APÉNDICE A. RESULTADOS DEL ANÁLISIS MEDIANTE EQUILIBRIO LÍMITE DE TALUDES SOMETIDOS A PRESIONES EN SU SUPERFICIE

A.1 GRÁFICOS PARA LA OBTENCIÓN DEL VALOR DE LA PRESIÓN SOBRE UN TALUD PARA GARANTIZAR UN CIERTO COEFICIENTE DE SEGURIDAD. CASO DE TALUD INDEFINIDO

Se presentan los gráficos que proporcionan los valores de la presión necesaria para un cierto coeficiente de seguridad, en función de la cohesión, el ángulo de rozamiento interno, la inclinación del talud y las condiciones de filtración, para el caso de talud indefinido.

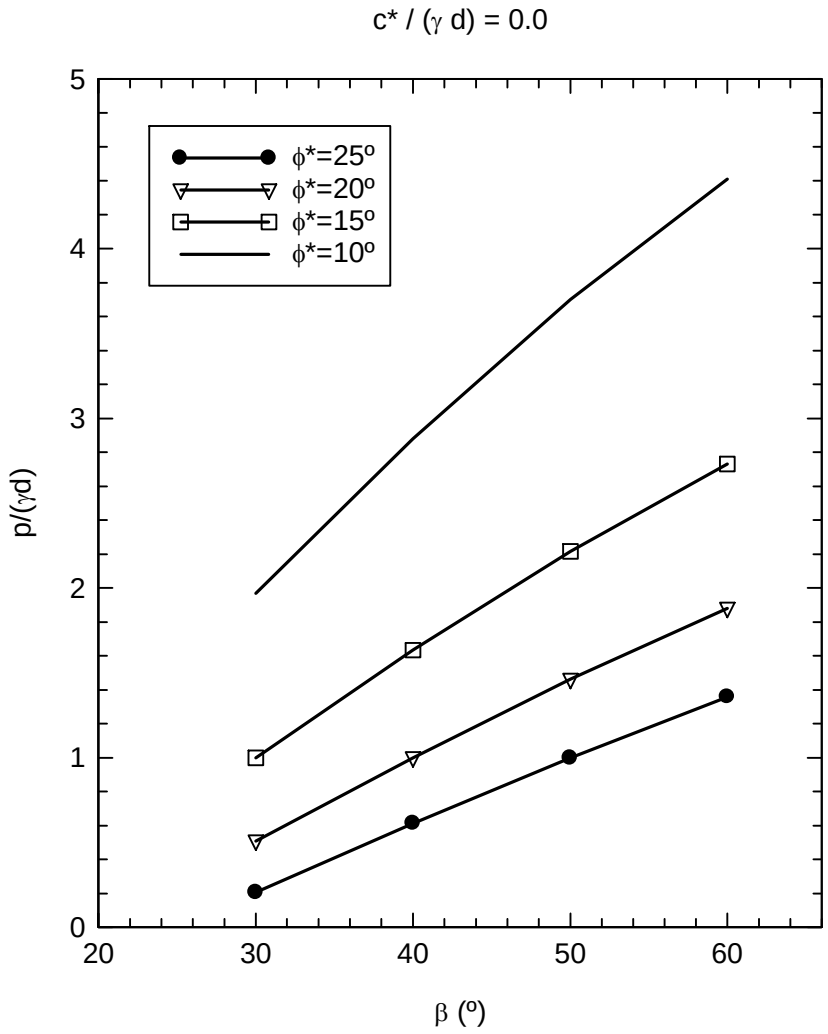


Figura A.1. Obtención de la presión en la superficie del talud para c*/(γ·d) = 0.0. Talud seco

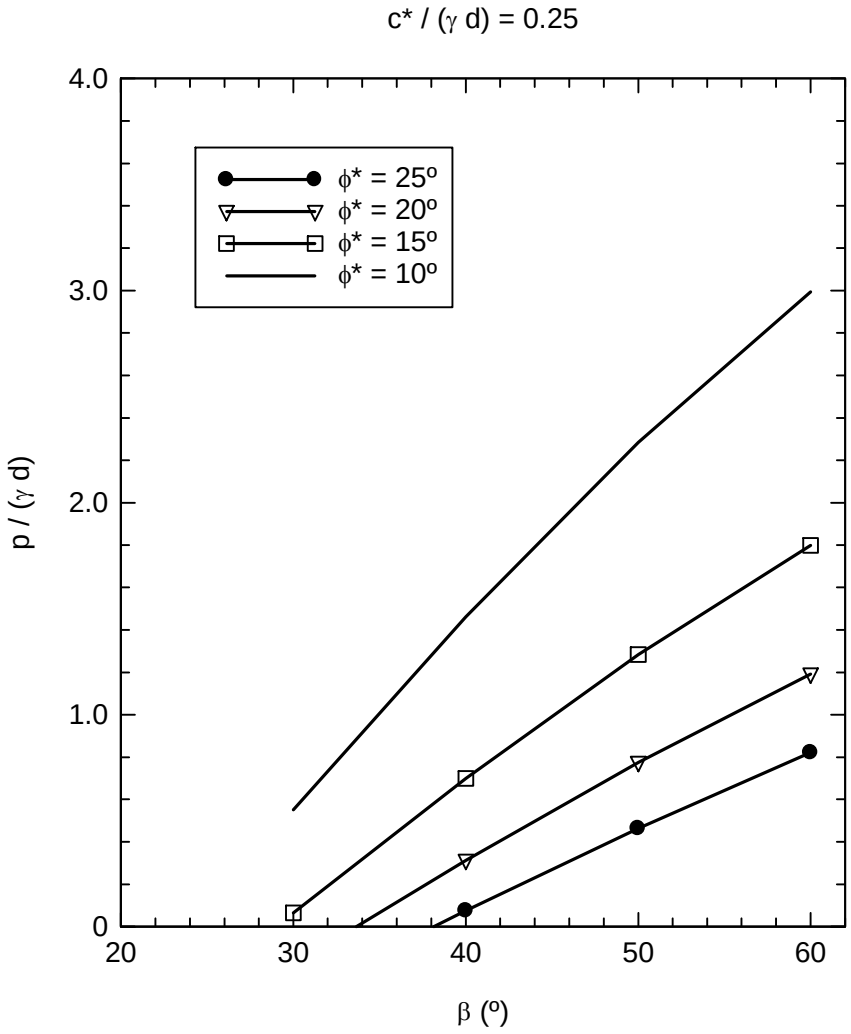


Figura A.2. Obtención de la presión en la superficie del talud para c*/(γ·d) = 0.25. Talud seco

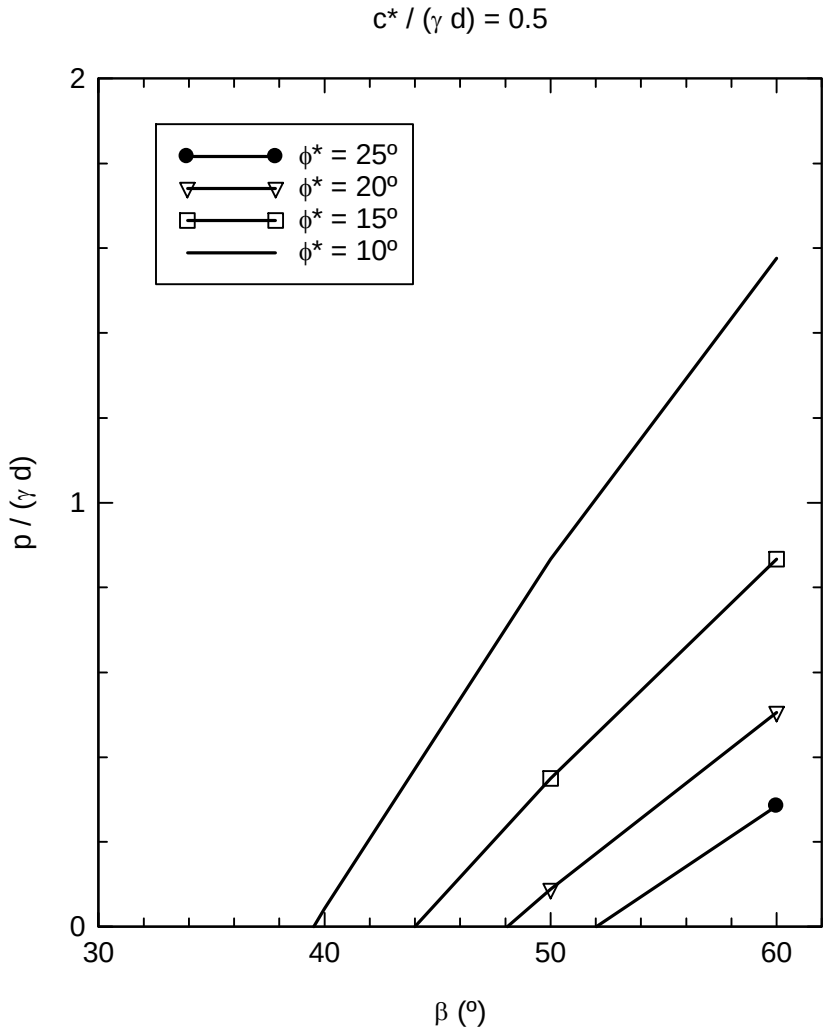


Figura A.3. Obtención de la presión en la superficie del talud para c*/(γ·d) = 0.5. Talud seco

