr</i>
1	4	3	9	0.2

**Ilustración 4-3: Resultados de la simulación 4.2**



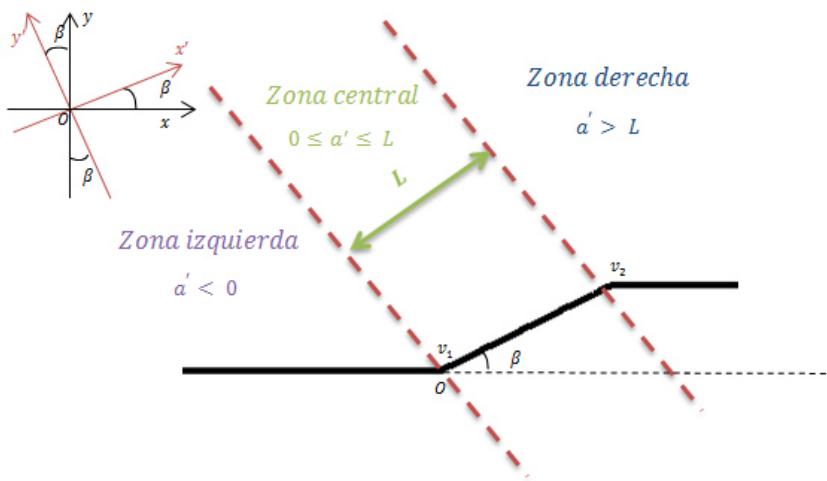
En esta simulación el radio va aumentando hasta toparse con una situación en la deja de producir soluciones factibles, por tratarse de una superficie de deslizamiento no válida. Este último problema solo se da en las situaciones en las que la circunferencia se sitúa por debajo del rasante superior del talud.  $b < H$ .

En vista de estas dos simulaciones mostradas, se propone la creación de una función que calcule la distancia más corta entre el punto que se esté evaluando y la primera circunferencia que se pueda considerar superficie de deslizamiento y otra función que calcule el radio máximo que se debería de tener en cuenta para dejar de iterar, siempre y cuando  $b < H$ .

#### 4.2.2 Radio mínimo

Para calcular la distancia mínima se procede como sigue. Se divide el espacio en tres zonas, tal y como muestra la imagen.

Ilustración 4-4: Cálculo del radio mínimo I



Se quiere averiguar en qué zona (ver Ilustración 4-4) se encuentra la circunferencia a partir del conocimiento de su centro. Si se giran los ejes girados un ángulo *beta* en sentido anti-horario sería muy sencillo averiguar dónde se encuentra dicho centro.

Si los ejes hasta ahora son  $OXY$ , entonces los nuevos ejes serán  $OX'Y'$ . Un punto representado en los ejes  $OXY$  viene dado por el punto  $P(x,y)$ . Un punto representado en los nuevos ejes vendrá dado por  $P'(x',y')$ . Conociendo el punto  $P(x,y)$  se desea conocer el punto  $P'(x',y')$ . Para ello se utilizará la matriz de transformación de ejes de tal forma que

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

El centro de la circunferencia viene dado por el punto  $C(a,b)$ , en los ejes  $OX'Y'$  vendrá dado por  $C'(a',b')$ . Usando la matriz de transformación se obtiene

$$\begin{pmatrix} a' \\ b' \end{pmatrix} = \begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix};$$

$$a' = a\cos\beta + b\sin\beta; b' = -a\sin\beta + b\cos\beta$$

es decir

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = \begin{bmatrix} a\cos\beta + b\sin\beta \\ -a\sin\beta + b\cos\beta \end{bmatrix}$$

Para saber en qué zona se encuentra el punto simplemente se evaluará la coordenada  $a'$  de tal forma que

- a. Si  $a' < 0$ , la circunferencia está en la zona izquierda
- b. si  $0 \leq a' \leq v_2$ , la circunferencia está en la zona central
- c. si  $a' > v_2$ , la circunferencia está en la zona derecha

Para simplificar esto, se procede a calcular  $v_2$  en los nuevos ejes

El vértice superior del talud en los ejes originales viene dado por el punto  $v_2$  ( $B, H$ ) en los ejes  $OX'Y'$  vendrá dado por  $v_2'$  ( $L, S$ ) '.

Usando la matriz de transformación

$$\begin{pmatrix} L \\ S \end{pmatrix} = \begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} B \\ H \end{pmatrix};$$

Desarrollando

$$\begin{aligned} L &= B\cos\beta + H\sin\beta \\ B &= \sqrt{B^2 + H^2} \cos\beta; H = \sqrt{B^2 + H^2} \sin\beta \end{aligned}$$

Sustituyendo

$$L = \sqrt{B^2 + H^2} [(\cos\beta)^2 + (\sin\beta)^2]$$

Es decir

$$L = \sqrt{B^2 + H^2}$$

Y por otro lado

$$S = -B\sin\beta + H\cos\beta$$

Utilizando el mismo razonamiento

$$S = -\sqrt{B^2 + H^2} \cos\beta \sin\beta + \sqrt{B^2 + H^2} \sin\beta \cos\beta = 0$$

Con lo cual

$$\nu_2 = \begin{bmatrix} L \\ S \end{bmatrix} = \begin{bmatrix} \sqrt{B^2 + H^2} \\ 0 \end{bmatrix}$$

Con lo cual, el conjunto de reglas que se dieron antes quedan ahora

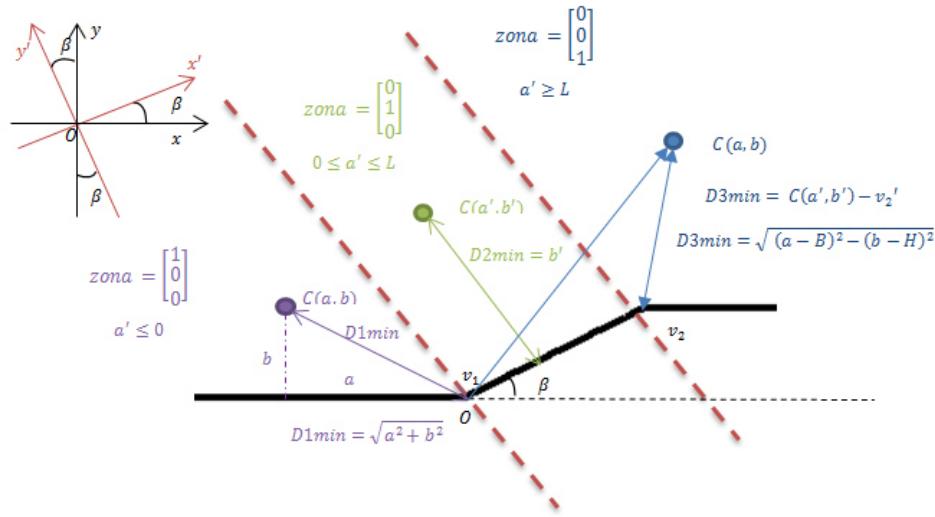
- a. Si  $a' < 0$ , la circunferencia está en la zona central izquierda
- b. si  $0 \leq a' \leq L$ , la circunferencia está en la zona central
- c. si  $a' > L$ , la circunferencia está en la zona derecha

A continuación, se crea el siguiente vector columna que almacena con un uno lógico la zona en la que se encuentra la circunferencia

$$\text{zona} = \begin{bmatrix} a' < 0 \\ 0 \leq a' \leq L \\ a' > L \end{bmatrix}$$

Se recuerda que todo esto se está haciendo para averiguar cuál es el radio mínimo a partir del cual se produce una iteración útil. Se han definido distintas zonas en el plano para ahora poder definir con sencillez la distancia mínima que hay desde un punto cualquiera del espacio hasta el primer radio que produce una simulación útil.

Ilustración 4-5: Cálculo del radio mínimo II



Dicho esto, se observa que dependiendo de la zona en la que se encuentra el punto, se definirá una distancia mínima.

Para cualquier punto situado en la zona exterior izquierda, la distancia mínima que debería de aumentar el radio para empezar a generar superficies de deslizamiento es la recta que hay entre el mismo punto y el vértice inferior izquierdo del talud, es decir, el origen de coordenadas. Es decir, el radio deberá ser mayor que esta distancia, la cual se denominará como  $D_1$  de tal forma que

$$D_1 = \sqrt{a^2 + b^2}$$

La zona derecha se calcula de manera similar que la zona izquierda, solo que cambiando de vértice. La distancia más corta se produce cuando intersecta al vértice derecho, llamando a esta distancia  $D_3$

$$D_3 = \sqrt[2]{(a-B)^2 + (b-H)^2}$$

Y finalmente la zona central, para los puntos que están en esta zona la distancia será  $D_2$ . Cuando se está en esta situación, la distancia más corta que existe entre el centro  $C(a',b')$  y la recta  $k_1$  será la dada por la recta que, siendo perpendicular a  $k_1$ , pasa por el centro  $C(a',b')$ . Una de las razones por las que se hizo la operación de rotación de ejes fue para que esta recta coincidiera exactamente con la coordenada  $b'$ , de tal suerte que

$$D_2 = b'$$

Para terminar se creará un vector tal que

$$D_{vect} = [D_1 \ D_2 \ D_3]$$

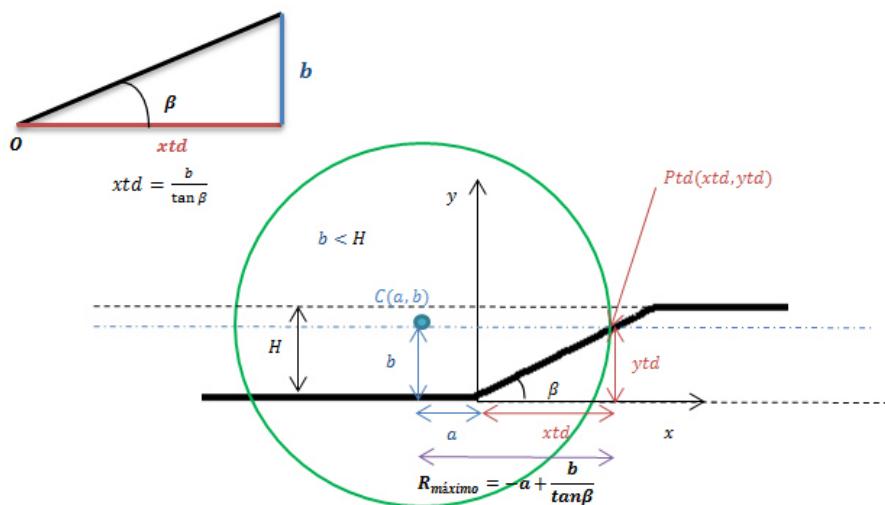
Finalmente, para garantizar que el primer punto sobre el que se realiza una iteración, -que es el objetivo de todo lo que se ha explicado en este apartado- no sea en vano, el radio  $R$  deberá ser mayor –o igual en el caso límite- a

$$R_{minimo} \geq D_{vect} * zona = [D_1 \ D_2 \ D_3] \begin{bmatrix} a' < 0 \\ 0 \leq a' \leq L \\ a' > L \end{bmatrix}$$

#### 4.2.3 Radio máximo para centros menores a la altura del talud

Aquellas circunferencias con centro  $C(a,b)$  que se encuentren en la situación de  $b < H$  se les podrá calcular un  $R$  máximo, a partir del cual comenzarán a generar iteraciones inútiles.

Ilustración 4-6: Cálculo del radio máximo cuando  $b < H$



Analizando la Ilustración 4-6 y relacionando las distancias se podrá comprender que el radio máximo vendrá dado por la expresión

$$R_{2_{\maximo}} = -a + \frac{b}{\tan\beta}$$

#### 4.2.4 Implementación algorítmica

Todo esto se llevará a cabo a través de la función *distminR1*. Esta función recibe como entrada los datos geométricos del talud y del centro de la circunferencia y saca como variables de salida la distancia  $D_1$  al vértice  $v_1$ , la distancia  $D_2$  a la recta  $k_1$ , la distancia  $D_3$  al vértice  $v_2$ , la distancia  $R_{2_{\max}}$ , que era el radio máximo hasta el que puede iterar en caso de que  $b < H$  y finalmente la distancia  $D$ , que será  $D_1$ ,  $D_2$  o  $D_3$  según la zona en la que se encuentre. Ahora mismo, de los únicos que interesa saber el valor es del radio mínimo  $D$  y  $R_{2_{\max}}$ . Los valores  $D_1$ ,  $D_2$ ,  $D_3$  ahora mismo solo interesan dentro de la función. El valor de  $R_2$  en el caso que  $b > H$  lo debe de proporcionar el usuario. Este valor debe de ser lo suficientemente grande para garantizar que se esté pasando por encima del FS mínimo. No se le dará importancia porque esta cuestión es retomada más adelante y solucionada. El Algoritmo 4-3 realiza la operación que se ha explicado

**Algoritmo 4-3: Función distminR1**

```

1  Funcion [D,R2max] <-distminR1 (a,b,B,H)
2      // Previamente se giran los ejes y se hacen los calculos explicados
3      zona=[aprime<0; 0<=aprime<=L; aprime>=L];
4      D1 <- sqrt(b^2 +a^2);
5      D2 <-bprima;
6      D3 <-sqrt((a-B)^2 + (b-H)^2);
7      Dvect=[D1 D2 D3];
8      D <- Dvect*zona
9      R2max <-a+b/tan(beta);
10 FinFuncion
11

```

Ya se puede generar un algoritmo capaz de no realizar iteraciones absurdas al comienzo y frenarse de acuerdo a una lógica robusta, cuando  $b < H$ . Faltaría por ver si habría alguna manera de poder frenar el algoritmo cuando  $b > H$ , pero esto forma parte de un conjunto de reglas que se ha decidido incluir en el *capítulo 6*. Por ahora se está en posición de escribir el pseudocódigo original como sigue

**Algoritmo 4-4: Pseudocódigo de la función FSRMinSinOpt**

```
1  Funcion FSRmin <- FSRMinSinOpt 2(a,b,B,H,pr,otras)
2      // Se invoca a la función recien definida distminR1
3      [D,R2max] <-distminR1 (a,b,B,H)
4      R1=D+pr;
5      Si b<=H Entonces
6          R2=R2max;
7      Sino
8          // En realidad R2 es tambien variable de entrada
9          // de esta función. Por claridad se ha expresado así.
10         leer R2
11     FinSi
12     // En este punto ya se está en la misma situación de antes.
13     j=0; //contador de radios//
14     Para R<-R inicial Hasta Rfinal Con Paso pr Hacer
15         j=j+1
16         // Se invoca a la función FSP para calcular el FS en a,b,R
17         // y se guarda en el vector FSR(j)
18         FSR(j)<-FSP(R,a,b,otras variables)
19     FinPara
20     // Se busca el menor FS del vector FSR(j)
21     FSRmin <- min(FS,FSR(j))
22 FinFuncion
```

El recuadro muestra la invocación a la función *distminR1* que se ha diseñado en este apartado.