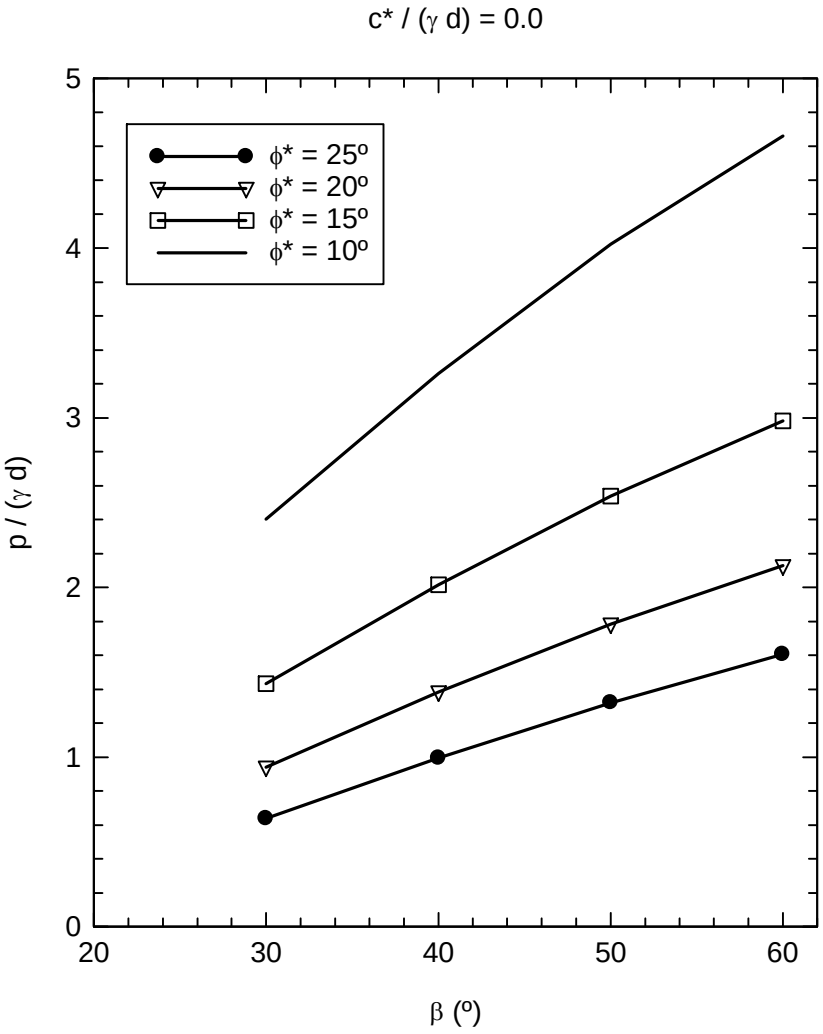


Figura A.4. Obtención de la presión en la superficie del talud para c*/(γ·d) = 0.0. Talud con filtración paralela

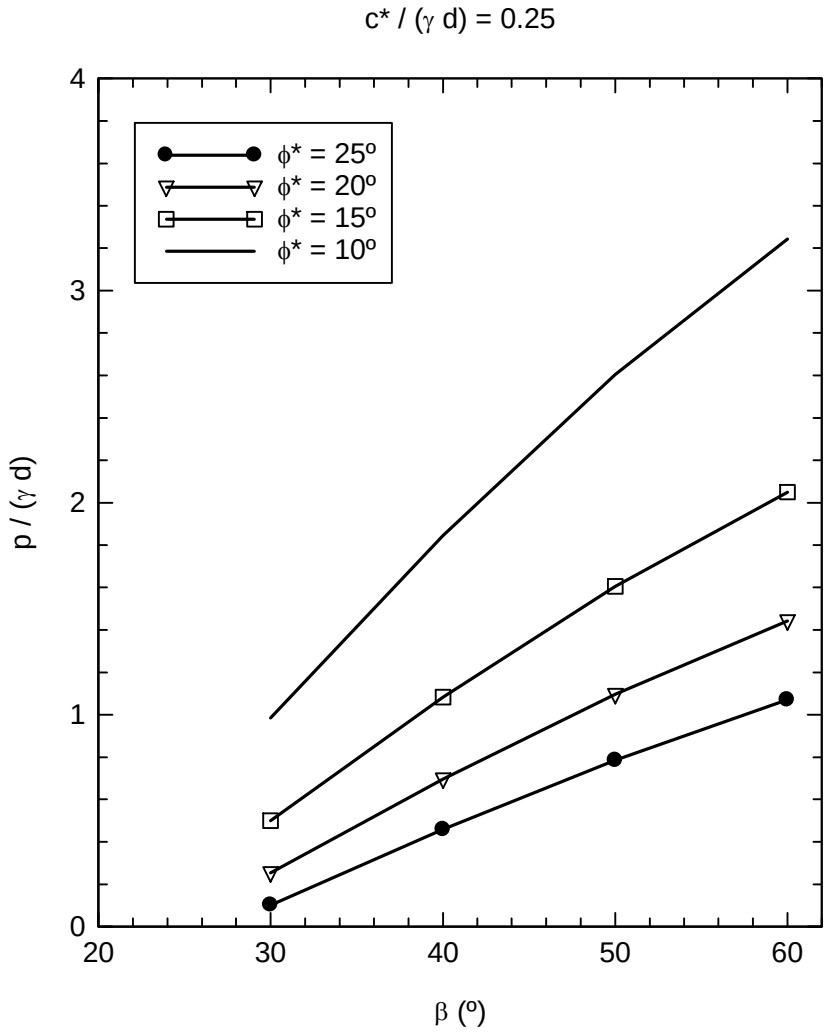


Figura A.5. Obtención de la presión en la superficie del talud para c*/(γ·d) = 0.25. Talud con filtración paralela

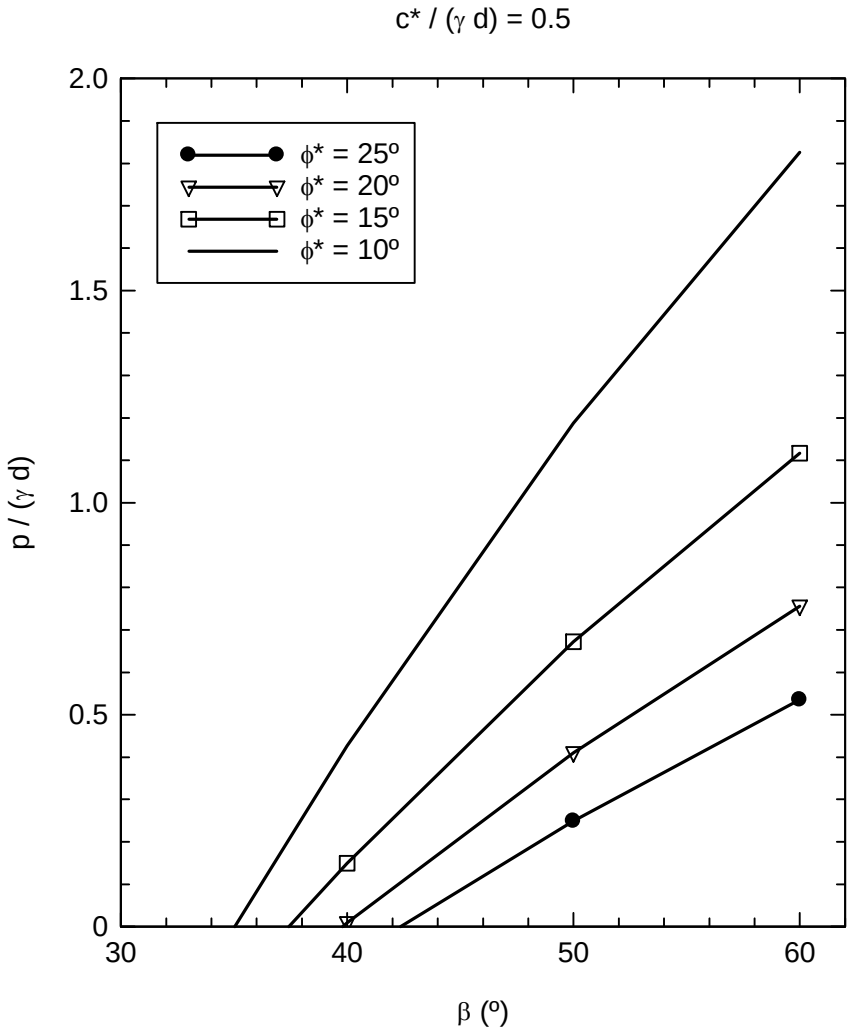


Figura A.6. Obtención de la presión en la superficie del talud para c*/(γ·d) = 0.5. Talud con filtración paralela

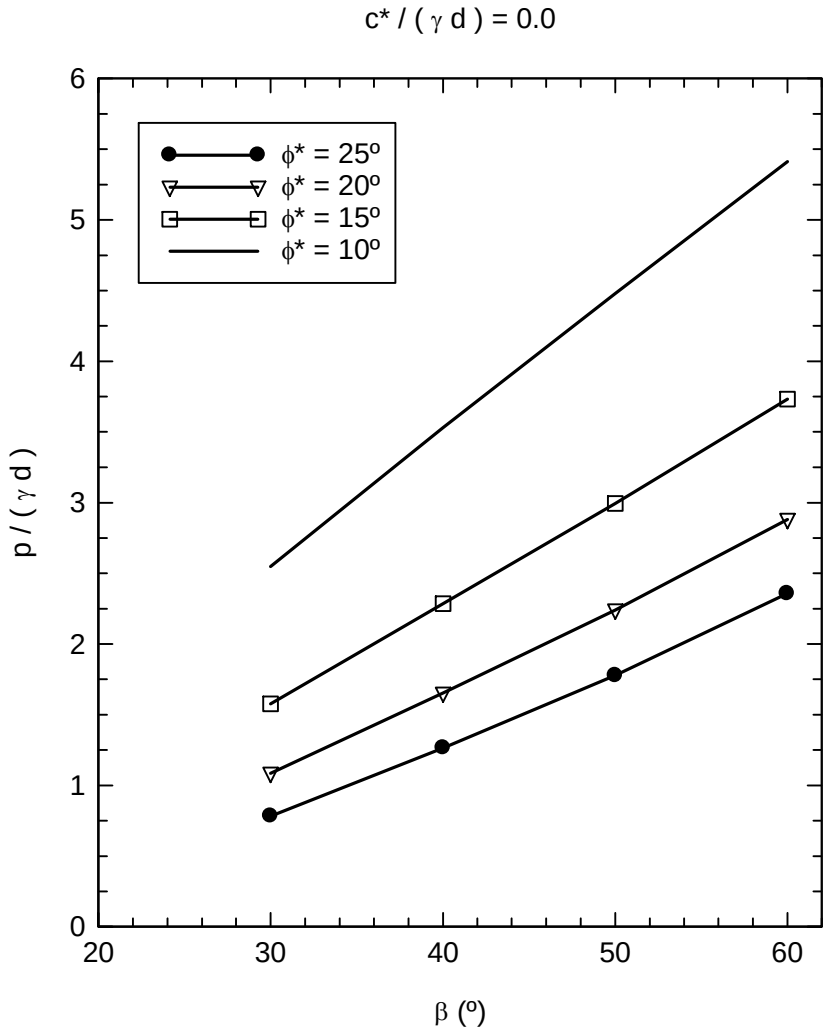


Figura A.7. Obtención de la presión en la superficie del talud para c*/(γ·d) = 0.0. Talud con filtración horizontal

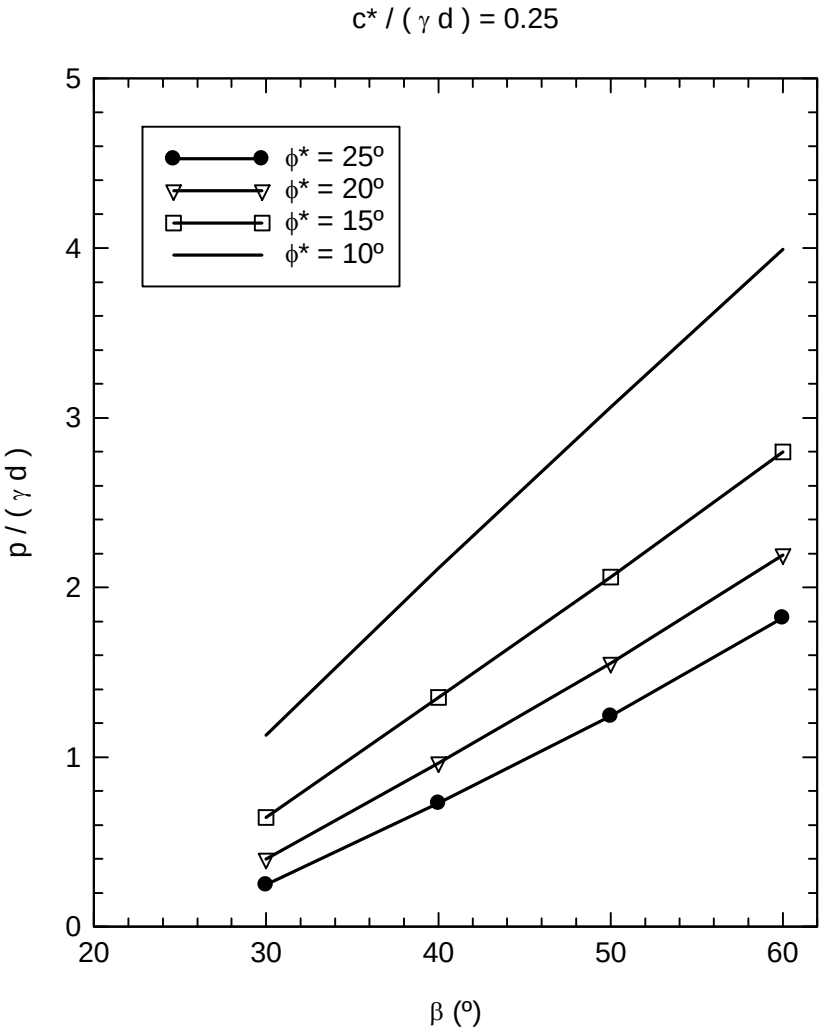


Figura A.8. Obtención de la presión en la superficie del talud para c*/(γ·d) = 0.25. Talud con filtración horizontal

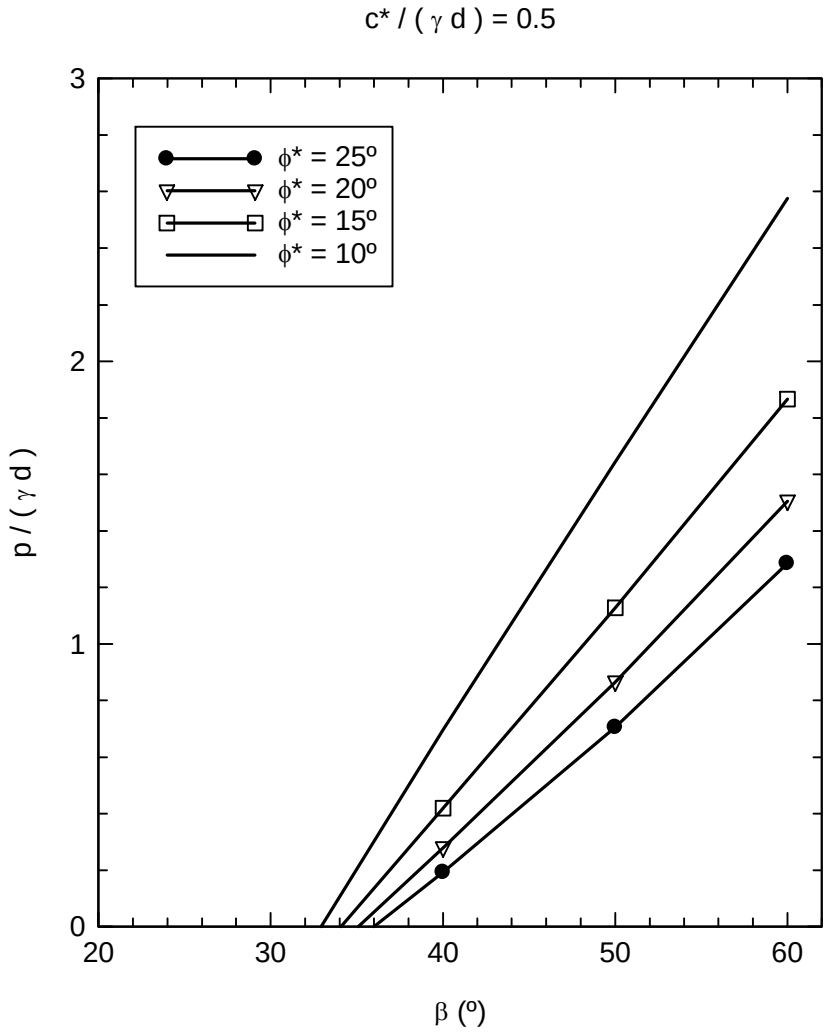


Figura A.9. Obtención de la presión en la superficie del talud para c*/(γ·d) = 0.5. Talud con filtración horizontal

A.2 EXPRESIONES PARA EL CÁLCULO DEL VALOR DE LA PRESIÓN NECESARIA SOBRE UN TALUD PARA GARANTIZAR UN CIERTO COEFICIENTE DE SEGURIDAD. CASO DE TALUD DE ALTURA FINITA

Se presentan las expresiones que proporcionan, para el caso de talud finito, los valores de las tensiones efectivas normales en los planos de deslizamiento, y de la presión normal que es necesario que transmita la red para conseguir un cierto coeficiente de seguridad al deslizamiento. Estas expresiones se presentan para diferentes casos, según cuales de las tensiones efectivas normales en los planos de deslizamiento sean negativas.

CASO I

Al resolver las ecuaciones se obtiene: N'_1, N'_2 y $N'_3 > 0$.

$$N'_2 = \frac{c'^* \cdot \left\{ d \cdot tg\phi'^* - (i-1) \cdot s - \frac{d}{tg\beta} \right\} + G_1 \cdot \left\{ sen\beta - \cos\beta \cdot tg\phi'^* \right\}}{1 - \left(tg\phi'^* \right)^2} + \\ + \frac{- s \cdot \left\{ tg\phi'^* + tg\delta \right\} \cdot \sum_{j=1}^{j=i-1} p_j + u_1 \cdot tg\phi'^* - u_2}{1 - \left(tg\phi'^* \right)^2} \qquad \text{(A.1)}$$

$$N'_1 = G_1 \cdot \cos\beta + \sum_{j=1}^{j=i-1} p_j \cdot s - c'^* \cdot d - N'_2 \cdot tg\phi'^* - u_1$$

(A.2)

$$p_i = \frac{N'_2 \cdot \left\{ 1 - k \cdot tg\phi'^* \right\} + G_2 \cdot \left\{ sen\beta - k \cdot \cos\beta \right\} - c'^* \cdot \left\{ s + 2 \cdot k \cdot d \right\}}{s \cdot \left(k + tg\delta \right)} + \\ + \frac{u_3 \cdot \left\{ k \cdot \cos\lambda - sen\lambda \right\} + u_2}{s \cdot \left(k + tg\delta \right)} \qquad \text{(A.3)}$$

$$N'_3 = \left\{\frac{k}{k+tg\delta}\right\}\cdot\left\{\frac{c'^*\{2\cdot d\cdot tg\delta - s\} + N'_2\cdot\{1+tg\delta\cdot tg\phi'^*\}}{sen\lambda+\cos\lambda\cdot tg\phi'^*} + \right.$$

$$\left. + \frac{G_2\cdot\{sen\beta+tg\delta\cdot\cos\beta\}+u_2-u_3\cdot\{sen\lambda+tg\delta\cdot\cos\lambda\}}{sen\lambda+\cos\lambda\cdot tg\phi'^*}\right\} \quad \text{(A.4)}$$