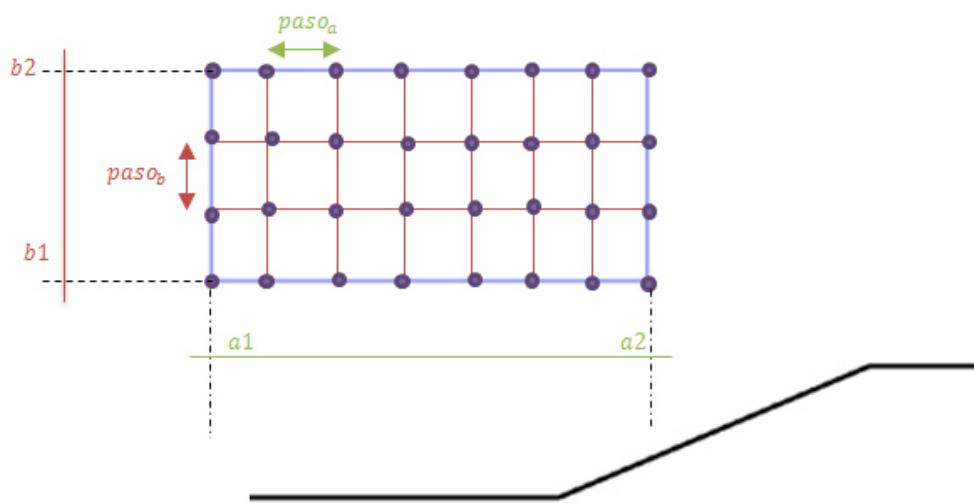
El resto del algoritmo es igual a *FSRminSinOpt\_1*.

### 4.3 Factor de seguridad en un marco de búsqueda

En este apartado se explicará cómo se puede localizar el FS mínimo asociado a un talud de tierra.

En el apartado anterior se mostró que a cada centro le corresponde un radio para el cual el *FS* es mínimo. Ahora se definirá una zona de búsqueda o marco lo suficientemente grande para considerar que la solución se encuentra en su interior y se definirá un paso o un número de puntos a evaluar dentro de ese marco de búsqueda.

Ilustración 4-7: Definición de marco de búsqueda



El marco queda definido por los valores extremos o esquinas del mismo

En la Ilustración 4-7 se observa una malla, donde cada punto representa al centro de una circunferencia con *FS* mínimo. Es decir, cada centro  $C(a,b)$  representa lo siguiente

$$C(a, b) \rightarrow [FS_{\min} \ R_{FS_{\min}}]$$

Esto supone una matriz equivalente a la malla mostrada en la imagen, donde las filas de la matriz son las posiciones  $b$  del centro, y las columnas las posiciones  $a$  del centro

$$\text{Matriz } FS_{\min} = \begin{bmatrix} FS_{b_2 a_1} & \cdots & FS_{b_2 a_2} \\ \vdots & \ddots & \vdots \\ FS_{b_1 a_1} & \cdots & FS_{b_1 a_2} \end{bmatrix}$$

La solución del problema será

$$FS_{\min \text{ global}} = \min[\text{Matriz } FS_{\min}]$$

Se ha considerado que la mejor forma de recorrer el marco es la siguiente. Para no complicar el asunto, en vez de elegir un paso para  $a$  y un paso para  $b$ , el usuario elige un *paso* en general con el que quiere avanzar, es decir, una distancia a la que le gustaría distar un centro de otro.

Por otra parte, también introduce unos vértices con los que se calculan los puntos  $pa$  que habrá que analizar en la dirección del eje x, y los puntos  $pb$  que habrá que analizar en la dirección y.

$$pa = \frac{(a2 - a1)}{paso}$$

$$pb = \frac{(b2 - b1)}{paso}$$

Los distintos centros que se deberán probar serán los siguientes:

$$a = linspace(a1, a2, pa)$$

$$b = linspace(b1, b2, pb)$$

Se ha resultado de esta forma, porque se quería estar seguro que las esquinas del marco eran analizadas como puntos finales o iniciales. Para llevar a cabo dicha tarea se ha creado la función *Rastreo*. La cual se puede observar en el siguiente recuadro

**Ilustración 4-8: Pseudocódigo de la función Rastreo**

```

1  Funcion FSminGlobalabR <- Rastreo (a1,a2,b1,b3,p,otros )
2      q=0; // Contador aumenta con el paso de cada centro
3      Para a <- a1 Hasta a2 Con Paso p Hacer
4          Para b <- b1 Hasta b2 Con Paso p Hacer
5              q=q+1;
6              // Se invoca al proceso FSRMinSinOpt en cada punto
7              FSRmab(1,:)<-FSRMinSinOpt_2(...)
8          FinPara
9      FinPara
10     // Se extrae el mínimo FS de FSRmab
11     FSMinGlobalabR=min(FS,FSRmab);
12 FinFuncion

```

# 5. Optimización de los algoritmos de búsqueda

## 5.1 Optimización de la búsqueda del FS ampliando el radio en un punto

### 5.1.1 Curvas típicas $FS(R)$ .

Se va a exponer en este apartado distintas imágenes de las curvas que presentan  $FS(R)$  con la intención de valorar si se puede atacar el problema con algún método de optimización numérica. Se presentarán la simulación A y la simulación B, de la cual se obtendrán las gráficas del  $FS(R)$  buscadas.

Se define la distancia que va del centro al vértice  $v_1$  como  $Dv_1$  al vértice  $v_2$  como  $Dv_2$ . Se vio anteriormente estos valores representaban el radio mínimo cuando el centro se encontraba en las zonas laterales respectivamente.

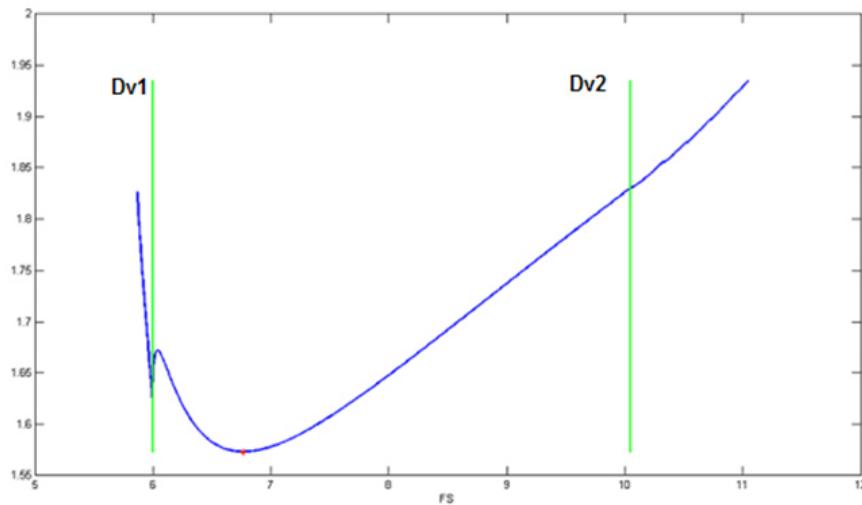
Como se verá a continuación, el conocimiento de estas distancias es esencial no solo para buscar el radio mínimo, sino porque en estos puntos la función sufre sobresaltos que la sacan de su comportamiento regular.

#### Simulación 5.1

Tabla 5-1: Valores de la simulación 5.1

Talud 1	Centro		Radios		
Puntos	a (m)	b (m)	R 1(m)	R2 (m)	pr (m)
P1	0	6	D+0.5	Dv2+1	0.0.1

Ilustración 5-1: Resultados de la simulación 5.1

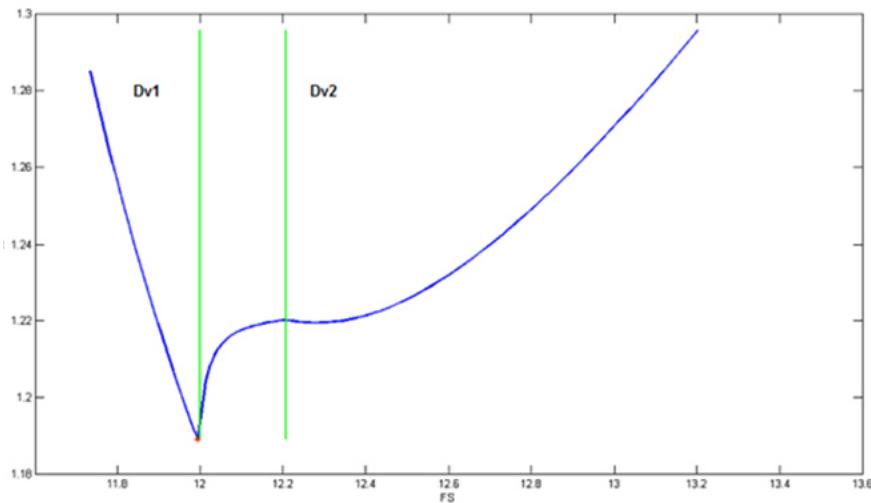


Simulacion 5.2

Tabla 5-2: Valores de la simulación 5.2

Talud 1	Centro		Radios		
Puntos	a (m)	b (m)	R 1(m)	R2 (m)	pr (m)
P1	0	12	Dmin+1	Dv2+1	0.0.1

Ilustración 5-2: Resultados de la simulación 5.2



En ambas simulaciones se está utilizando la función *FSRMinSinOpt* explicada en el capítulo anterior, y como se dijo era necesario introducir un valor que limitara el *R2*. En este caso se escoge sumar una cantidad a la cantidad *Dv<sub>2</sub>* para delimitar el *R2*. (ver Tabla 5-1 y Tabla 5-2)

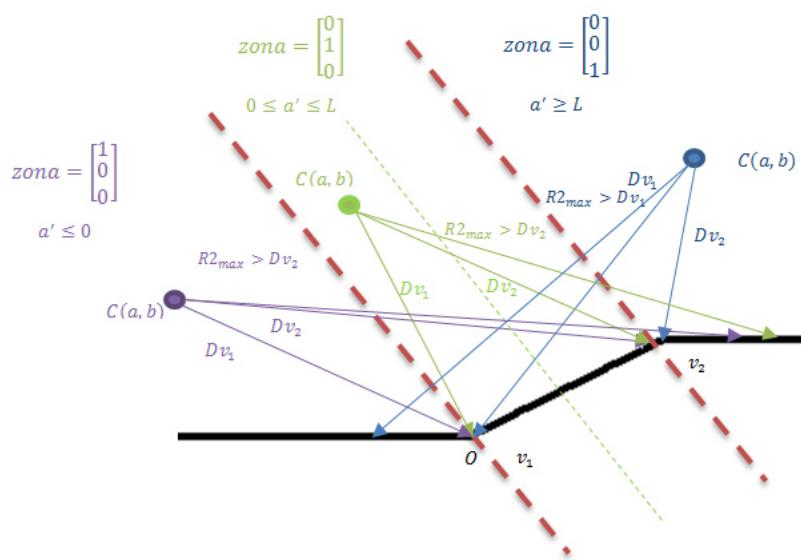
Se ve que las curvas  $FS(R)$  tienen puntos de no derivabilidad en los vértices del talud. Esta situación dificulta la implementación de métodos de optimización clásicos, como los métodos de la biseción o los métodos de descenso. Sin embargo se propone la implementación de un método que, sin abusar del cálculo de la derivada, la usa en las zonas regulares para acercarse hasta encontrar el mínimo de la función.

### 5.1.2 Algoritmo de optimización

Dentro del comportamiento irregular que tienen las curvas  $FS(R)$ , entre los vértices la función parece ser suave y derivable. Aun así, no se considera prudente entrar en estos tramos abusando de la evaluación de la derivada, ya que esto podría llevar a errores que no se desean para nada. Por esa razón se propone el siguiente algoritmo para intentar minimizar el número de búsquedas y a la vez dar un resultado robusto. Límite de  $R2$  cuando  $b > H$

En primer lugar, se necesita limitar el valor de  $R2$ . Para ello se procederá como sigue. Como se observa que en los vértices se experimentan cambios en la función, habrá que hacer crecer el radio al menos hasta traspasar el último vértice, ya que después de esto la función se comporta de forma regular. En la Ilustración 5-3 se muestra que dependiendo de la zona en la que se encuentre el centro, el último vértice que se traspasará, será uno u otro.

**Ilustración 5-3 : Cálculo del radio máximo cuando  $b > H$**



Pero lo que sí es seguro, es que la distancia al vértice más lejano simplemente será la mayor de ambas distancias. Entonces se dirá que como mínimo

$$R2_{max} > \max([Dv_1, Dv_2])$$

Se quiere poder evaluar la pendiente de un punto. Debido a que no se trata de una función conocida, sino de una función que se va creando punto por punto, se calculará la derivada aplicando el cálculo en diferencias finitas. La expresión que se usará para calcular la diferencia finita será la derivada (diferencia finita) centrada para un punto  $R_i$ . Esta fórmula tiene como expresión la siguiente

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$

Donde  $h$  es la distancia que hay entre cada punto  $x_i$ .

Particularizando para nuestro problema la expresión anterior se escribe como

$$f'(R_i) = \frac{f(R_{i+1}) - f(R_{i-1})}{2pr}$$

Donde  $f'(R_i)$  es la pendiente del FS para el radio  $R_i$ ,  $f(R_{i+1})$  el FS para el radio inmediatamente posterior y  $f(R_{i-1})$  el FS para el radio inmediatamente anterior.

Con lo cual, a partir de un radio superior a  $\max([Dv_1, Dv_2])$  se va evaluando la derivada aumentando el radio a pasos más o menos grandes. Cuando suceda que la pendiente es menor a cero será porque el factor de seguridad comienza a aumentar sus valores y como ya se ha atravesado el último vértice, se espera que la función no vuelva a descender, con lo cual se puede afirmar que de ahí en adelante no se espera encontrar el mínimo, es decir no tiene sentido seguir iterando.

#### Algoritmo 5-1: Pseudocódigo del subprocesso Radiomaximo

```

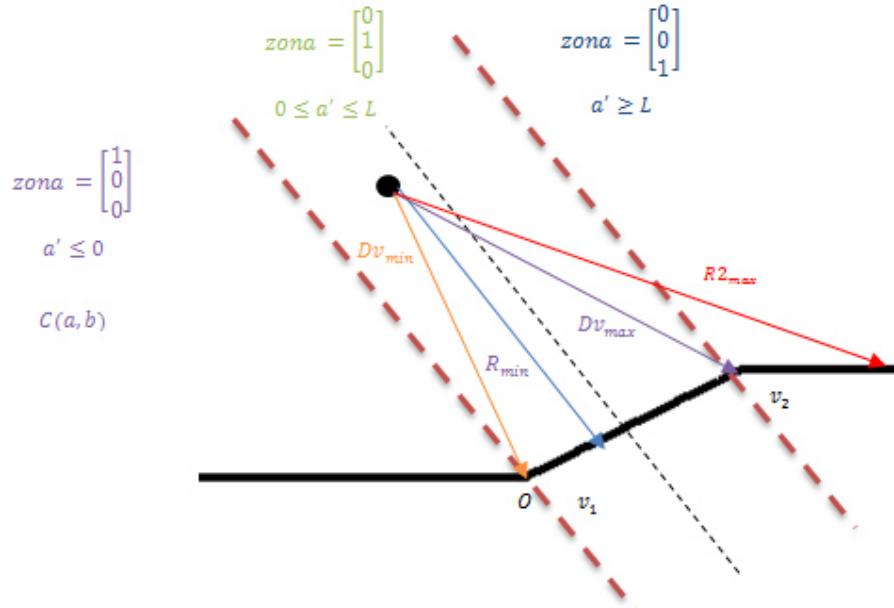
1  SubProceso R2max <-RadioMaximo ( dv1,dv2,otras )
2      R<-max([dv1,dv2]);
3      q=0; //contador
4      Mientras 2<3 Hacer
5          R <-R+0.5;
6          //Invocamos a la función FSP
7          FS(q)<-FSP(R,...);
8          // Se invoca a la función FSderivada para evaluar la derivada
9          derivada<-FSderivada(...);
10         Si derivada>0 Entonces
11             R2max=R;
12             BREAK
13         FinSi
14     FinMientras
15 FinSubProceso

```

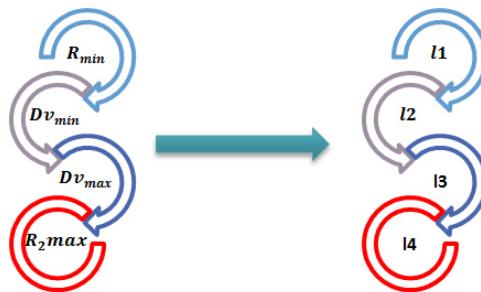
##### 5.1.2.1.1 División de la búsqueda en distintos tramos

Para dividir la función en distintos tramos, se va a suponer que el radio  $R$  siempre hace el siguiente recorrido.

Ilustración 5-4: Etapas al aumentar el radio



Como se puede ver, el radio aumenta desde  $R_{min}$  hasta  $R_{2max}$  pasando por los vértices. Este proceso se divide en distintas etapas, tal y como se explica a continuación

Ilustración 5-5: Asignación de variables  $l_i$ 

En este esquema se ve que para formular el problema de forma general se ha acudido a las variables  $l_1, l_2, l_3$  y  $l_4$ , esto se ha hecho así porque los valores de las distintas etapas, e incluso el número de etapas, dependen de la posición del centro.

Para comprender esto, solo habría que prestar atención a que si el centro se encontrara por ejemplo en el lado izquierdo, entonces la distancia mínima coincidiría con la distancia al vértice uno, significando que habría una etapa menos que la que se ha mostrado en la Ilustración 5-4.

Si además de esto, se tiene en cuenta que si el centro  $b$  está por debajo de  $H$  el valor de  $R2_{max}$  es otro, entonces se comprende mejor el uso de las variables  $l_i$ .

Con lo cual, la segunda etapa del algoritmo de optimización es establecer un algoritmo que asigne correctamente los valores de  $l_i$ . La forma más sencilla de transmitir esta idea es a través de la inspección del código fuente. Este subproceso se denominará *asignaciónli*.

#### Algoritmo 5-2: Subproceso de la función FSRMinConOpt

```

28  %Subproceso: asignacionli
29  %Las distintas posibilidades son
30  if b<=H
31      if isequal(zona,[0 1 0]) %Dmin==Dmc Se encuentra en la zona central
32          if min([Dv1 R2max])==Dv1 % Primero llega a Dv1
33              l1=Dmin; l2=Dv1; l3=R2max; l4=0;
34          elseif min([Dv1 R2max])==R2max % No llega a Dv1
35              l1=Dmin; l2=R2max; l3=0; l4=0;
36          end
37      elseif isequal(zona,[1 0 0])%Dmin~=Dmc Se encuentra en la zona exterior
38          l1=Dv1; l2=R2max; l3=0; l4=0;
39      end
40  elseif b>H
41      if isequal(zona,[0 1 0]) && Dv1~=Dv2
42          l1=Dmin; l2=min([Dv1 Dv2]); l3=max([Dv1 Dv2]); l4=Dfin;
43      elseif isequal(zona,[1 0 0]) || isequal(zona,[0 0 1])
44          l1=Dmin; l2=max([Dv1 Dv2]); l3=Dfin; l4=0;
45      elseif isequal(zona,[0 1 0]) && Dv1==Dv2
46          l1=Dmin; l2=Dv1; l3=Dfin; l4=0;
47      end
48  end

```

### 5.1.3 Algoritmo final

Ahora que se tienen los límites de cada tramo, se puede tratar cada tramo de forma independiente. ¿Qué significa esto? Pues significa que se puede entrar en cada uno de esos tramos y esperar que mientras no se traspase su límite, la función se comportará de forma suave y derivable.

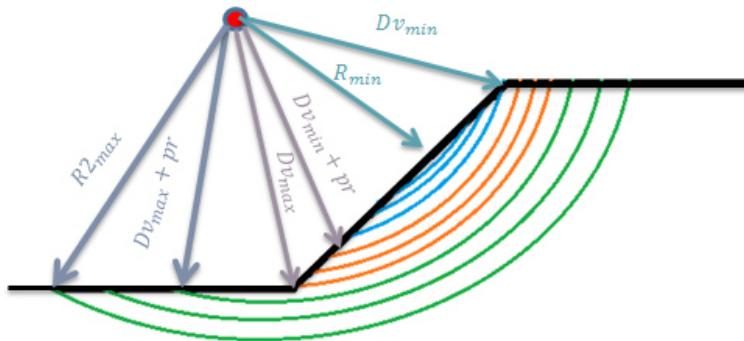
La estrategia será ahora la siguiente. El usuario elige desde un principio el número  $N$  máximo de iteraciones que le gustaría hacer para encontrar el mínimo. Este número  $N$  coincide con el número de radios que se van a probar.

Como la función debe ser tratada como si cada tramo fueran funciones independientes, lo que se hará será repartir los  $N$  radios ponderadamente en cada tramo. Esta tarea lo hará el subproceso *reparticiónN*.

Como ya se tienen los radios que se van a calcular de cada tramo. El siguiente paso es hacer la simulación de cada tramo y localizar el FS mínimo en cada tramo.

El método es realmente sencillo e intuitivo de seguir y cómo es posible que el lector no vea la claridad de este entre tantos párrafos se exponen a continuación dos ilustraciones que deberían aclarar bastante lo que se trata de explicar.

**Ilustración 5-6: Ejemplo desimulación usando la función FSRMinConOpt**

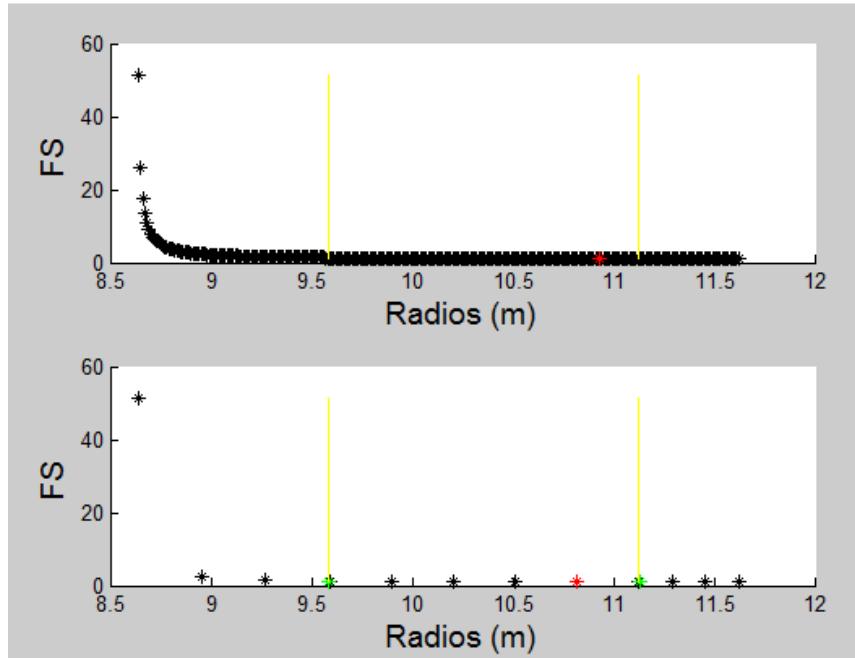


En la *Ilustración 5-6* se intenta ilustrar buena parte del método. Las tres circunferencias azules representan las que se toman entre  $l_1 = R_{min}$  y  $l_2 = Dv_{min}$ . Una vez el radio supera el valor de  $Dv_{min}$  se entra en el siguiente tramo, que se ha representado con el color naranja. Finalmente el radio alcanza el valor del vértice dos y comienza el tercer tamo, en verde, que termina cuando el radio alcanza el valor del radio máximo.

En la *Ilustración 5-7* se observa una comparación de las dos simulaciones. En el eje y se representa el FS y en el eje x se representan los distintos radios. Las marcas amarillas separan los distintos tramos.

La simulación de arriba se hizo sin optimizar, y como se ve, al ser necesario avanzar con un paso  $pr$  hasta llegar al final, el número de puntos calculados es bastante grande. En la de abajo se hizo la técnica de optimización con catorce puntos. Los puntos en verde muestran los mínimos en cada tramo. El punto en rojo muestra el mínimo global.

Ilustración 5-7: Comparación de la función FSRMinConOpt y FSRMinSinOpt



El siguiente paso es seleccionar al mínimo de cada tramo, y hacer una búsqueda minuciosa en su entorno. Esta búsqueda se hará con ayuda de la derivada, ya que finalizará cuando se tope con un cambio en el valor de esta, o bien cuando se tope con uno de los extremos del tramo. Después de esta operación se vuelve a tener un representante mínimo -más preciso- de cada tramo. El paso que queda para finalizar es escoger el mínimo de todos.

Sin duda alguna, la mejor forma de comprender el algoritmo es observando el pseudocódigo siguiente, el cual se denomina *minimotramo1*.

**Algoritmo 5-3: Subproceso de la función FSRMinConOpt**

```

1   Subproceso minFSRiminFSR1<-minimotramo1(p1,p2,p3,pr,11,12,13,14,otros )
2       //tramo1
3       //p1,p2,p3 son los pasos calculados para cada tramo
4       //pr es el paso con el que se moverá el radio en la etapa de precisión
5       //11,12,13,14 son los límites de cada tramo
6
7       FinSubProceso
8       q=0;
9       Para R<-11 Hasta 12 Con Paso p1 Hacer
10      q=q+1;
11      //Invocamos a la función FSP
12      FSR1(q)<-FSP(R,...);
13
14      FinPara
15      FSR1min=min(FS,FSR1);
16      R=FSR1min(2); // El vector FSR1min contiene al radio crítico
17      //Calculamos la derivada
18      derivada<-FSderivada(R...);
19      //Extraemos el signo de la derivada
20      //El cual nos va a indicar hacia dónde está el mínimo
21      signo=derivada/abs(derivada);
22      j=0
23      // Mientras no pase por encima del mínimo, o se salga del tramo
24      Mientras signo==nuevosigno o R=11 o R=12
25          j=j+1;
26          R=R+signo*pr; //Si el signo<0, entonces resta.
27          FSR1minFSR1(j)<-FSP(R,...);
28          nuevaderivada<-FSderivada(R...)
29          nuevosigno=nuevaderivada/abs(nuevaderivada);
30
31      FinMientras
31      minFSRiminFSR1=min(FS,FSR1minFSR1);
31      FinSubProceso

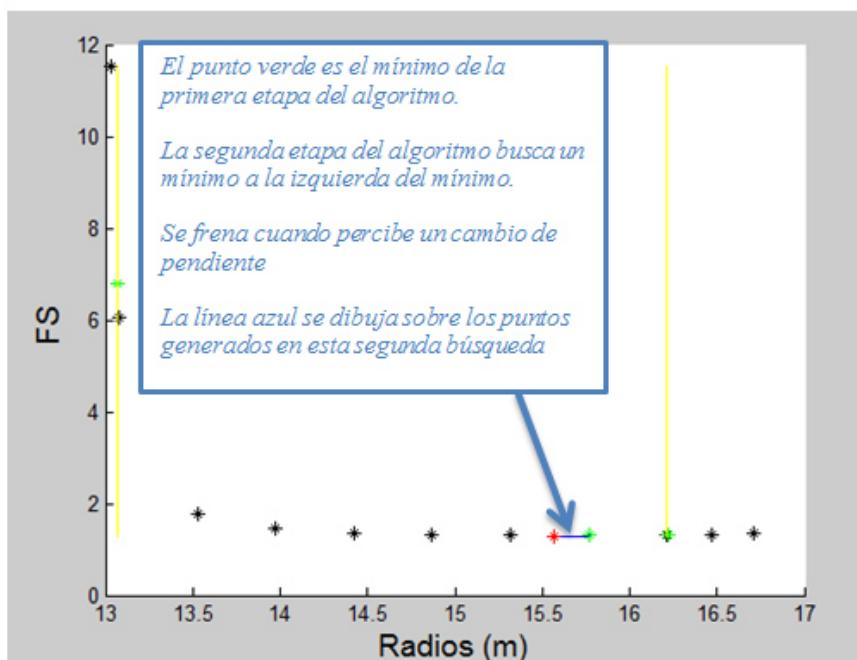
```

**Etapa 1.**  
Búsqueda de  
mínimo  
general por  
tramo

**Etapa 2.**  
Afinamiento  
de la etapa  
1

En la siguiente imagen se podrá observar cómo funciona este algoritmo

**Ilustración 5-8: Segunda etapa del algoritmo de optimización**



Este subprocesso es el encargado de averiguar cuál es el mínimo del tramo 1 haciendo un proceso optimizado. Para los otros tramos el subprocesso es idéntico pero con sus variables correspondientes.