CASO II

Al resolver las ecuaciones se obtiene: N'_1 y $N'_2 > 0$ y $N'_3 < 0$.

$$N'_2 = \frac{c'^*\cdot\left\{d\cdot tg\phi'^* - (i-1)\cdot s - \frac{d}{tg\beta}\right\} + G_1\cdot\{sen\beta - \cos\beta\cdot tg\phi'^*\}}{1-(tg\phi'^*)^2} +$$

$$+ \frac{-s\cdot\{tg\phi'^*+tg\delta\}\cdot\sum_{j=1}^{j=i-1}p_j + u_1\cdot tg\phi'^* - u_2}{1-(tg\phi'^*)^2} \quad \text{(A.5)}$$

$$N'_1 = G_1\cdot\cos\beta + \sum_{j=1}^{j=i-1}p_j\cdot s - c'^*\cdot d - N'_2\cdot tg\phi'^* - u_1 \quad \text{(A.6)}$$

$$p_i = \frac{u_3\cdot(s\cdot\cos\lambda + d\cdot sen\lambda) - N'_2\cdot(d+s\cdot tg\phi'^*) - c'^*\cdot d\cdot s}{s^2 - d\cdot s\cdot tg\delta} +$$

$$+ \frac{-G_2\cdot(d\cdot sen\beta + s\cdot\cos\beta) - u_2\cdot d}{s^2 - d\cdot s\cdot tg\delta} \quad \text{(A.7)}$$

$$F_3 = \frac{c'\cdot s}{u_2 + N'_2 + G_2\cdot sen\beta - u_3\cdot sen\lambda - p_i\cdot s\cdot tg\delta} \quad \text{(A.8)}$$

CASO III

Al resolver las ecuaciones se obtiene: $N'_1 > 0$ y $N'_2 < 0$.

$$N'_1 = \frac{G_1 \cdot sen\beta - u_2 - c'^{*} \cdot \left\{(i-1)\cdot s + \frac{d}{tg\beta}\right\} - \sum_{j=1}^{j=i-1} p_j \cdot tg\delta \cdot s}{tg\phi'^{*}} \qquad (A.9)$$

$$F_2 = \frac{c' \cdot d}{G_1 \cdot \cos\beta + \sum_{j=1}^{j=i-1} p_j \cdot s - u_1 - N'_1} \qquad (A.10)$$

1) $N'_3 > 0$.

$$N'_3 = \left\{\frac{k}{k + tg\delta}\right\} \cdot \left\{ \frac{c'^{*}\{d \cdot tg\delta - s\} + \frac{c'}{F_2} \cdot d \cdot tg\delta}{sen\lambda + \cos\lambda \cdot tg\phi'^{*}} + \right.$$

$$\left. + \frac{G_2 \cdot \{sen\beta + tg\delta \cdot \cos\beta\} + u_2 - u_3 \cdot \{sen\lambda + tg\delta \cdot \cos\lambda\}}{sen\lambda + \cos\lambda \cdot tg\phi'^{*}} \right\} \qquad (A.11)$$

$$p_i = \frac{G_2 \cdot \{sen\beta - k \cdot \cos\beta\} - c'^{*} \cdot \{s + d \cdot k\} - \frac{c'}{F_2} \cdot d \cdot k}{s \cdot (k + tg\delta)} +$$

$$+ \frac{u_3 \cdot \{k \cdot \cos\lambda - sen\lambda\} + u_2}{s \cdot (k + tg\delta)} \qquad (A.12)$$

2) N'$_3$ < 0.

$$p_i = \frac{u_3 \cdot \left(s \cdot \cos\lambda + d \cdot sen\lambda\right) - \frac{c'}{F_2} \cdot d \cdot s - G_2 \cdot \left(d \cdot sen\beta + s \cdot \cos\beta\right) - u_2 \cdot d}{s^2 - d \cdot s \cdot tg\delta} \qquad \text{(A.13)}$$

$$F_3 = \frac{c' \cdot s}{u_2 + G_2 \cdot sen\beta - u_3 \cdot sen\lambda - p_i \cdot s \cdot tg\delta} \qquad \text{(A.14)}$$

CASO IV

Al resolver las ecuaciones se obtiene: N'$_1$ <0 y N'$_2$ > 0.

$$N_2' = \frac{G_1 \cdot \cos\beta + \sum_{j=1}^{j=i-1} p_j \cdot s - c'^{*} \cdot d - u_1}{tg\phi'^{*}} \qquad \text{(A.15)}$$

$$F_1 = \frac{c' \cdot \left\{(i-1) \cdot s + \frac{d}{tg\beta}\right\}}{G_1 \cdot sen\beta - u_2 - N_2' - \sum_{j=1}^{j=i-1} p_j \cdot s \cdot tg\delta} \qquad \text{(A.16)}$$

1) N'$_3$ > 0.

$$N_3' = \left\{\frac{k}{k + tg\delta}\right\} \cdot \left\{\frac{c'^{*}\left\{2 \cdot d \cdot tg\delta - s\right\} + N_2' \cdot \left\{1 + tg\delta \cdot tg\phi'^{*}\right\}}{sen\lambda + \cos\lambda \cdot tg\phi'^{*}} + \right.$$

$$\left. + \frac{G_2 \cdot \left\{sen\beta + tg\delta \cdot \cos\beta\right\} + u_2 - u_3 \cdot \left\{sen\lambda + tg\delta \cdot \cos\lambda\right\}}{sen\lambda + \cos\lambda \cdot tg\phi'^{*}}\right\} \qquad \text{(A.17)}$$

$$p_i = \frac{N'_2 \cdot \{1 - k \cdot tg\phi'^*\} + G_2 \cdot \{sen\beta - k \cdot \cos\beta\} - c'^* \cdot \{s + 2 \cdot k \cdot d\}}{s \cdot (k + tg\delta)} + \\ + \frac{u_3 \cdot \{k \cdot \cos\lambda - sen\lambda\} + u_2}{s \cdot (k + tg\delta)} \qquad \text{(A.18)}$$

2) N'$_3$ < 0.

$$p_i = \frac{u_3 \cdot (s \cdot \cos\lambda + d \cdot sen\lambda) - N'_2 \cdot (d + s \cdot tg\phi'^*) - c'^* \cdot d \cdot s}{s^2 - d \cdot s \cdot tg\delta} + \\ + \frac{-G_2 \cdot (d \cdot sen\beta + s \cdot \cos\beta) - u_2 \cdot d}{s^2 - d \cdot s \cdot tg\delta} \qquad \text{(A.19)}$$

$$F_3 = \frac{c' \cdot s}{u_2 + N'_2 + G_2 \cdot sen\beta - u_3 \cdot sen\lambda - p_i \cdot s \cdot tg\delta} \qquad \text{(A.20)}$$

CASO V

Al resolver las ecuaciones se obtiene: N'$_1$ < 0 y N'$_2$ < 0.

$$F_1 = \frac{c' \cdot \left\{(i-1) \cdot s + \frac{d}{tg\beta}\right\}}{G_1 \cdot sen\beta - u_2 - \sum_{j=1}^{j=i-1} p_j \cdot s \cdot tg\delta} \qquad \text{(A.21)}$$

$$F_2 = \frac{c' \cdot d}{G_1 \cdot \cos\beta + \sum_{j=1}^{j=i-1} p_j \cdot s - u_1} \qquad \text{(A.22)}$$

1) N'$_3$ > 0.

$$N'_3 = \left\{\frac{k}{k+tg\delta}\right\}\cdot\left\{\frac{c'^{*}\left\{d\cdot tg\delta - s\right\} + \frac{c'}{F_2}\cdot d\cdot tg\delta + G_2\cdot\left\{sen\beta + tg\delta\cdot\cos\beta\right\}}{sen\lambda + \cos\lambda\cdot tg\phi'^{*}} + \right.$$

$$\left. + \frac{u_2 - u_3\cdot\left\{sen\lambda + tg\delta\cdot\cos\lambda\right\}}{sen\lambda + \cos\lambda\cdot tg\phi'^{*}}\right\} \qquad \text{(A.23)}$$

$$p_i = \frac{G_2\cdot\left\{sen\beta - k\cdot\cos\beta\right\} - c'^{*}\cdot\left\{s + d\cdot k\right\} - \frac{c'}{F_2}\cdot d\cdot k}{s\cdot\left(k+tg\delta\right)} +$$

$$+ \frac{u_3\cdot\left\{k\cdot\cos\lambda - sen\lambda\right\} + u_2}{s\cdot\left(k+tg\delta\right)} \qquad \text{(A.24)}$$

2) N'$_3$ < 0.

$$p_i = \frac{u_3\cdot\left(s\cdot\cos\lambda + d\cdot sen\lambda\right) - \frac{c'}{F_2}\cdot d\cdot s - G_2\cdot\left(d\cdot sen\beta + s\cdot\cos\beta\right)}{s^2 - d\cdot s\cdot tg\delta} +$$

$$+ \frac{-u_2\cdot d}{s^2 - d\cdot s\cdot tg\delta} \qquad \text{(A.25)}$$

$$F_3 = \frac{c'\cdot s}{u_2 + G_2\cdot sen\beta - u_3\cdot sen\lambda - p_i\cdot s\cdot tg\delta} \qquad \text{(A.26)}$$

donde:

$$G_1 = \frac{\gamma \cdot d}{2} \cdot \left\{ 2 \cdot (i-1) \cdot s + \frac{d}{tg\beta} \right\} \tag{A.27}$$

$$G_2 = \frac{\gamma \cdot d}{2} \cdot s \tag{A.28}$$

$$k = \frac{sen\lambda + \cos\lambda \cdot tg\phi'^{*}}{\cos\lambda - sen\lambda \cdot tg\phi'^{*}} \tag{A.29}$$

$$\lambda = arctg\left(\frac{d}{s}\right) \tag{A.30}$$

$$u_1 = \begin{cases} \psi \cdot \dfrac{\gamma_w \cdot d^2 \cdot \cos\beta}{2 \cdot tg\beta} & para \quad i = 1 \\ \psi \cdot \dfrac{\left[2 \cdot (i-1) \cdot s + \dfrac{d}{tg\beta} - d \cdot tg(\beta - \lambda) \right] \cdot \gamma_w \cdot d \cdot \cos\alpha}{2 \cdot \cos(\beta - \lambda)} & para \quad i = 2,3,.... \end{cases} \tag{A.31}$$

$$u_2 = \begin{cases} \psi \cdot \dfrac{\gamma_w \cdot d^2 \cdot \cos\beta}{2} & para \quad i = 1 \\ \psi \cdot \dfrac{\gamma_w \cdot d^2 \cdot \cos\alpha}{2 \cdot \cos(\beta - \alpha)} & para \quad i = 2,3,.... \end{cases} \tag{A.32}$$

$$u_3 = \begin{cases} \psi \cdot \dfrac{\gamma_w \cdot d^2 \cdot \cos\beta}{2 \cdot sen\lambda} & para \quad i = 1 \\ \psi \cdot \dfrac{\gamma_w \cdot d^2 \cdot \cos\alpha}{2 \cdot sen\alpha \cdot \cos(\beta - \alpha)} & para \quad i = 2,3,.... \end{cases} \tag{A.33}$$

$$\psi = \begin{cases} 0 & no \quad presencia \quad de \quad agua \\ 1 & presencia \quad de \quad agua \end{cases} \tag{A.34}$$

$$tg\phi'^{*} = \frac{tg\phi'}{F_0} \tag{A.35}$$

$$c'^{*} = \frac{c'}{F_0} \tag{A.36}$$

A.3 GRÁFICOS PARA LA OBTENCIÓN DEL FACTOR DE REDUCCIÓN POR ALTURA FINITA DE TALUD

Se presentan las gráficas del factor de reducción por altura finita de talud y cohesión nula, para distintos ángulos de rozamiento interno, inclinación de talud y condiciones de filtración.