El siguiente algoritmo presenta un esquema de cómo funciona toda la función, llamando a los subprocessos y funciones que aquí se han explicado. La función que engloba a todos estas rutinas se denomina *FSRMinConOpt*.

**Algoritmo 5-4: Pseudocódigo de la función FSRMinConOpt**

```

1  Funcion FSRminOPT <- FSRMinConOpt (PT,...)
2      // PT es el número de puntos con el que se quiere realizar la simulacion//
3      // Se invoca al subprocesso RadioMaximo
4      R2max <-RadioMaximo (...)
5      // Se invoca al subprocesso asignacionli
6      [11,12,13,14] <-asignacionli (...)
7      // Se invoca al subprocesso asignacionli
8      // Se invoca a los subprocessos minimotramoi
9      minFSR1minFSR1<-minimotramo1(.... )
10     minFSR1minFSR2<-minimotramo2(.... )
11     minFSR1minFSR3<-minimotramo3(.... )
12     // Se calcula el minimo
13     FSRminOPT=min(FS,[minimotramo1 minimotramo2 minimotramo3])
14  FinFuncion

```

### 5.1.4 Conclusión

Para comparar este algoritmo con el original, se va a hacer una simulación para comprobar que ambos valores coinciden. Además se compararán los tiempos que tarda uno con respecto a otro.

#### *Simulación 5.3*

**Tabla 5-3: Valores de la simulación 5.3**

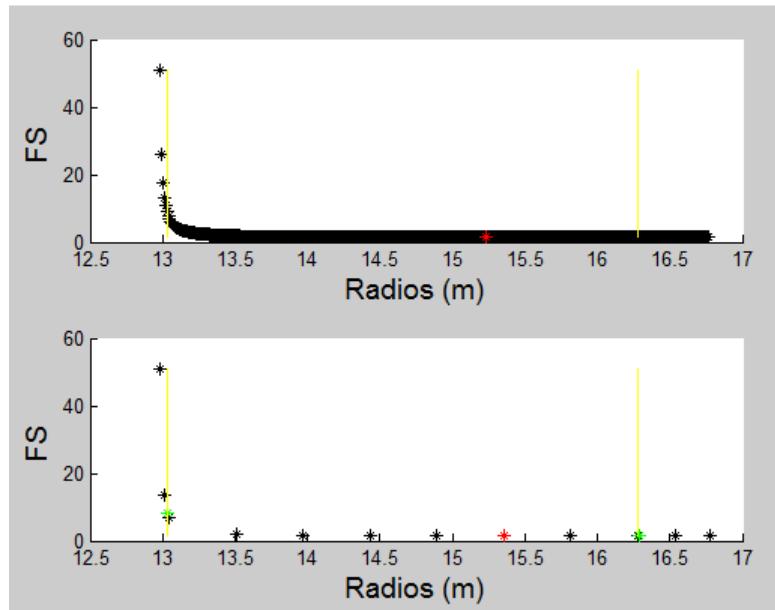
a(m)	b(m)	talud	método	rebanadas	PTR
3	16	3	M-P	100	7

Los resultados de la simulación son los siguientes

**Tabla 5-4: Resultados de la simulación 5.3**

Sin optimizar			
FS	R(m)	t(s)	FSP
1.3534	15.229	0.82	762
Con optimización			
FS	R(m)	t(s)	FSP
1.3539	15.3558	0.02	19

Ilustración 5-9: Resultados de la simulación 5.3



Se observa que con optimización se obtiene un resultado casi idéntico al que se hace sin optimizar. Además, se podría mejorar el resultado aumentando el número de simulaciones (PTR) y aumentando la sensibilidad de la derivada.

La columna cuatro de la Tabla 5-4 se observan los tiempos de la simulación. Dividiendo el mayor entre el menor se tiene

$$rt = \frac{t \sin opt.}{t \text{ con opt.}} = \frac{0.82}{0.02} = 40.4$$

Por otro lado, la columna cinco de la misma tabla aparece el número de invocaciones que se hicieron a la función FSP. Operando de la misma forma que con el tiempo se tiene

$$rp = \frac{\text{Invocaciones de FSP sin opt.}}{\text{Invocaciones de FSP con opt.}} = \frac{769}{18} = 40.4$$

Es decir, en esta simulación se consigue obtener el mismo resultado mejorando la eficiencia del programa. Por cada simulación que se hace optimizando, sin optimizar se deben hacer cuarenta, o dicho de otra forma, por cada minuto invertido en buscar el FS optimizando, sin optimizar harían falta cuarenta minutos.

## 5.2 Optimización de la búsqueda del FS en un marco

### 5.2.1 Curvas típicas FS(a,b)

De la misma forma que en el apartado anterior, se comenzará analizando las curvas típicas que resultan de recorrer una malla. A continuación se expone una simulación de la que se sacarán dichas curvas

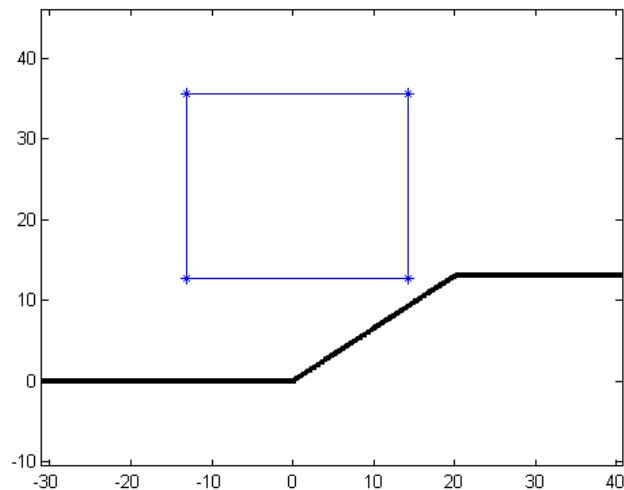
*Simulación 5.4*

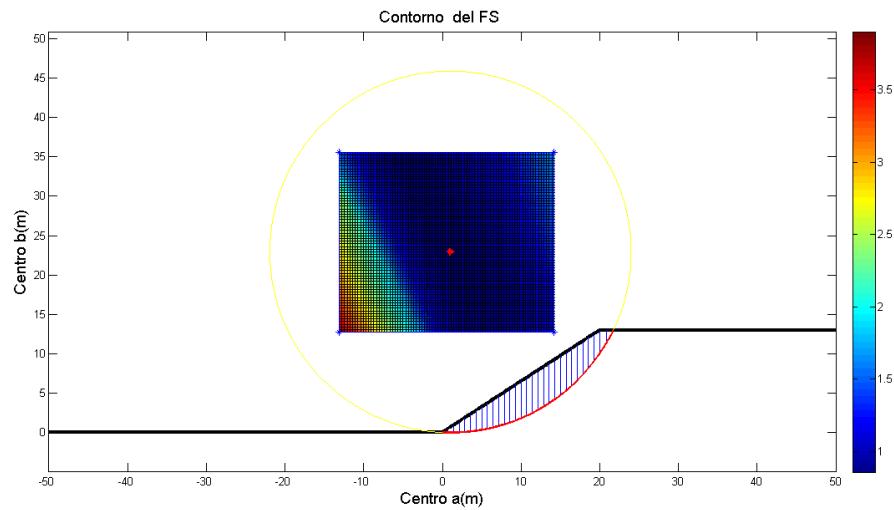
Tabla 5-5: Valores de la simulación 5.4

Talud	<b>7</b>	
paso	<b>0.33 m</b>	
Método	<b>Fellenius</b>	
Vértices	<b>V1 (m)</b>	<b>V2 (m)</b>
<b>vx</b>	-13.153	14.199
<b>vy</b>	35.542	12.72

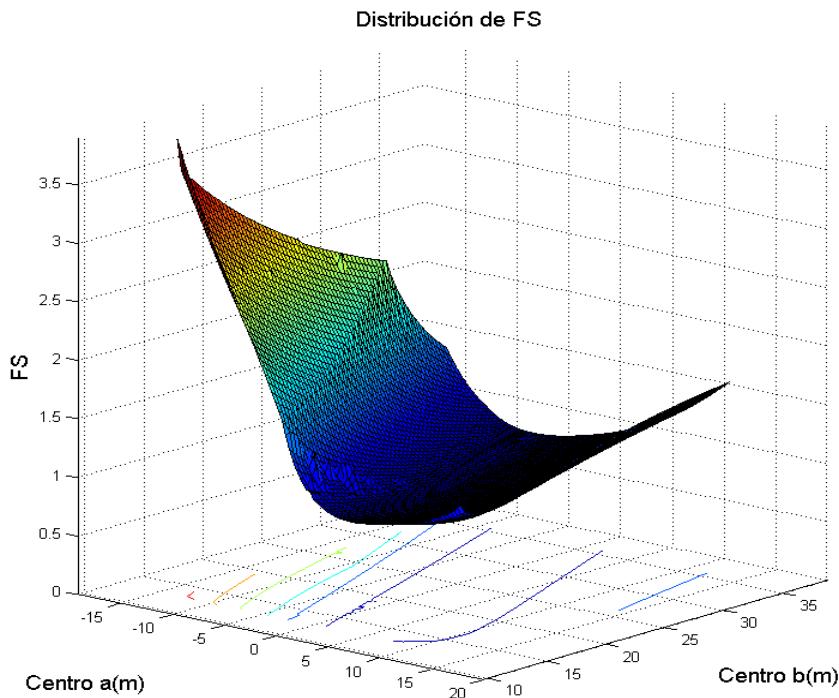
Se muestra en la *ilustración 5-10* el talud junto con el marco que se utilizará para buscar el *FS* mínimo

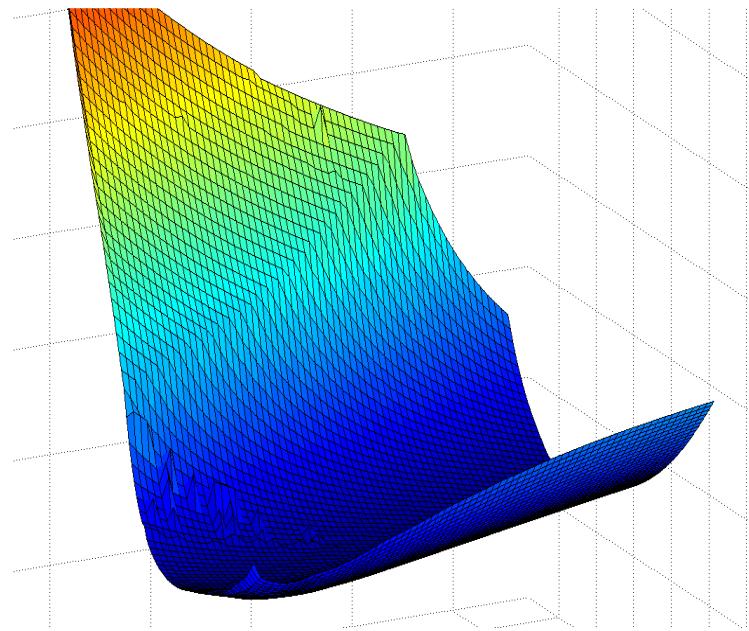
Ilustración 5-10: Marco inicial de la simulación 5.4



**Ilustración 5-11: Marco con distribución del FS y FS crítico**

En la *ilustración 5-11* se observa el resultado de la búsqueda. En esta se puede observar el mismo talud y el mismo marco, pero en su interior queda representada la distribución del *FS* en el marco. A la derecha aparece una barra de colores que permite asociar cada color a cada valor del *FS*. Por otro lado, la solución se representa con un punto rojo.

**Ilustración 5-12: Distribución 3D del FS**

**Ilustración 5-13: Zoom de la distribución del FS en 3D**

En la *ilustración 5-12* se puede observar la distribución del *FS* en el marco mostrado en la imagen anterior. Los ejes que están sobre el suelo representan al *centro a* y al *centro b*. El eje z representa los valores del *FS*. La *imagen 5-13* es un zoom en la *imagen 5-12*, este se tomó para poder ver más de cerca como es la superficie de dicha curva.

De estas imágenes se desprende la idea de que la curva es más o menos suave y contiene un único mínimo. Sin embargo, al hacer un zoom sobre ella se puede observar que hay ciertas discontinuidades y por ello no es aconsejable aplicar un método de optimización basado en el cálculo del gradiente, quedando descartados los métodos de descenso y todas sus variantes.

### 5.2.2 Algoritmo de optimización

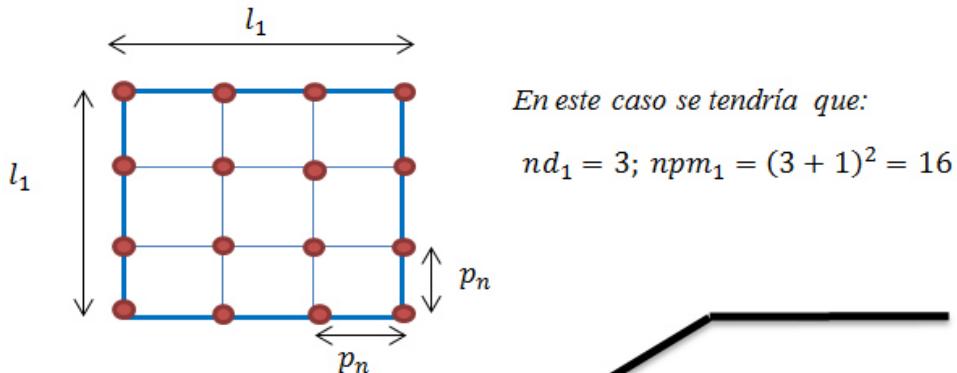
Dado un marco inicial de forma cuadrada cuyos lados tienen una longitud de  $l_1$  y dada una precisión  $p_1$  que se desea obtener. El número de divisiones  $nd$  que se harán sobre un lado de este marco será

$$nd_1 = \frac{l_1}{p_1}$$

El número de puntos que se analizarán será

$$npm_1 = (nd_1 + 1)^2$$

Ilustración 5-14: Marco con variables de optimización



En la imagen se ha dibujado un marco cuadrado con tres divisiones por lado. Los puntos sobre los que se evaluará cada *FS* se han dibujado como puntos en rojo oscuro.

Si  $l_1$  comparado con  $p_1$  es relativamente grande entonces  $nd$  podría ser bastante grande y esto no interesa para nada, ya que los tiempos de simulación serían demasiado grandes.

Se propone hacer el siguiente método para optimizar esta tarea. Se realizará en primer lugar un recorrido general sobre un marco inicial y luego un recorrido más minucioso sobre un marco más pequeño. Este marco más pequeño se dibujará en el entorno del mínimo encontrado en el marco general. A la hora de realizar esto, surgen varias preguntas que se irán respondiendo a medida que se vaya avanzando.