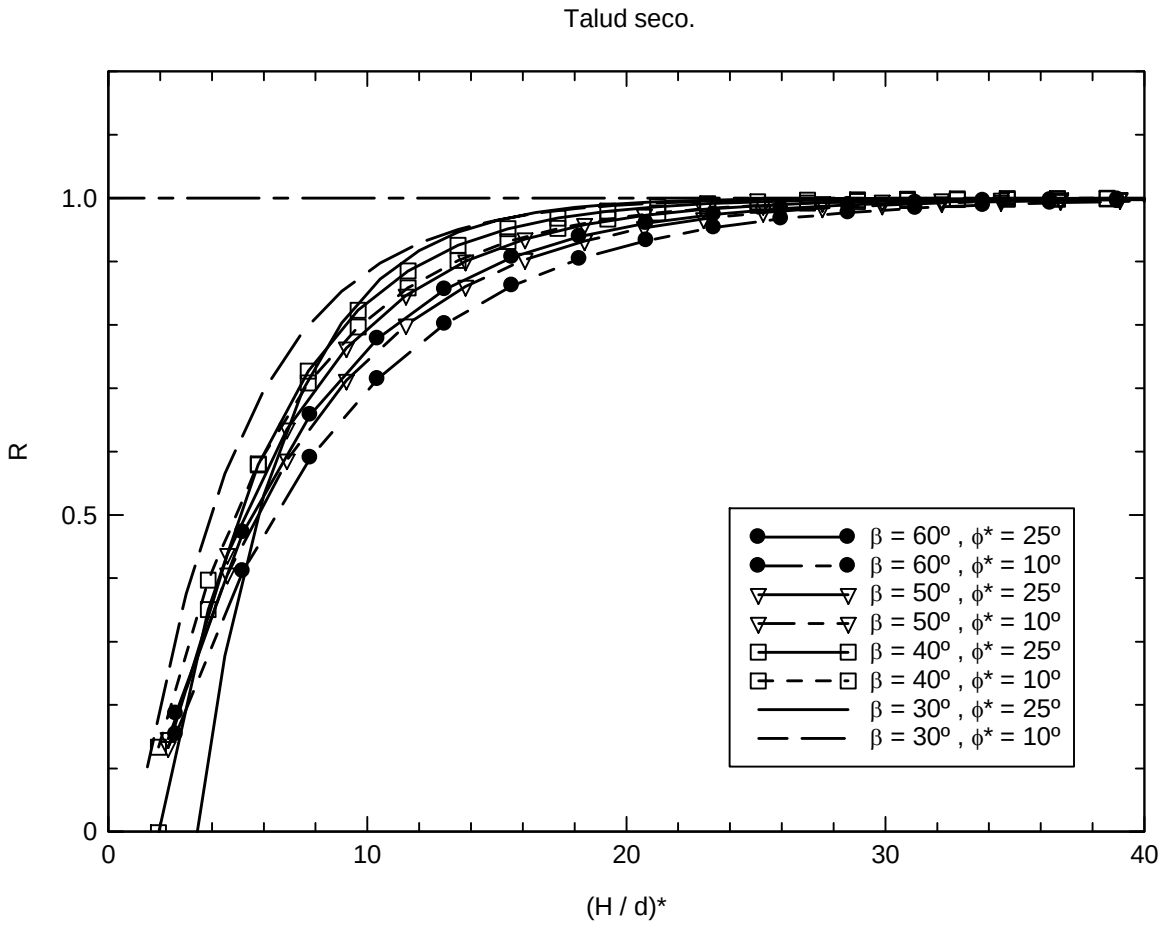


Figura A.10. Factor de reducción (c = 0). Talud seco

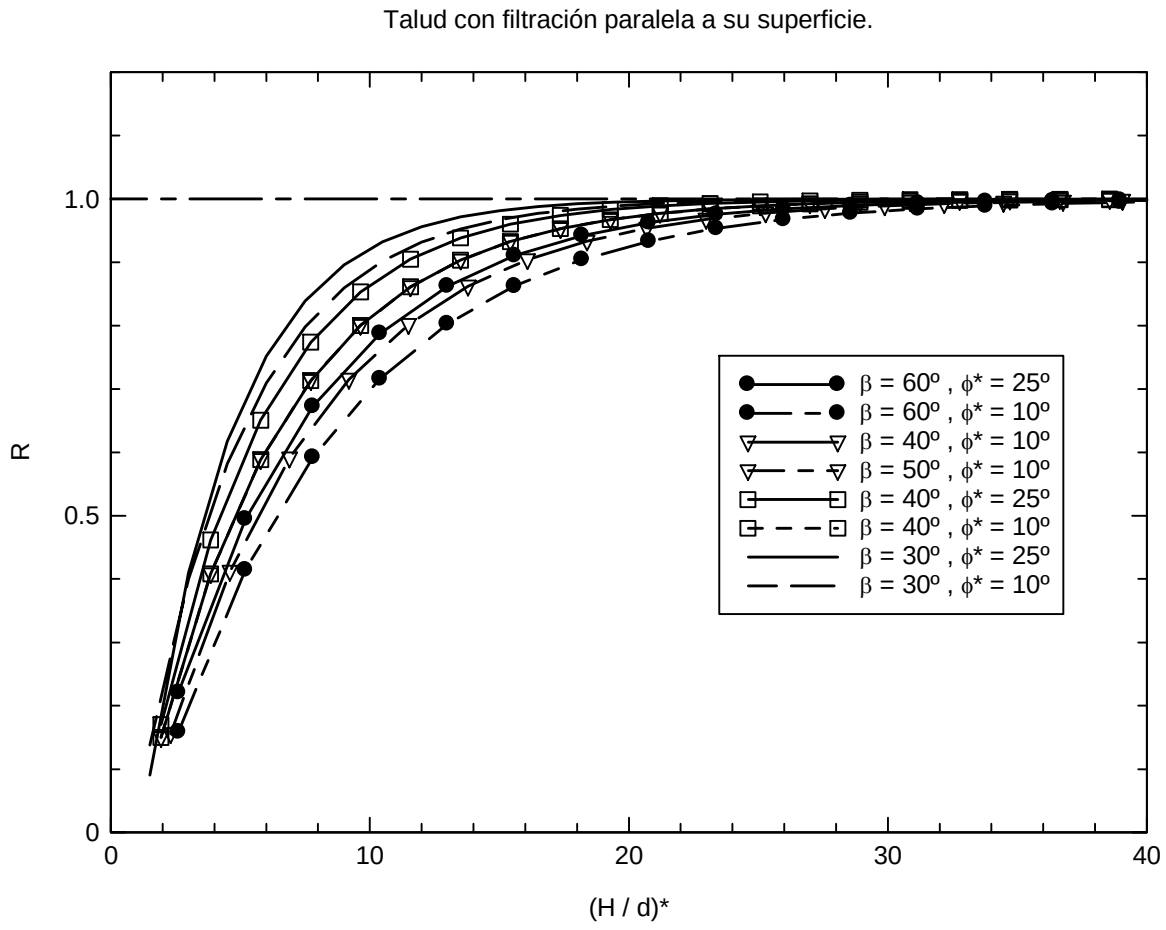


Figura A.11. Factor de reducción (c = 0). Talud con filtración paralela

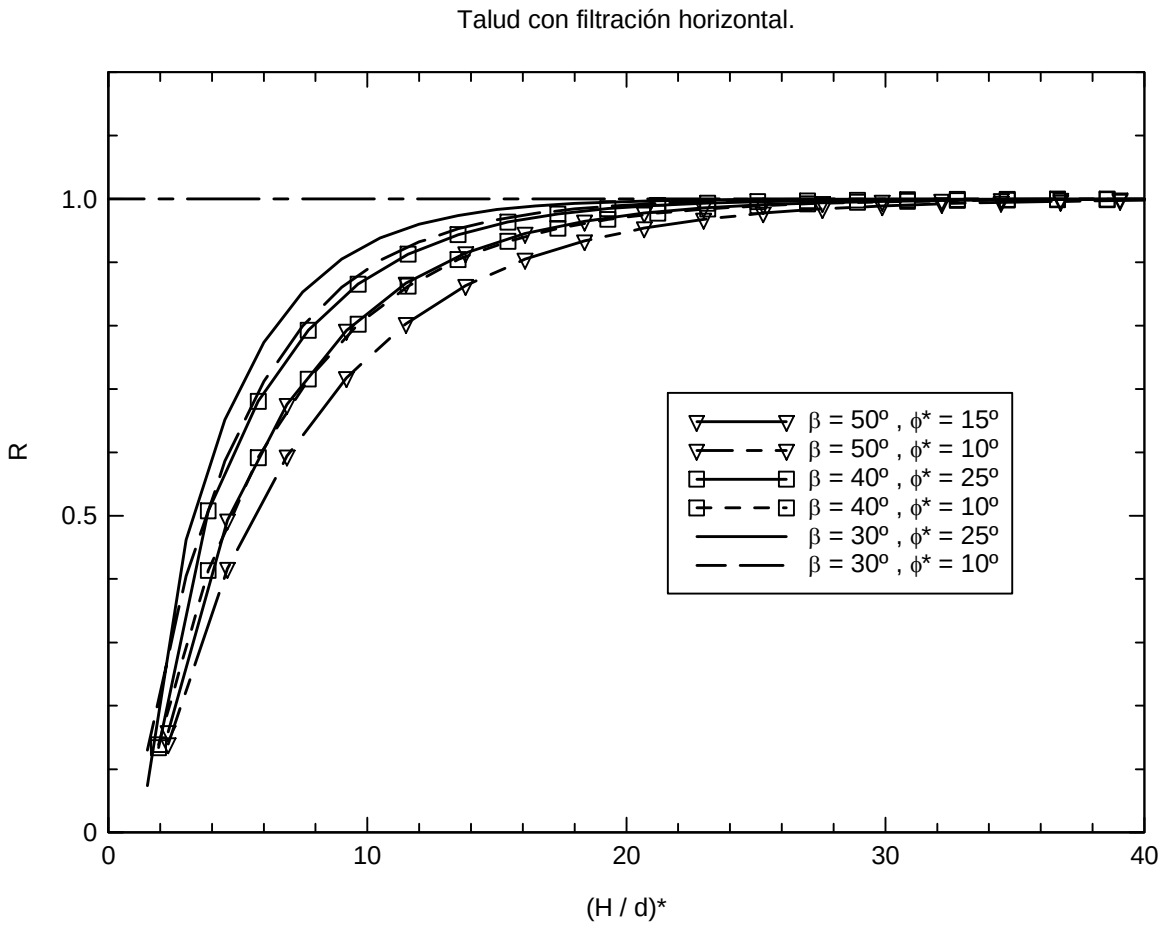


Figura A.12. Factor de reducción (c = 0). Talud con filtración horizontal

A.4 GRÁFICOS PARA LA OBTENCIÓN DE LA REDUCCIÓN DE ALTURA DE UN TALUD DEBIDO A LA COHESIÓN

Se presentan los gráficos correspondientes a la reducción en la altura del talud debido a la cohesión, para diferentes valores de ángulo de rozamiento interno, cohesión, inclinación del talud y condiciones de filtración.

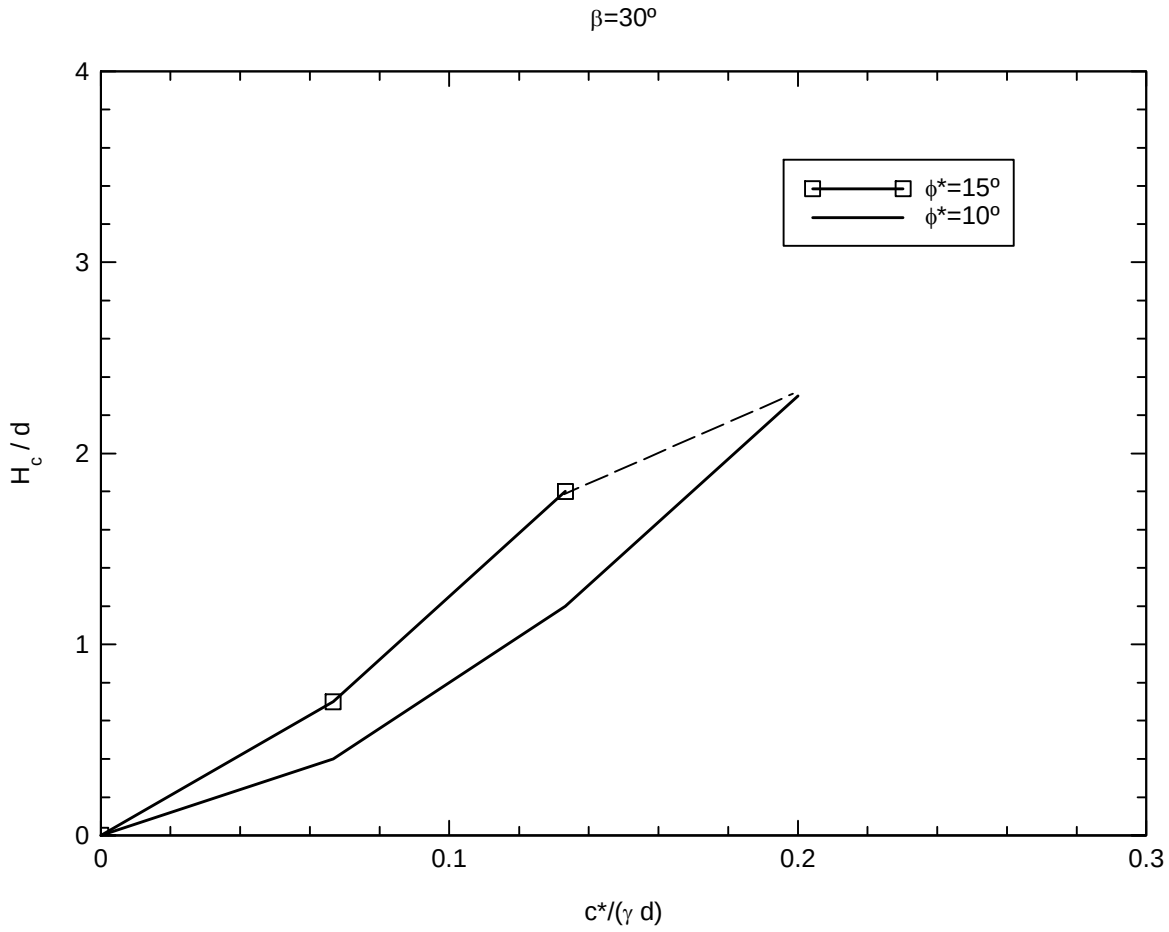


Figura A.13. Reducción de altura por cohesión (β = 30º). Talud seco

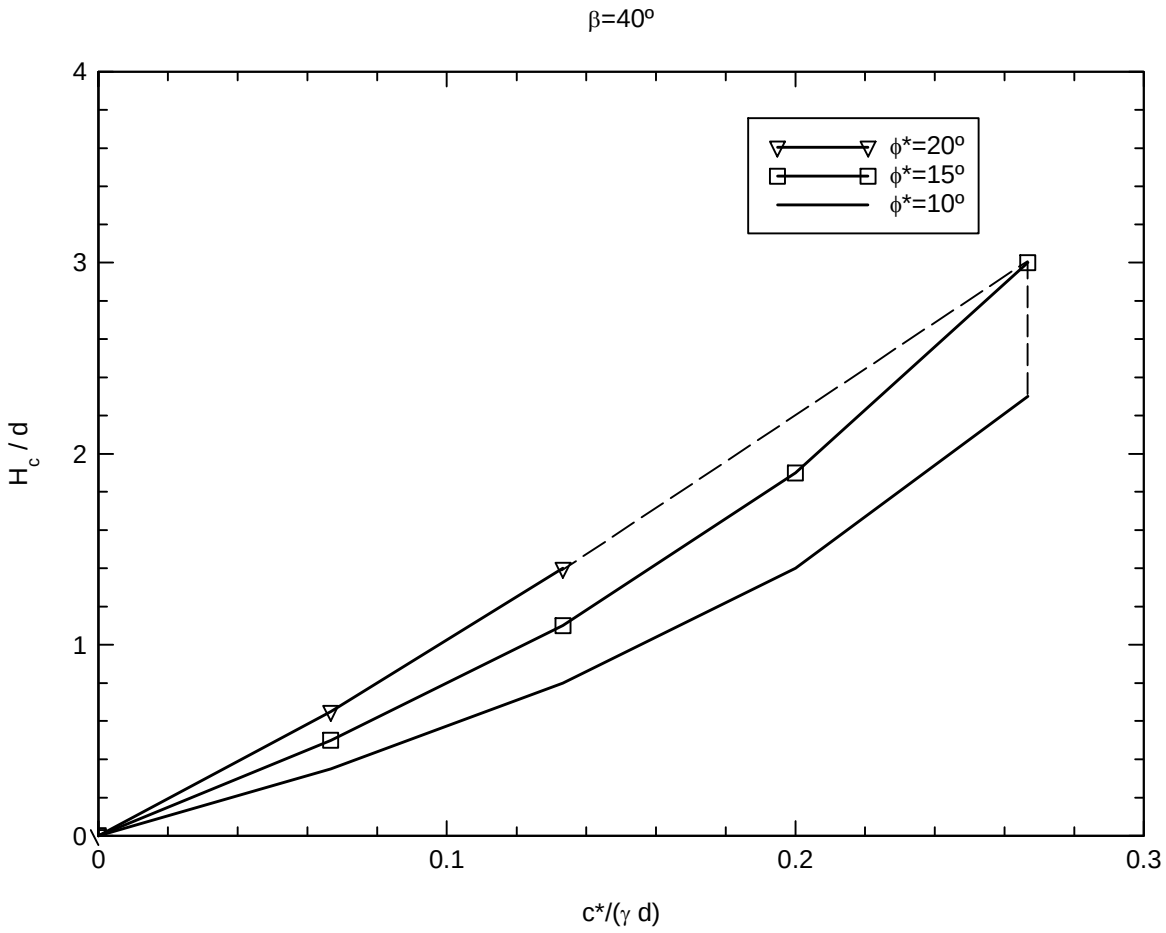


Figura A.14. Reducción de altura por cohesión (β = 40º). Talud seco

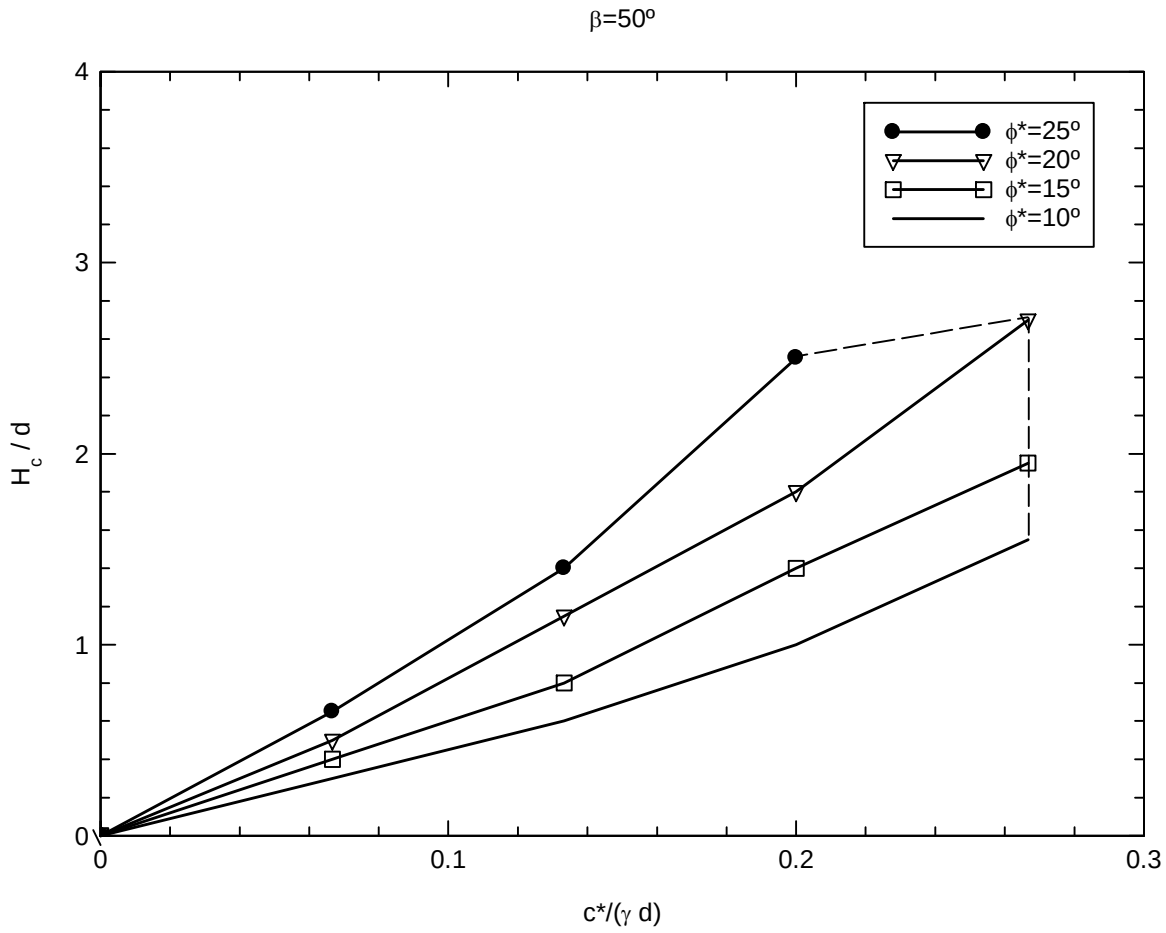


Figura A.15. Reducción de altura por cohesión (β = 50º). Talud seco

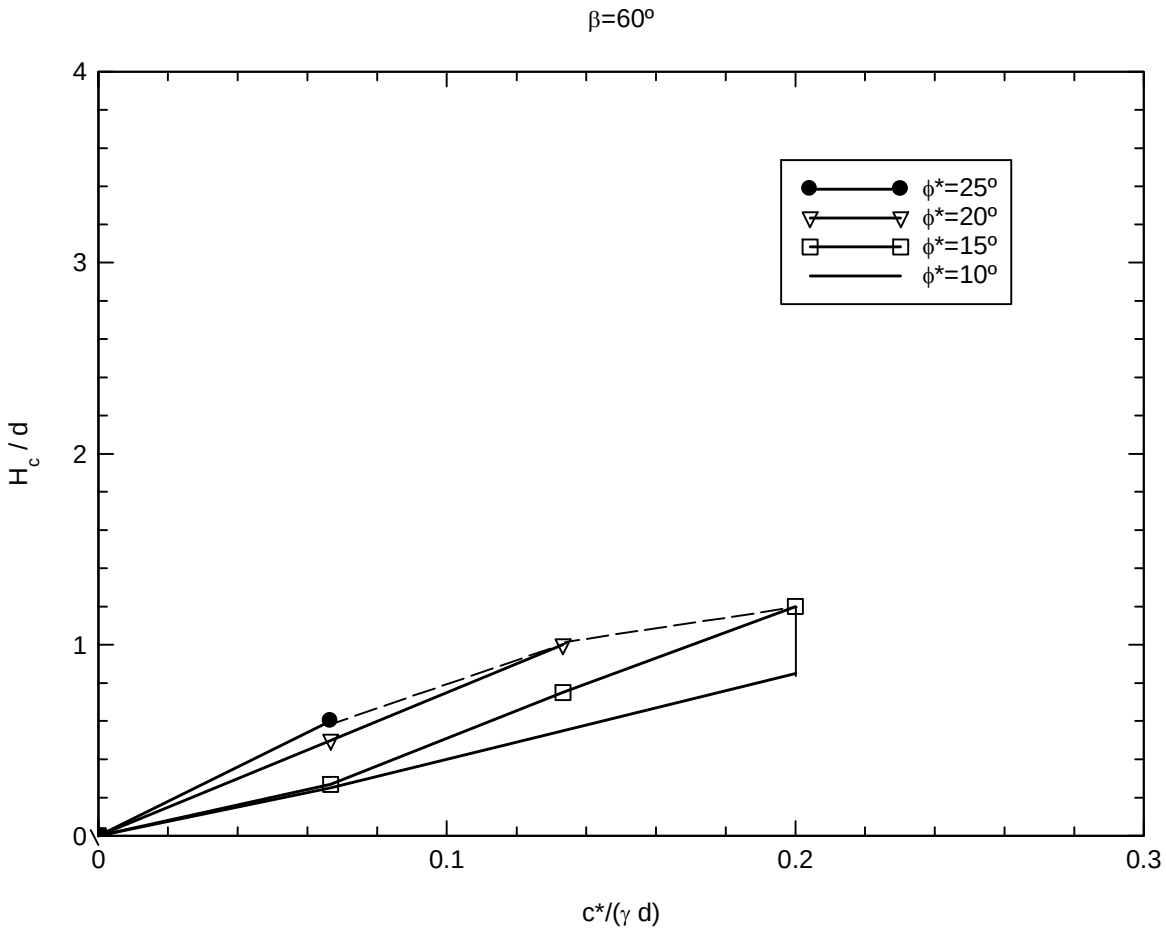


Figura A.16. Reducción de altura por cohesión (β = 60º). Talud seco