Para ayudar a la comprensión del método se irá resolviendo a la vez un ejemplo a lo largo del desarrollo, y así observar cómo se mejoran los resultados.

1)

Se desea analizar un marco cuyo lado  $l_1 = 10 \text{ m}$  y se quiere dar un *FS* mínimo global con una precisión de  $p_2 = 0.01 \text{ m}$ . Se considera también que el marco es lo suficientemente grande para garantizar que el mínimo está en su interior.

$$nd_1 = \frac{l_1}{p_1} = \frac{10}{0.01} = 1000$$

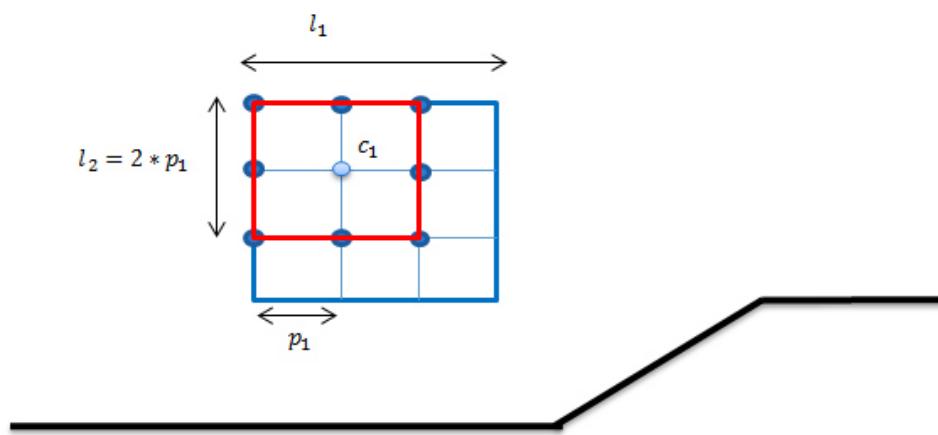
$$npm_1 = (1000 + 1)^2 = 1.002.001$$

Como se verá es un número bastante grande.

### 5.2.2.1 Tamaño mínimo necesario del marco secundario

Dado un marco inicial, siendo el tamaño de sus lados  $l_1$  que se rastrea con una precisión de  $p_1$  y se le encuentra un mínimo en el punto  $c_1$ . Se quiere hacer un segundo marco, más pequeño que el primero y rastrearlo con una precisión  $p_2$  menor que  $p_1$  previamente estipulada por el usuario. ¿Cuál debería de ser el tamaño del marco secundario para garantizar que en su interior hay un mínimo?

**Ilustración 5-15: Marco consecutivo mínimo**



En el dibujo se observa un marco inicial de longitud  $l_1$  que fue rastreado con un paso  $p_1$ . Se localizó el punto con *FS* mínimo en el punto  $c_1$  y entorno a este punto se encuentran los puntos que son inmediatamente más grandes que este. Si la función es lo suficientemente suave entonces se puede garantizar que el marco dibujado en rojo, se encuentra el mínimo.

Con lo cual, se ve prudente decir que el siguiente marco en el que se debería de hacer la búsqueda debería decir igual o mayor que el que se ve pintado en rojo. Siendo  $l_2$  la longitud del lado de este marco entonces

$$l_2 = 2 * p_1$$

Ahora se podrá hacer un segundo rastreo dentro del segundo marco con la precisión deseada  $p_2$ . Sobre el lado del primer recuadro se hacen el siguiente número de divisiones

$$nd_1 = \frac{l_1}{p_1}$$

Sobre el segundo

$$nd_2 = \frac{l_2}{p_2}$$

Y la suma de ambos

$$nd_{total} = (nd_1 + nd_2) = \frac{l_1}{p_1} + \frac{l_2}{p_2}$$

Ahora, el número total de puntos sobre los que se calculó un *FS* es

$$npmt_{total} = (nd_1 + 1)^2 + (nd_2 + 1)^2 = \left(\frac{l_1}{p_1} + 1\right)^2 + \left(\frac{l_2}{p_2} + 1\right)^2$$

Si se piensa bien, existen diversas combinaciones para realizar este método. Es decir, definidos previamente el lado del marco inicial  $l_1$  y la precisión  $p_2$  a la que se desea llegar, la precisión  $p_1$  -que es la con la que se rastrea el primer marco y determina el tamaño del segundo marco es una variable que no está determinada. El número de puntos que al final se acabe analizando dependerá de la elección del valor de esta.

Se resuelve ahora el ejemplo 1), con los nuevos conceptos que se han explicado. Además se podrá ver que el resultado dependerá de la elección de  $p_1$ .

En el primer caso, la precisión inicial es  $p_1 = 1$  y en el siguiente ejemplo la precisión inicial será  $p_1 = 0.1$ . Se verá que los resultados son drásticamente diferentes

2)

$$l_1 = 10 \text{ m}; p_1 = 1 \text{ m}; p_2 = 0.01 \text{ m}$$

$$npmt_1 = \left(\frac{10}{1} + 1\right)^2 + \left(\frac{2*1}{0.01} + 1\right)^2 = 121 + 40401 = 40522$$

3)

$$l_1 = 10 \text{ m}; p_1 = 0.1 \text{ m}; p_2 = 0.01 \text{ m}$$

$$npmt_2 = \left(\frac{10}{0.1} + 1\right)^2 + \left(\frac{2*0.1}{0.01} + 1\right)^2 = 10201 + 441 = 10642$$

Se deduce que debe de existir un  $p_1$  que hace que el número de puntos sea mínimo. En vez de resolver este problema, se va a formular primero de forma general. Es decir, si en vez de practicar esta técnica con un marco inicial y uno final, se plantea el problema para  $N$  marcos.

### 5.2.2.2 Número $N$ de marcos

Antes de comenzar con la explicación de este método, se debe resaltar que los desarrollos planteados en esta subsección 5.2.2.2 y posterior 5.2.2.3 son suposiciones cuyas conclusiones derivadas se han aplicado de forma satisfactoria en los ejemplos de la parte final del proyecto. Dicho esto se puede comenzar la explicación y desarrollo de dicho método.

Dado un número  $N$  de marcos, y dado el conocimiento de la longitud del marco inicial  $l_1$  y de la precisión final  $p_n$  con la que se desea conocer el resultado se tendrá que el número de puntos totales vendrá determinado por

**Ecuación 5-1**

$$npm_{total} = \left(\frac{l_1}{p_1} + 1\right)^2 + \left(\frac{l_2}{p_2} + 1\right)^2 + \dots + \left(\frac{l_i}{p_i} + 1\right)^2 + \dots + \left(\frac{l_n}{p_n} + 1\right)^2$$

recordando que

**Ecuación 5-2**

$$l_i = 2p_{i-1}$$

entonces se podrá escribir que

**Ecuación 5-3**

$$npm_{total} = \left(\frac{2*p_0}{p_1} + 1\right)^2 + \left(\frac{2*p_1}{p_2} + 1\right)^2 + \dots + \left(\frac{2*p_{i-1}}{p_i} + 1\right)^2 + \dots + \left(\frac{2*p_{n-1}}{p_n} + 1\right)^2$$

Si el problema planteado inicialmente tenía un grado de libertad, siendo este el paso  $p_1$  con el que se recorría el cuadro inicial, entonces el problema general tendrá  $n-1$  grados de libertad, siendo  $n$  el número de marcos o etapas totales.

Se quiere obtener el conjunto de valores  $p = \{p_1, p_2, p_3, \dots, p_i, \dots, p_{n-1}\}$  que hacen que la función  $npm_{total}$  sea la mínima posible.

Si  $npm_{total}$  es mínima, también lo será  $nd_{total}$ , y viceversa. Con lo cual el objetivo del problema será minimizar la siguiente función

**Ecuación 5-4**

$$nd_{total} = \frac{2 * p_0}{p_1} + \frac{2 * p_1}{p_2} + \dots + \frac{2 * p_{i-1}}{p_i} + \dots + \frac{2 * p_{n-1}}{p_n}$$

Se tiene un problema de optimización de una función de varias variables. Con lo cual se procede ahora como corresponde con un problema de estas características.

Un diferencial de la función  $nd_{total}$  vendrá expresado de la siguiente forma

$$d(nd_{total}) = \frac{\partial(nd_{total})}{\partial p_1} * dp_1 + \frac{\partial(nd_{total})}{\partial p_2} * dp_2 + \dots + \frac{\partial(nd_{total})}{\partial p_i} * dp_i + \dots + \frac{\partial(nd_{total})}{\partial p_n} dp_n$$

Es decir

$$d(nd_{total}) = \sum_1^n \frac{\partial(nd_{total})}{\partial p_i} dp_i$$

Resolviendo se tiene

$$\frac{\partial [nd_{total}(p_1, p_2, p_3, \dots, p_i, \dots, p_n)]}{\partial p_i} = \frac{\partial}{\partial p_i} \left( \frac{2 * p_0}{p_1} + \frac{2 * p_1}{p_2} + \dots + \frac{2 * p_{i-1}}{p_i} + \frac{2 * p_i}{p_{i+1}} + \dots + \frac{2 * p_{n-1}}{p_n} \right)$$

Teniendo finalmente que

$$\frac{\partial [nd_{total}(p_1, p_2, p_3, \dots, p_i, \dots, p_n)]}{\partial p_i} = \frac{\partial}{\partial p_i} \left( \frac{2 * p_{i-1}}{p_i} + \frac{2 * p_i}{p_{i+1}} \right)$$

Resolviendo se obtiene que

$$\frac{\partial(nd_{total})}{\partial p_i} = \frac{-2 * p_{i-1}}{p_i^2} + \frac{2}{p_{i+1}}$$

Igualando a cero se tiene

$$\frac{\partial(nd_{total})}{\partial p_i} = 0; \frac{-2 * p_{i-1}}{pi^2} + \frac{2}{p_{i+1}} = 0$$

Obteniendo finalmente

$$\frac{2p_{i-1}}{pi} = \frac{2p_i}{p_{i+1}}$$

Es decir, todos los términos de la ecuación son iguales. Por otro lado, despejando de esta ecuación  $p_i$  se tiene

$$p_i = \sqrt[2]{p_{i-1} * p_{i+1}}$$

A continuación se hace lo siguiente por comodidad

$$nd = \frac{2 * p_o}{p_1} \xrightarrow{\text{se divide entre dos}} \frac{nd}{2} = \frac{p_o}{p_1} = nd^*;$$

La forma de proceder será la siguiente, se sabe que  $\frac{p_o}{p_1} = nd^*$ , pero no se sabe cuantos valen los siguientes valores  $\left\{ \frac{p_o}{p_2}, \frac{p_o}{p_3}, \dots, \frac{p_o}{p_i}, \dots, \frac{p_o}{p_{n-1}} \right\}$ . Si se pudieran dejar todos los elementos de este conjunto en función del valor de algo conocido, entonces el problema estaría resuelto. Así que, usando las ecuaciones que se han obtenido se plantean las siguientes operaciones

$$\frac{p_o}{p_2} = \frac{p_o}{\underline{p_1^2}} = \left( \frac{p_o}{p_1} \right)^2 = nd^{*2}$$

$$nd^{*2} = \frac{p_o}{p_2}; nd^* = \sqrt[2]{\frac{p_o}{p_2}}$$

Se sigue con

$$\frac{p_o}{p_3} = \frac{p_o}{\frac{p_2^2}{p_1}} = \frac{p_o}{(\frac{p_1^2}{p_o})^2} = \frac{p_o}{\frac{p_1^3}{p_o^2}} = \left(\frac{p_o}{p_1}\right)^3 = nd^{*3}$$

$$nd^{*3} = \frac{p_o}{p_3}; nd^* = \sqrt[3]{\frac{p_o}{p_3}}$$

Si se extiende este método hasta  $n$ , entonces se resolverá que

$$nd^* = \sqrt[n]{\frac{p_o}{p_n}}; nd = 2 * ndt^*$$

**Ecuación 5-5**

$$nd = 2 * \sqrt[n]{\frac{p_o}{p_n}}$$

Con lo cual, para conseguir el vector  $p^*$ , el cual contiene el valor de todas las variables que optimizan el problema se hará lo siguiente

$$\frac{2p_{i-1}}{p_i} = \frac{2p_i}{p_{i+1}} = nd = 2 * \sqrt[n]{\frac{p_o}{p_n}}$$

$$\frac{p_o}{p_i} = \sqrt[n]{\left(\frac{p_o}{p_n}\right)^i}$$

$$p_i = \sqrt[n]{\left(\frac{p_n}{p_o}\right)^i} * p_o; \sqrt[n]{\left(\frac{p_n}{p_o}\right)^i * p_o^n} = \sqrt[n]{p_n^i * p_o^{n-i}};$$

El conjunto de valores  $p^*$  vendrá dado por

**Ecuación 5-6**

$$p^* = \left\{ \sqrt[n]{p_n^0 * p_o^n}, \sqrt[n]{p_n^1 * p_o^{n-1}}, \dots, \sqrt[n]{p_n^i * p_o^{n-i}}; \dots, \sqrt[n]{p_n^n * p_o^0} \right\}$$

Si se sigue con el ejemplo que se propuso anteriormente

4)

$$l_1 = 10; p_o = \frac{l_1}{2} = 5; p_n = 0.01$$

$$p_1 = \sqrt[2]{5 * 0.01} = \frac{\sqrt[2]{5}}{10}$$

$$\frac{2po}{p1} + \frac{2p1}{pn} = \frac{2 * 5}{\sqrt[2]{5}} + \frac{2 \frac{\sqrt[2]{5}}{10}}{0.01} = 20\sqrt[2]{5} + 20\sqrt[2]{5} = 40\sqrt[2]{5} = 89.44 = 89;$$

$$npm_{total} = (nd_1 + 1)^2 + (nd_2 + 1)^2 = 2 * (nd + 1)^2 = 2 * (20\sqrt[2]{5} + 1)^2 = 4181$$

Como se puede observar, el valor obtenido vuelve a ser menor, siendo este caso el menor posible, como ha sido demostrado.

### 5.2.2.3 Número óptimo de marcos

Ahora la pregunta que cabe hacerse es la siguiente. ¿Cuántos marcos son necesarios para que  $npm_{total}$  sea el menor posible? Partiendo de la expresión de  $nd_{total}$  (Ecuación 5-4)

$$nd_{total} = \frac{2 * po}{p1} + \frac{2 * p1}{p2} + \dots + \frac{2p_{i-1}}{pi} + \dots + \frac{2p_{n-1}}{pn}$$

Teniendo en cuenta la Ecuación 5-5

$$nd = 2 * \sqrt[n]{\frac{po}{pn}}$$

Sumando n veces se tendrá

Ecuación 5-7

$$nd_{total} = 2n * \sqrt[n]{\frac{po}{pn}}$$

Si se deriva esta expresión y se iguala a cero se sabrá cuál es el número  $n$  de marcos óptimos

$$\frac{d(\text{nd}_{total})}{dn} = \frac{d\left(2n * \sqrt[n]{\frac{po}{pn}}\right)}{dn} = 0$$

$$2n' * \sqrt[n]{\frac{po}{pn}} + 2n * \sqrt[n]{\frac{po'}{pn}}$$

$$\frac{d(\text{nd}_{total})}{d(n)} = 2 * \sqrt[n]{\frac{po}{pn}} - 2n * \frac{\sqrt[n]{po}}{n^2} * \ln \frac{po}{pn}$$

Se iguala a cero

$$2 * \sqrt[n]{\frac{po}{pn}} - 2n * \frac{\sqrt[n]{po}}{n^2} * \ln \frac{po}{pn} = 0$$

$$2 * \sqrt[n]{\frac{po}{pn}} = 2n * \frac{\sqrt[n]{po}}{n^2} * \ln \frac{po}{pn}$$

Simplificando esta igualación se obtiene que el número de etapas que optimiza el problema es

**Ecuación 5-8**

$$et = \ln \frac{po}{pn}$$

Volviendo al ejemplo que se ha ido resolviendo para ver si las soluciones mejoran se tiene

5)

$$l_1 = 10; po = \frac{l_1}{2} = 5; pn = 0.01$$

$$et = \ln \frac{po}{pn} = \ln \frac{5}{0.01} = 6.21 \approx 6$$

Con lo cual,

$$\text{nd}_{total} = 2n * \sqrt[n]{\frac{po}{pn}} = 2 * 6 * \sqrt[6]{\frac{5}{0.01}} = 33.80;$$

Y como el valor que interesa realmente es  $npm_{total}$

$$npm_{total} = n * \left( 2 * \sqrt[n]{\frac{po}{pn}} + 1 \right)^2 = 6 * \left( 2 * \sqrt[6]{\frac{5}{0.01}} + 1 \right)^2 = 264.1$$

En la siguiente tabla se puede ver la evolución del problema a medida que se fueron incorporando las distintas técnicas de optimización

Tabla 5-6: Resultados de la optimización

<i>Tipo de optimización</i>	<i>Puntos total analizados en cada simulación</i>
<b>1) Sin optimizar</b>	<b>1.002.001</b>
<b>2) Dos marcos con <math>p_1 = 1m</math></b>	<b>40522</b>
<b>3) Dos marcos con <math>p_1 = 0.1m</math></b>	<b>10642</b>
<b>4) Dos marcos con <math>p_1 = p_1 óptimo</math></b>	<b>4181</b>
<b>5) Número de marcos óptimo con pasos óptimo</b>	<b>264</b>

Como se puede observar, la reducción de puntos a analizar es enorme.

Una modificación del método sería a la hora de establecer que la longitud del marco siguiente debería de ser igual al doble del paso del marco anterior. Cuando se dice que

$$li = fe * 2p_{i-1}$$

Se está llevando el problema a un extremo. Es decir, más pequeño que esto, no podría ser. Se puede decir que ésta sería la mejor de las opciones, pero para ello se necesitaría saber con certeza que la función es derivable en todos los puntos del espacio factible, y esto no es del todo cierto. Sí es cierto que a medida que se acerca al mínimo, la función es menos sobresaltada y se comporta de forma más regular.

Lo que se propone es establecer un factor de ensanche  $fe$  de la siguiente forma

$$li = fe * 2p_{i-1}$$

Este factor de ensanche se podría expresar como se ve a continuación

$$fe = \frac{li}{2p_{i-1}} = \frac{li_{seguridad}}{li_{opt}}$$

En vista de esto, todas las ecuaciones a las que se han llegado en el desarrollo del método quedan de la siguiente forma

El número de divisiones por cada lado  $nd$  de cualquiera de los marcos vendrá dado por

**Ecuación 5-9**

$$nd = 2fe * \sqrt[n]{\frac{po}{pn}}$$

el número de puntos que se calculará en cada lado de cualquiera de los marcos será

$$nd + 1 = 2fe * \sqrt[n]{\frac{po}{pn}} + 1$$

el número de puntos que se tendrá cualquiera de los marcos será

**Ecuación 5-10**

$$npm = (2 * fe)^2 \left( \sqrt[n]{\frac{po}{pn}} + 1 \right)^2$$

el número de puntos total que se calculará en una simulación de  $N$  marcos será

**Ecuación 5-11**

$$npm_{total} = (2 * fe)^2 * n \left( \sqrt[n]{\frac{po}{pn}} + 1 \right)^2$$

el conjunto de pasos  $p^*$ , comenzando en  $p_0$ , y terminando en  $p_n$  hace optima la reducción de  $N$  marcos será

$$p^* = \{p_0, p_1, p_2, p_3, \dots, p_i * \dots * p_{n-1}, p_n\}$$

Es decir (Ecuación 5-6)

$$p^* = \left\{ \sqrt[n]{pn^0 * po^n}, \sqrt[n]{pn^1 * po^{n-1}}, \dots, \sqrt[n]{pn^i * po^{n-i}}, \dots, \sqrt[n]{pn^n * po^0}; \right\}$$

Y por último, el número de marcos o etapas  $et$  que hace óptimo el resultado del problema (Ecuación 5-8)

$$et = \ln \frac{po}{pn}$$

#### 5.2.2.4 Implementación del método

La implementación del método es inmediata. Para ello se crea la función *marcosoptimos*. Esta función, al margen de las variables necesarias para calcular el FS en punto, pide también un centro  $C(a,b)$  inicial, pide también el valor  $l_1$  y el valor  $p_n$ .

Con estos datos de entrada, se podrá dibujar el primer marco y los parámetros de la optimización.

Una vez se ha hecho esto ya se puede continuar con la simulación de forma sencilla. Se irá avanzando dentro de cada marco como se ha hecho anteriormente cuando solo existía uno. Dos bucles for, uno para cada coordenada rastrearán cada marco, buscarán el mínimo y se seguirá con el siguiente, así hasta terminar con la simulación.

Sin entrar demasiado en detalle se muestra el pseudocódigo de la función *marcosoptimos*.

Algoritmo 5-5: Función marcosoptimos

```

1  Funcion FSminGlobal <- marcosoptimos (inputsRastreo,ca,cb,l1,pn )
2      //Se calculan valores optimos: p,npt, et
3      // en función de los valores l1,pn
4      Para q <- 1 Hasta et Con Paso 1 Hacer
5          //Se calculan vertices del marco(q): a1,a2,b1,b2
6          // en función de los valores ca,cb,l1,pn
7          Para a<-a1 Hasta a2 Con Paso p(q) Hacer
8              Para b1<-b1 Hasta b2 Con Paso p(q) Hacer
9                  // Se invoca a la función Rastreo para calcular Fmin
10                 // y el nuevo centro ca,cb
11                 [FSmin,ca,cb] <- Rastreo (inputsRastreo)
12                 FSminmarco(q)<-FSmin
13
14                 Mientras C(ca,cb)= Bordes del marco(q) Hacer
15                     Si se entra en este bucle 10 veces Entonces
16                         break
17                         FinSi
18                         [FSmin,ca,cb] <- Rastreo (inputsRastreo)
19                         FinMientras
20
21             FinPara
22         FinPara
23     FinPara
24     FSminGlobal<-FSminmarco(end);
25 FinFuncion

```

Hasta ahora se ha supuesto que el marco inicial formado por el lado  $l_1$  encerraba en su interior el mínimo del problema. Sin embargo esto no tiene por qué ser así. De esta forma se añadirá en el pseudocódigo anterior un pequeño proceso que identifica cuando el centro encontrado se encuentra sobre los bordes del marco. En esta situación, no se dará paso al siguiente marco, sino que se repetirán operaciones con el mismo marco hasta que el centro deje de estar sobre los bordes del mismo.

Como no se quiere correr el riesgo de que el programa se quede atrapado en un bucle infinito y tampoco se quiere estar esperando demasiado tiempo-no hay que olvidar que se está tratando de optimizar- esta operación se restringe a ser realizada con un máximo de diez veces. Esta operación que se acaba de explicar es la recuadrada en el Algoritmo 5-5

A continuación se muestran dos simulaciones en las que se muestra cómo funciona el programa que se ha implementado.

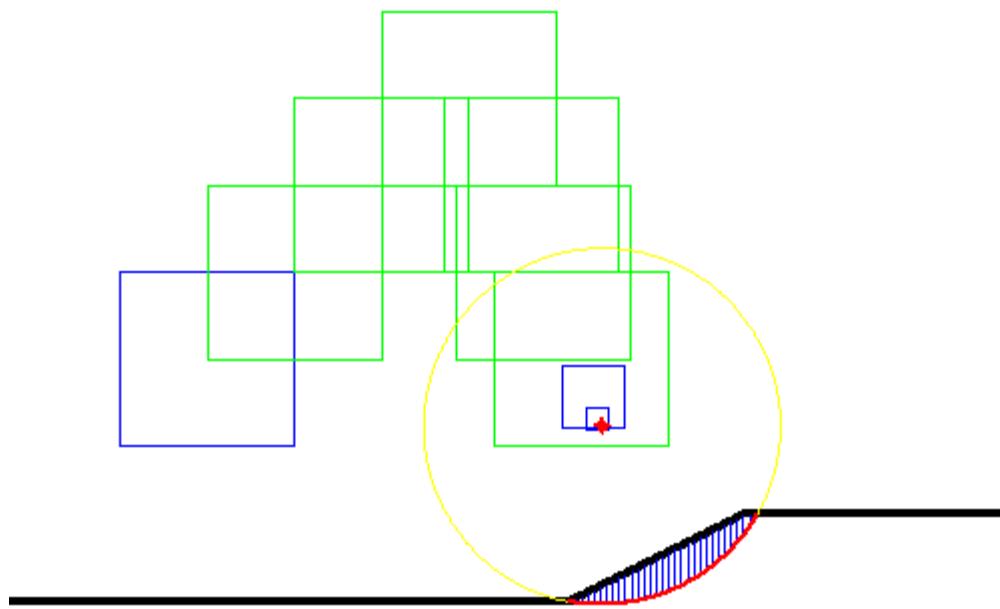
#### *Simulación 5.6*

En la siguiente tabla se muestra, por un lado las características de la simulación y por otro lado los resultados que se obtuvieron con ella. Además se adjunta más abajo una imagen del resultado de la simulación. Los cuadros azules son los marcos de las simulaciones principales y los cuadros verdes son aquellos rastreos que se tuvieron que hacer porque se detectó que el mínimo se situaba en los bordes del marco.

Se puede observar cómo el proceso ha sido capaz de detectar el FS mínimo, a pesar de que se partió de un marco que distaba bastante de dicho mínimo.

**Tabla 5-7: Valores de entrada y resultados de la simulación 5.6**

<b>Iniciación de la simulación</b>				<b>Resultado</b>					
<i>Ca(m)</i>	<i>Ca(m)</i>	<i>L1(m)</i>	<i>Pn(m)</i>	<i>FSmin</i>	<i>Ca(m)</i>	<i>Cb(m)</i>	<i>npt</i>	<i>et</i>	<i>Tiempo(s)</i>
-20	15	10	0.01	1.0945	1.878	10.06	832	6	16.43

**Ilustración 5-16: Resultados de la simulación 5.6**

A continuación se vuelve a realizar una simulación partiendo de otro centro inicial, con el ánimo de comprobar si localiza el mínimo.

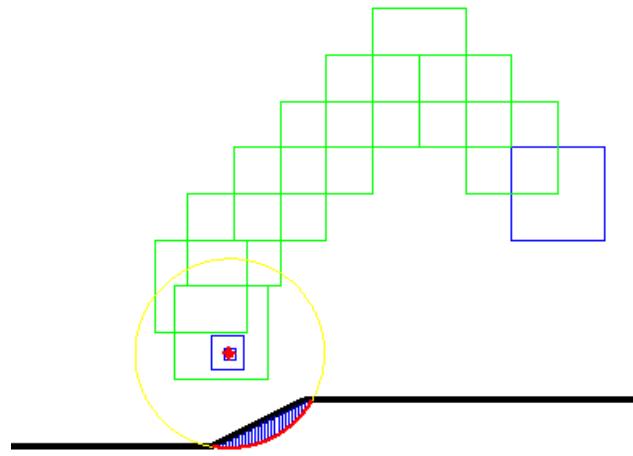
#### *Simulación 5.7*

**Tabla 5-8: Valores iniciales y resultados de la simulación 5.7**

Iniciación de la simulación				Resultado						
$Ca(m)$	$Ca(m)$	$L1(m)$	$Pn(m)$	$FSmin$	$Ca(m)$	$Cb(m)$	$npt$	$et$	Tiempo(s)	
37	27	10	0.01	1.0945	1.877	10.063	1024	6	20.8156	

Se observa partiendo de un punto situado al otro lado del talud, la solución es idéntica. La única variación se encuentra en el tiempo de simulación y el número de puntos analizados, que en este caso es mayor. Esto se debe a que este segundo punto necesitó más recuadros secundarios (verdes) para encontrar la solución.

Ahora que se sabe dónde se encuentra la solución , se pasa a hacer la misma operación, para mostrar cómo funciona el proceso cuando la solución se encuentra en su interior.

**Ilustración 5-17: Resultado de la simulación 5.7**

A continuación y con ánimo de mostrar cómo funciona el proceso cuando el mínimo está dentro se hace una simulación encima del centro del que se encontró el *FS* mínimo.

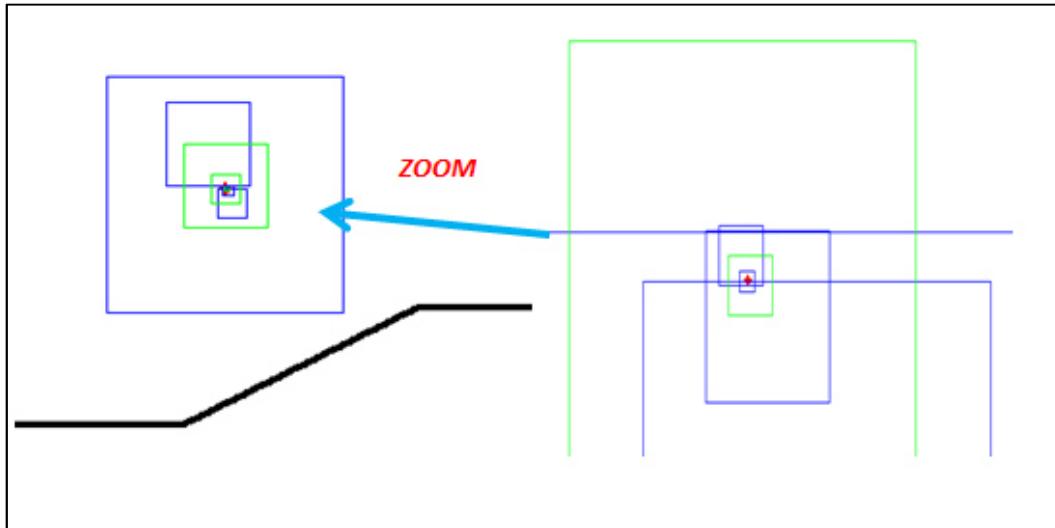
#### *Simulación 5.8*

**Tabla 5-9: Valores iniciales y resultados de la simulación 5.8**

Iniciación de la simulación				Resultado						
$Ca(m)$	$Ca(m)$	$L1(m)$	$Pn(m)$	$FSmin$	$Ca(m)$	$Cb(m)$	$npt$	$et$	$Tiempo(s)$	
1.877	10.063	10	0.01	1.0945	1.879	10.058	576	6	6.63	

Se observa que los resultados son casi idénticos que los anteriores. En este caso el número de simulaciones es bastante menor que en los casos anteriores, y por ende el tiempo de simulación.

Ilustración 5-18: Resultados de la simulación 5.8



Se observa en la imagen que, aunque el centro se encuentra en el interior del marco inicial, se hace necesario realizar algunos marcos secundarios mientras se acerca a la solución. A la derecha se realizó un zoom sobre la imagen izquierda para poder apreciar mejor el comportamiento interno de los marcos.

### 5.3 Conclusión

Se procede ahora a comparar una búsqueda de mínimo en un marco sin aplicar ninguna técnica de optimización y aplicando los dos métodos de optimización explicados en este capítulo. Ambas se harán por el método de Fellenius y se extrapolará los resultados a los demás métodos.

*Simulación 5.8***Tabla 5-10:** Valores iniciales de la simulación 5.8

Talud	Método	Marco	Precisión
3	Fellenius	10	0.01

**Tabla 5-11:** Resultados sin optimizar. Simulación 5.8

Sin optimizar			
FSP (nº de invocaciones)		Tiempo (s)	
628.254.627		3.2902x10 <sup>5</sup> s (91,39 horas)	
FS y centro, y radio críticos obtenidos			
FS	a	b	R
0.7203	0.6	13.5	13.6

**Tabla 5-12:** Resultados optimizando. Simulación 5.8

Con técnicas de optimización			
FSP (nº de invocaciones)		Tiempo (s)	
11457		6.3 s	
FS y centro, y radio críticos obtenidos			
FS	a	b	R
0,72069	0,622	13,487	13,5013

Se puede apreciar en los resultados que los valores son prácticamente idénticos, pudiendo mejorar los resultados de las técnicas con optimización simplemente disminuyendo la sensibilidad de la derivada y aumentando el factor de ensanche *fe*.

## 6. Comparación de resultados con programas de referencia

---

Con el objetivo de dar validez al programa que se ha escrito en este proyecto, se realizarán una serie de pruebas tomando como referencia los siguientes programas. Se usará para este cometido *el PFC de Jahn Kuhn, "Programa para cálculo de estabilidad de taludes mediante métodos de Fellenius, Bishop simplificado y Janbu simplificado", EIIC ULPGC 2012, tutores Francisco Chirino y David Greiner* y por otro lado la versión de prueba del *GEO5 - Estabilidad de suelos*.

Se tomarán como referencia la siguiente tabla de características de taludes

Tabla 6-1: Características de taludes de referencia

Talud	H (m)	B (m)	C (kPa)	Peso (kN/m <sup>3</sup> )	f <sub>i</sub> (°)
1	5	10	3	20	19.6
2	8,4	16,8	3	20	19.6
3	10	10	12	19	11
4	10	5	25	21	15
5	10	10	19	18	9
6	10	20	20	19	15
7	13	20	5	17	20

### 6.1 Factor de seguridad para una superficie de rotura determinada

Para estas simulaciones se tomarán los taludes uno y tres de la tabla de taludes. Para cada talud se definirán cinco puntos diferentes. Estos puntos aparecen en las tablas 6-2 y 6-3.

**Tabla 6-2: Puntos y radios para las simulaciones con el talud 1**

Talud 1	Circunferencia		
Puntos	a (m)	b (m)	R (m)
P1	3,49	11,31	11,59
P2	3,34	9,57	13,12
P3	0	5	6
P4	11	9	7
P5	3,5	4	4

**Tabla 6-3: Puntos y radios para las simulaciones con el talud 3**

Talud 3	Circunferencia		
Puntos	a (m)	b (m)	R (m)
P1	2	12	11
P2	-0,5	19.61	17.64
P3	0	15	17
P4	0,9	12,33	11.66
P5	1,65	10,69	11,08

En cada punto se buscará el *FS* por los tres métodos que se han explicado y con un número de rebanadas igual a cien. Este proceso se llevará a cabo con el programa diseñado en este proyecto y con la versión de prueba del GEO5. Los resultados son los que se muestran en la tabla 6-4.

**Tabla 6-4: Resultados de la comparación del FS para una superficie de rotura**

Talud	Método	Programa	P1	P2	P3	P4	P5
1	Fellenius	Matlab	1,2159	1.5684	1.6832	4.0818	1,3767
		Geo5	1,3	1,57	1,69	4,09	1,4
	Bishop S.	Matlab	1,3033	1.8402	1.8847	4.498	1,5861
		Geo5	1,3	1,85	1,9	4,51	1,61
	M-P	Matlab	1,3082	1.8635	1.8993	4.5124	1,6069
		Geo5	1,3	1,85	1,89	4,5	1,6
Talud	Método	Programa	P1	P2	P3	P4	P5
3	Fellenius	Matlab	0.77204	0.8775	0.83208	0,76153	0,74985
		Geo5	0,78	0.88	0.84	0,76	0,76
	Bishop S.	Matlab	0.79348	0.88743	0.89845	0,77948	0,78074
		Geo5	0.80	0.89	0.90	0,78	0,79
	M-P	Matlab	0.7932	0.8868	0.90279	0,77908	0,78101
		Geo5	0.82	0.89	0.90	0,78	0,82

La versión demo del GEO5 presenta los resultados hasta el segundo decimal. Se puede observar que los resultados son prácticamente idénticos.

## 6.2 Factor de seguridad mínimo global

A continuación se exponen los resultados que se obtuvieron al buscar el *FS* mínimo global de cada talud (ver Tabla 6-1: Características de taludes de referencia Tabla 6-1: características de taludes de referencia). Para ello se muestran una tabla por talud, donde cada cual contiene todos los resultados.

Para cada talud se procedió de idéntica forma. En primer lugar se hicieron las simulaciones usando el método de Fellenius, ya que es el método que menos tarda en obtener una solución. Se hizo una búsqueda optimizada con los siguientes valores:

**Tabla 6-5: Valores iniciales I**

Lado (m)	10
Precisión	0.01
Centro [a,b]	[0,15]

Luego se realizó el método de Bishop, el cual se hizo usando como referencia la solución encontrada con el método de Fellenius. Este se realizó como indica la siguiente tabla

Tabla 6-6: Valores iniciales II

Lado (m)	5
Precisión	0.01
Centro [a,b]	Centro Fellenius

El método de Morgenstern-Price se hizo de manera idéntica que el método de Bishop, pero esta vez el recuadro se centró en la solución obtenida en Bishop. El método M-P y el método de Bishop suelen dar valores muy parecidos, por ello las características de las simulaciones fueron

Tabla 6-7: Valores iniciales III

Lado	1
Precisión	0.01
Centro [a,b]	Centro Bishop

Los resultados de dichas simulaciones se muestran en las tablas que siguen

Tabla 6-8: FS crítico con talud 1

Talud 1					
Método	Programa	FS	a (m)	b (m)	R (m)
Felle	Matlab	1,0941	1,8852	10,0521	10,2274
	GEO5	1,1	1,86	10,09	10,26
	ULPGC	1,126			
Bishop	Sergio	1,1544	0,95971	12,4176	12,4546
	Geo5	1,16	0,88	12,66	12,69
	Referencia	1,191			
M-P	Sergio	11.559	0,9014	125.614	125.936
	Geo5	1,15	1,01	12,31	12,35
	Referencia	1,192			

Tabla 6-9: FS crítico con talud 2

Talud 2					
Método	Programa	FS	a (m)	b (m)	R (m)
Felle	Matlab	0,97326	2,042	18,6156	18,7272
	GEO5	0,97	1,95	18,86	18,96
	ULPGC	0,972			
Bishop	Sergio	1,02	0,14967	23,1737	23,1741
	Geo5	1,02	0,11	23,27	23,27
	Referencia	1,035			
M-P	Sergio	1,0211	0,034	23,4323	23,4324
	Geo5	1,02	0,07	23,5	23,5
	Referencia	1,034			

Tabla 6-10: FS crítico con talud 3

Talud 3					
Método	Programa	FS	a (m)	b (m)	R (m)
Felle	Matlab	0,72069	0,622	13,487	13,5013
	GEO5	0,72	0,53	13,71	13,72
	ULPGC	0,731			
Bishop	Sergio	0,74136	0,035	14,335	14,335
	Geo5	0,74	-0,06	14,54	14,54
	Referencia	0,75			
M-P	Sergio	0,74076	0	14,391	14,391
	Geo5	0,75	-0,33	15,67	15,67
	Referencia	0,749			

Tabla 6-11: FS crítico con talud 4

Talud 4					
Método	Programa	FS	a (m)	b (m)	R (m)
Felle	Matlab	0,97	-4,068	13,821	14,4072
	GEO5	0,97	-4	13,92	14,98
	ULPGC	0,976			
Bishop	Sergio	0,94849	-2,0283	10,0013	10,2049
	Geo5	0,96	-3,6	13,43	13,9
	Referencia	0,966			
M-P	Sergio	0,95989	-2,75	11,6	11,9225
	Geo5	0,99	-5,51	15,94	16,85
	Referencia	1,013			

Tabla 6-12: FS crítico con talud 5

Talud 5					
Método	Programa	FS	a (m)	b (m)	R (m)
Felle	Matlab	0,93207	1,757	13,596	13,7091
	GEO5	0,93	1,68	13,91	14,01
	ULPGC	0,941			
Bishop	Sergio	0,94733	1,495	13,702	13,7833
	Geo5	0,95	1,31	14,27	14,33
	Referencia	0,959			
M-P	Sergio	0,9472	1,47	13,7977	13,8758
	Geo5	0,96	0,96	15,36	15,39
	Referencia	0,96			

Tabla 6-13: FS crítico con talud 6

Talud 6					
Método	Programa	FS	a (m)	b (m)	R (m)
Felle	Matlab	1,4869	6,971	16,509	18,1754
	GEO5	1,49	6,87	16,81	18,45
	ULPGC	1,576			
Bishop	Sergio	1,5703	6,219	19,309	20,2858
	Geo5	1,57	6,2	19,6	20,56
	Referencia	1,647			
M-P	Sergio	1,5718	6,1543	19,4698	20,4194
	Geo5	1,57	6,29	19,28	20,28
	Referencia	1,66			

Tabla 6-14: FS crítico con talud 7

Talud 7					
Método	Programa	FS	a (m)	b (m)	R (m)
Felle	Matlab	0,85791	0,899	23,063	23,0805
	GEO5	0,86	0,98	22,96	22,98
	ULPGC	0,861			
Bishop	Sergio	0,9033	-1,797	28,384	28,4408
	Geo5	0,9	-2,03	28,99	29,06
	Referencia	0,916			
M-P	Sergio	0,90408	-1,9905	28,731	28,8
	Geo5	0,9	-2,05	29,14	29,21
	Referencia	0,913			

## 7. Resolución de un caso práctico

Se quiere saber cómo varía el  $FS$  (de los tres métodos que se han implementado en este proyecto) de un talud, con las características del terreno mostradas en la tabla 7-1, cuando su altura aumenta su tamaño de 10 metros hasta 100 metros en saltos de 10. En todas estas variaciones, la pendiente del talud permanece constante.

Las características del talud son las que se muestran a continuación

**Tabla 7-1: Características del talud de un caso práctico**

Talud	C (kPa)	Peso(kN/m^2)	F <sub>i</sub> (°)	Pendiente (°)
Caso práctico	3	20	19.6	26.5651

La tabla 7-2 muestra las características del lado del marco, la precisión y el centro que se utilizó para cada talud, de tal forma que las simulaciones podrían ser repetidas por otro usuario y llegar a los mismos resultados.

**Tabla 7-2: Valores iniciales**

Altura (m)	lado (m)	precisión(m)	Centro C(a,b) (m)
10	40	0.01	[10,40]
20	40	0.01	[2,20]
30	40	0.01	[-0,50]
40	70	0.01	[-10,90]
50	100	0.01	[-10,125]
60	100	0.01	[-20,170]
70	150	0.01	[-40,250]
80	150	0.01	[-60,300]
90	300	0.01	[-80,350]
100	400	0.01	[-105,445]

Tabla 7-3: Caso práctico; H=10

Talud Caso práctico 1. H=10 m; B=20m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,9424	1,944	22,965	23,0471
Bishop	0,98507	-0,462	28,649	28,6527
M-P	0,98611	-0,609	28,978	28,9844

Tabla 7-4: Caso práctico; H=20

Talud Caso práctico 2. H=20 m; B=40m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,85297	-0,48767	53,6263	53,6286
Bishop	0,88208	-6.557	67,143	67,4624
M-P	0,88278	-6,89	67,855	68,2039

Tabla 7-5: Caso práctico; H=30

Talud Caso práctico 3. H=30 m; B=60m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,81831	-5,2286	88,7516	88,9055
Bishop	0,84131	-15,6083	111,2811	112,3704
M-P	0,84131	-16,119	112,3637	113,5139

Tabla 7-6: Caso práctico; H=40

Talud Caso práctico 4. H=40 m; B=80m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,79918	-11,5804	126,99	127,5169
Bishop	0,81854	-26,819	159,605	161,8426
M-P	0,81898	-27,522	161,077	163,4113

Tabla 7-7: Caso práctico; H=50

Talud Caso práctico 5. H=50 m; B=100m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,78682	-19,5767	168,5113	169,6447
Bishop	0,80201	-39,844	211,469	215,1899
M-P	0,80409	-40,723	213,302	217,1546

Tabla 7-8: Caso práctico; H=60

Talud Caso práctico 6. H=60 m; B=120m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,77806	-29,55	214,0323	216,0626
Bishop	0,79318	-54,349	266,3557	271,844
M-P	0,79332	-55,5077	268,6973	274,3708

Tabla 7-9: Caso práctico; H=70

Talud Caso práctico 7. H=70 m; B=140m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,77148	-38,9386	257,9636	260,8858
Bishop	0,78524	-69,4948	321,8794	329,296
M-P	0,78536	-70,9351	325,1594	332,8069

Tabla 7-10: Caso práctico; H=80

Talud Caso práctico 8. H=80 m; B=160m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,76685	-36,6917	277,3963	279,8124
Bishop	0,77899	-86,1105	381,025	390,6342
M-P	0,7791	-88,3127	385,5847	395,5688

Tabla 7-11: Caso práctico; H=90

Talud Caso práctico 9. H=90 m; B=180m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,76216	-60,914	353,028	358,2447
Bishop	0,77392	-105,4593	445,8337	458,1358
M-P	0,77403	-106,5747	447,7187	460,2284

Tabla 7-12: Caso práctico; H=100

Talud Caso práctico 10. H=100 m; B=200m				
	FS	a (m)	b (m)	R (m)
Fellenius	0,75871	-75,2457	407,3496	414,241
Bishop	0,76972	-127,7333	514,2044	529,259
M-P	0,76982	-126,3293	512,6206	527,9573

Si se representa en un eje coordenado los resultados se tiene

Ilustración 7-1: Resultados del caso práctico. Método de Fellenius

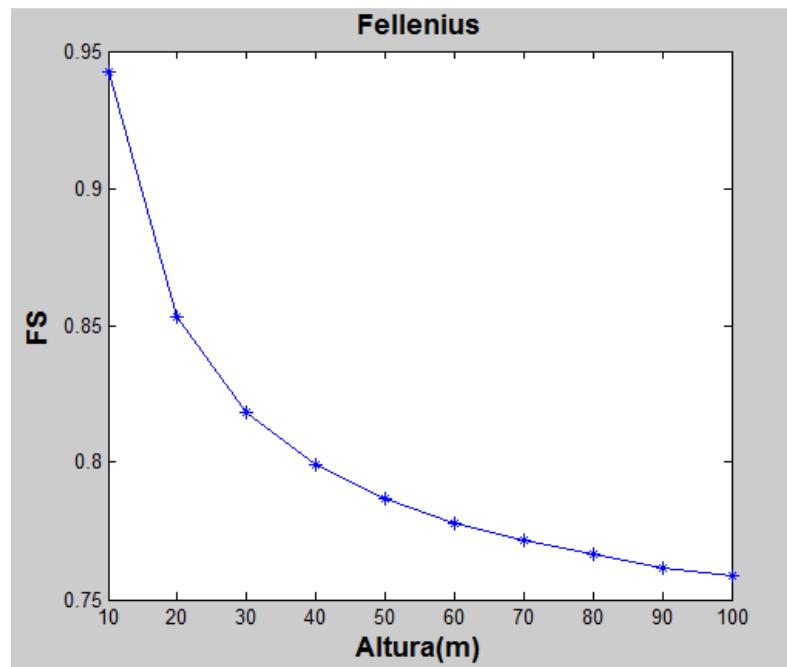


Ilustración 7-2: Resultado del caso práctico. Método de Bishop simplificado

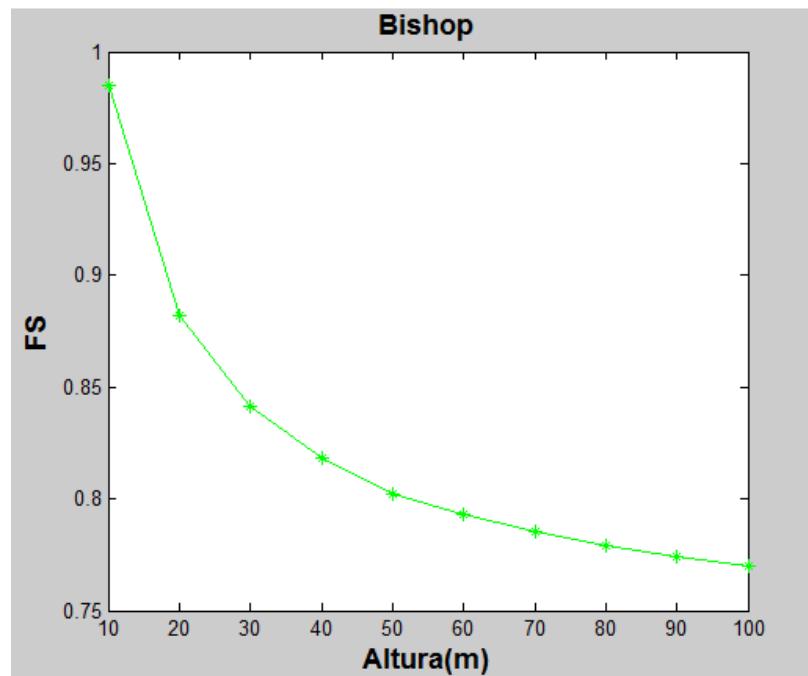


Ilustración 7-3: Resultado del caso práctico. Método de Morgenstern-Price:

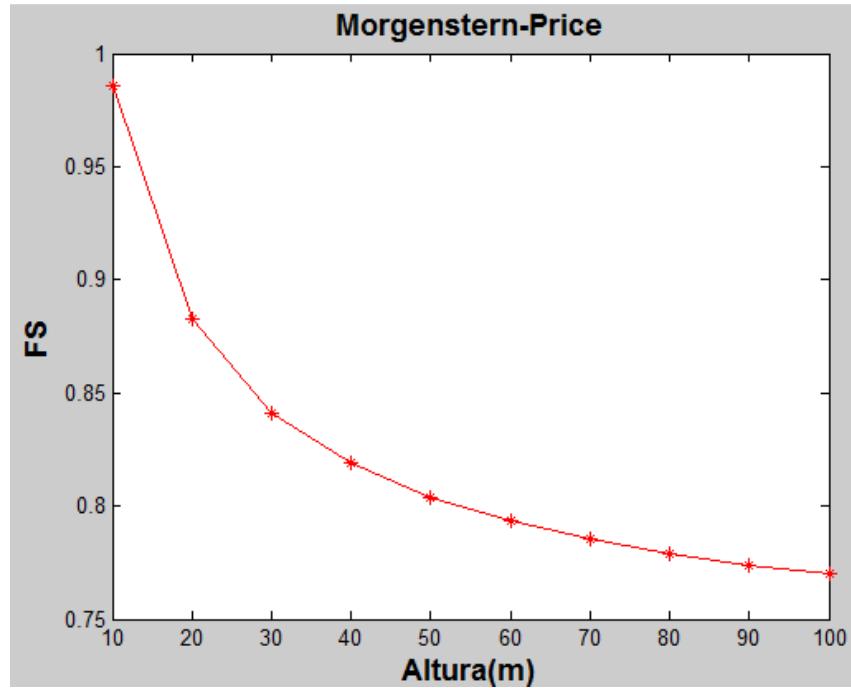
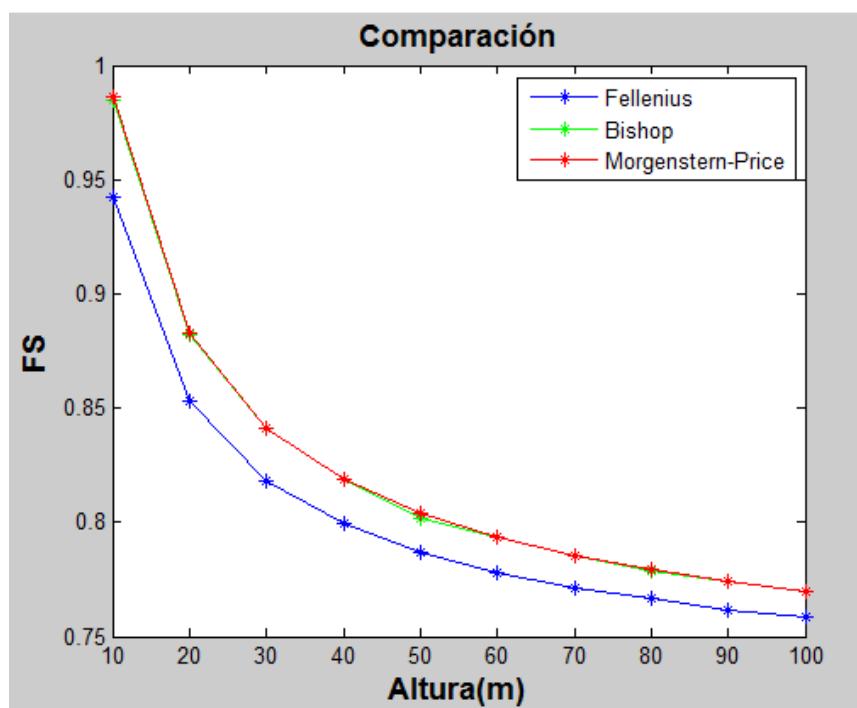


Ilustración 7-4: Resultado del caso práctico. Comparación de los métodos



Se observa que a medida que aumenta el tamaño del talud, el FS desciende de forma hiperbólica. Los tres métodos obtienen resultados similares, siendo el de Fellenius el más pesimista. Por otra parte, sorprende lo similares que son los resultados obtenidos por el método de Bishop y el método de Morgenstern-Price.

## 8. Conclusiones y líneas futuras de trabajo

### 8.1 Conclusión

Se definió como objetivo fundamental del proyecto la implementación de los métodos de Morgenstern-Price utilizando como referencia el artículo *A concise algorithm for computing the factor of safety using the Morgenstern–Price method*". *Canadian Geotechnical Journal*, Autores: D.Y. Zhu, C.F. Lee, Q.H. Qian, and G.R. Chen (2005). También se estableció como objetivos la implementación de los métodos de Fellenius y los métodos de Bishop simplificado.

En el capítulo tres se implementó el método de Morgenstern-Price, el cual se vio que es bastante más complejo de implementar que los demás. Se pudo observar que hay valores del método de Morgenstern-Price que no tienen una solución coherente. Estos valores coinciden con los obtenidos en los programas comerciales y ya fueron mencionados por los mismos Morgenstern y Price (Morgenstern & Price, 1965). La técnica que se usó en un principio para estos puntos fue la de variar la función  $f(x)$  hasta que se encontrara una solución coherente, sin embargo no da muy buenos resultados, ya que estos puntos no suelen converger en un resultado coherente, independientemente de la función usada. Se recomienda ignorar estos puntos.

En el capítulo cuatro podría considerarse el proyecto concluido, ya que se consigue el objetivo propuesto, que es el de proporcionar el factor de seguridad crítico de un talud por los métodos mencionados. El problema es que los tiempos de búsqueda son demasiado grandes, y por esa razón se desarrolló en el capítulo cinco los métodos optimización que permitieron mejorar los tiempos de búsqueda de dicho factor de seguridad. En la tabla siguiente se puede apreciar el orden de magnitud de la mejora en tiempos conseguida

**Tabla 8-1: Comparación de tiempos y simulaciones**

<b>Sin optimizar</b>		<b>Optimizando</b>	
<b>FSP(nº iteraciones)</b>	<b>Tiempo(horas)</b>	<b>FSP(nº iteraciones)</b>	<b>Tiempo(seg)</b>
628.254.627	91,39 horas	11457	6.3 segundos

Como se ve en la tabla 8-1 se consiguen reducir los tiempos de búsqueda. Finalmente, los resultados fueron comparados con los resultados obtenidos por otros programas de referencia, pudiéndose apreciar la validez de dichos resultados.

## 8.2 Líneas futuras

---

Los métodos que se han implementado han partido de la hipótesis de que las únicas fuerzas actuantes sobre el talud era la de su propio peso. En base a esto se puede establecer como futura línea de trabajo la incorporación los siguientes esfuerzos desestabilizadores:

- Presión intersticial del agua
- Cargas externas al propio talud.
- Esfuerzos relacionados con actividad sísmica

Otra de las hipótesis de partida fue la de considerar el terreno homogéneo y por ello otra línea futura de trabajo es precisamente implementar la posibilidad de definir distintas zonas del terreno con distintas características.

Se comentó en las conclusiones que el método de Morgenstern-Price no converge en algunas superficie de rotura. Por ello, se podría realizar una investigación que trate de justificar este resultado y por otra parte, desarrollar algún método que permita dar una solución a estas situaciones de no convergencia.

Finalmente, aunque este proyecto tiene un capítulo dedicado a la optimización, los tiempos de cálculo con el método de Morgenstern-Price siguen siendo excesivos en comparación con algunos programas comerciales, y por ello se podría realizar un estudio sobre cómo mejorar los tiempos de simulación de dicho método.

Además, se propone, investigar y desarrollar la implementación de métodos de optimización metaheurísticos al proceso de búsqueda.

## 9. Bibliografía

---

- AYALA CARCEDO, F. J., & ANDREU POSSE, F. J. (1987). Manual de Ingeniería de Taludes. En F. J. AYALA CARCEDO, & F. J. ANDREU POSSE, *Manual de Ingeniería de Taludes*. Madrid: Instituto Geológico Minero de España.
- Gonzales, M. (2001). *El Terreno*. Barcelona: Ediciones UPC.
- Iglesias, C. (1997). Mecánica de Suelos. En C. Iglesias, *Mecánica de Suelos* (pág. 24). Madrid: Editorial SÍNTESIS.
- Labad, F. M. (2007). *Mecánica de suelo y cimentaciones Vol.1*. Madrid: Fundación Escuela de la Edificación.
- Moore, H. (s.f.). *Matlab para ingenieros*. Pearson Prentice Hall.
- Morgenstern, N. R., & Price, V. E. (1967). A numerical method for solving the equation of stability on general slip surfaces.
- Ortuño Abad, L. (2004). Curso de estabilidad de taludes en suelo. *Curso de Geotecnica para estructuras*. Sevilla: Uriel y Asociados, S.A.
- Perez, E. (2005). *Estabilidad de taludes*. Barcelona: Universitat Politecnica de Catalunya.
- Ramirez, P., & Alejano, L. (2009). *Mecánica de rocas: Fundamento e ingeniería de taludes*. Madrid.
- Whitlow, R. (1994). Mecánica de Suelos. En R. Whitlow, *Mecánica de Suelos* (pág. 1). México: Compañía Editorial Continental.
- Zhu, D., Lee, C., Qian, Q., & Chen, G. (2005). *A concise algorithm for computing the factor of safety using the Morgenstern-Price method*. Hong Kong: Canadian Geotechnical Journal.

## *10. Anexo*

---