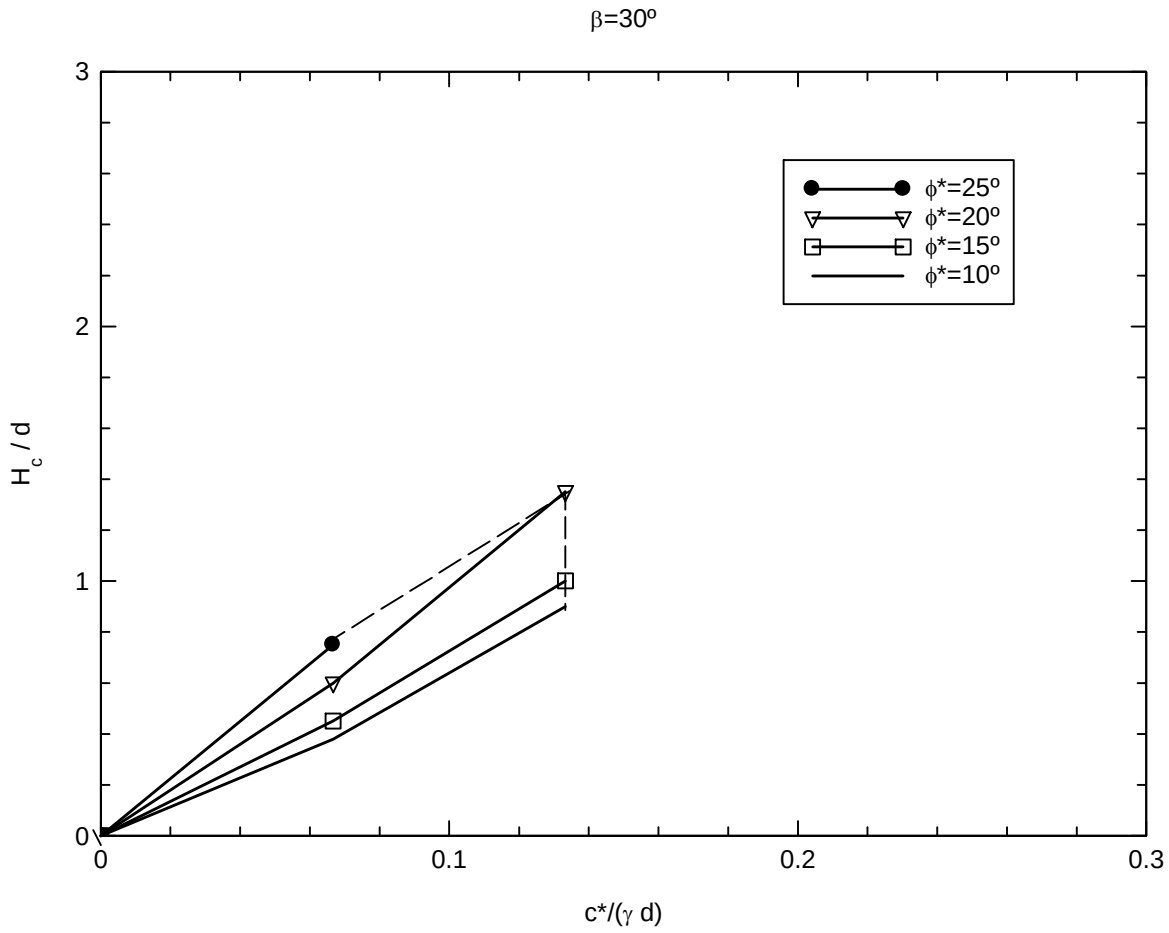


Figura A.17. Reducción de altura por cohesión (β = 30º). Talud con filtración paralela

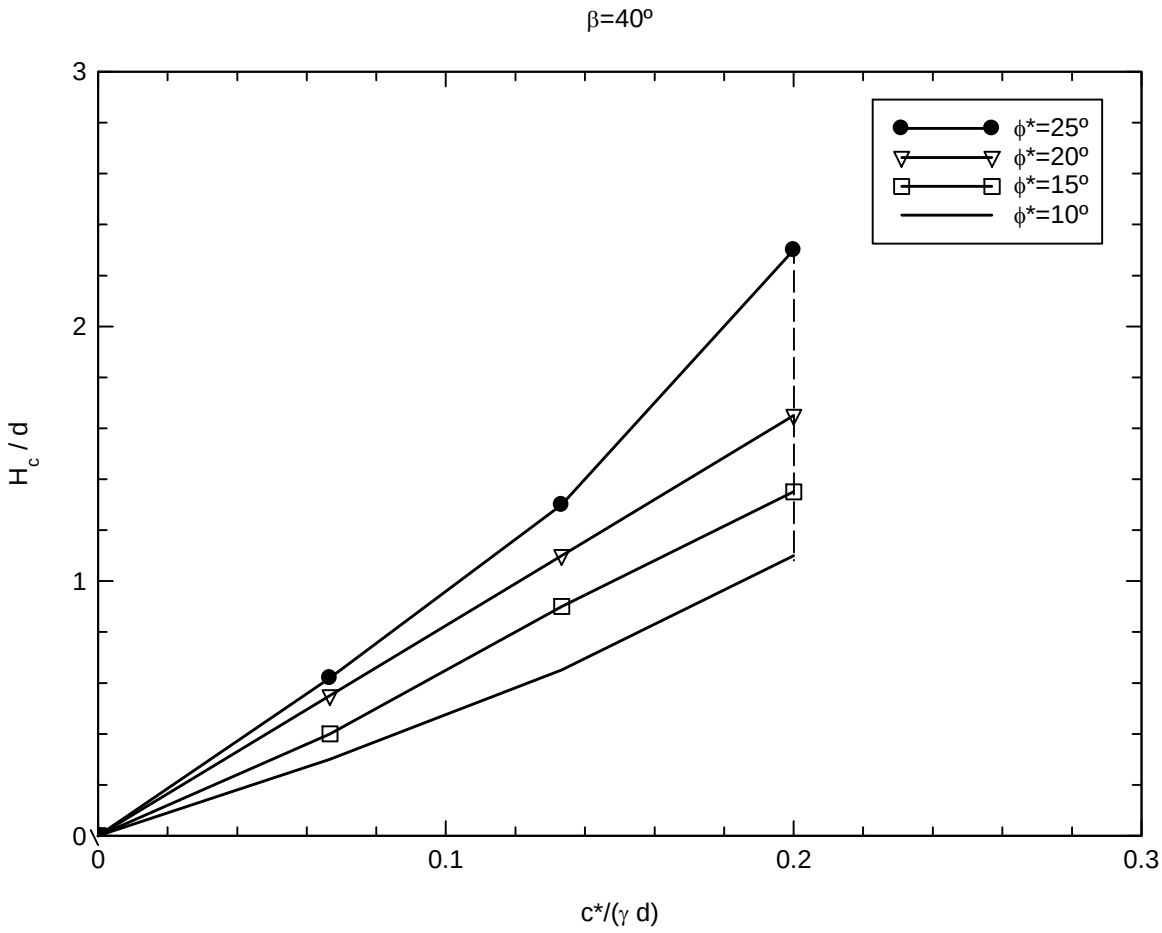


Figura A.18. Reducción de altura por cohesión (β = 40º). Talud con filtración paralela

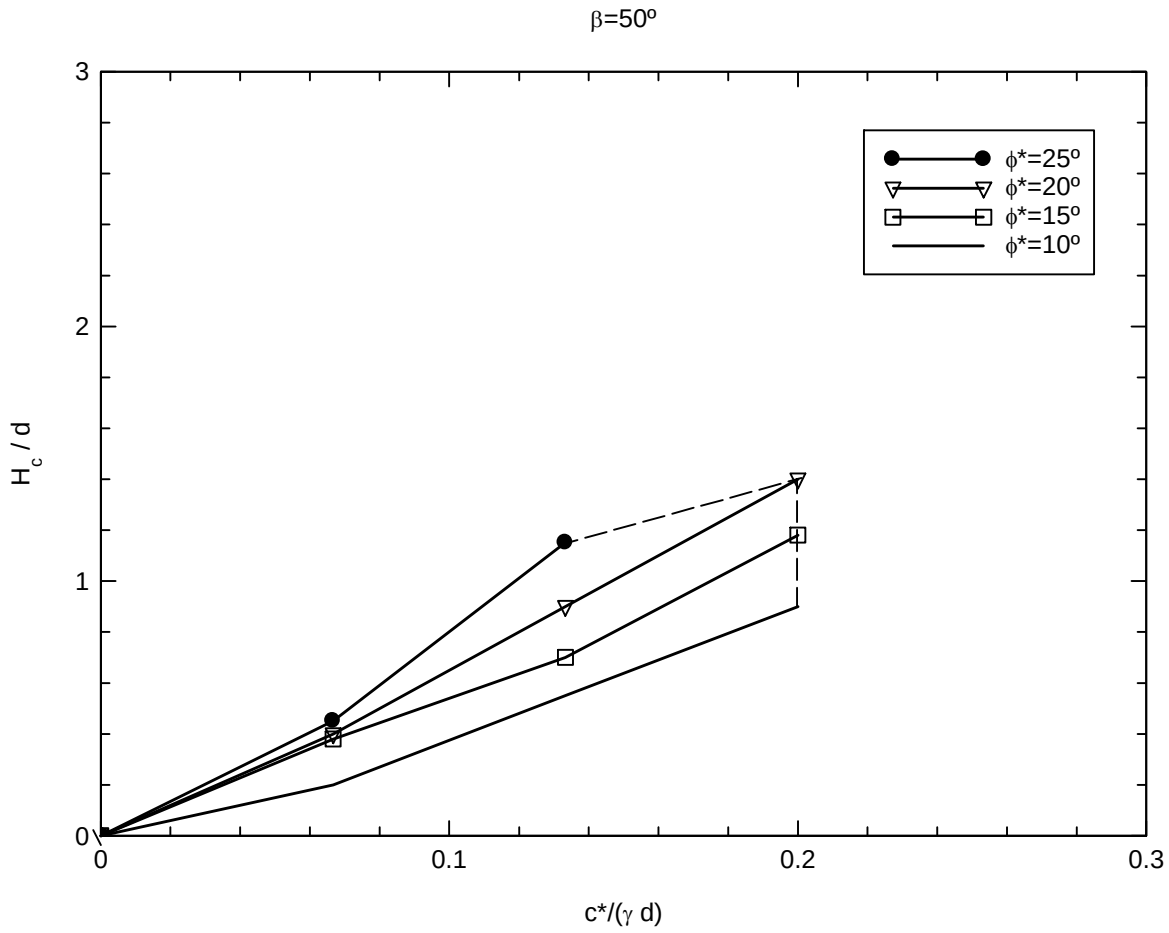


Figura A.19. Reducción de altura por cohesión (β = 50º). Talud con filtración paralela

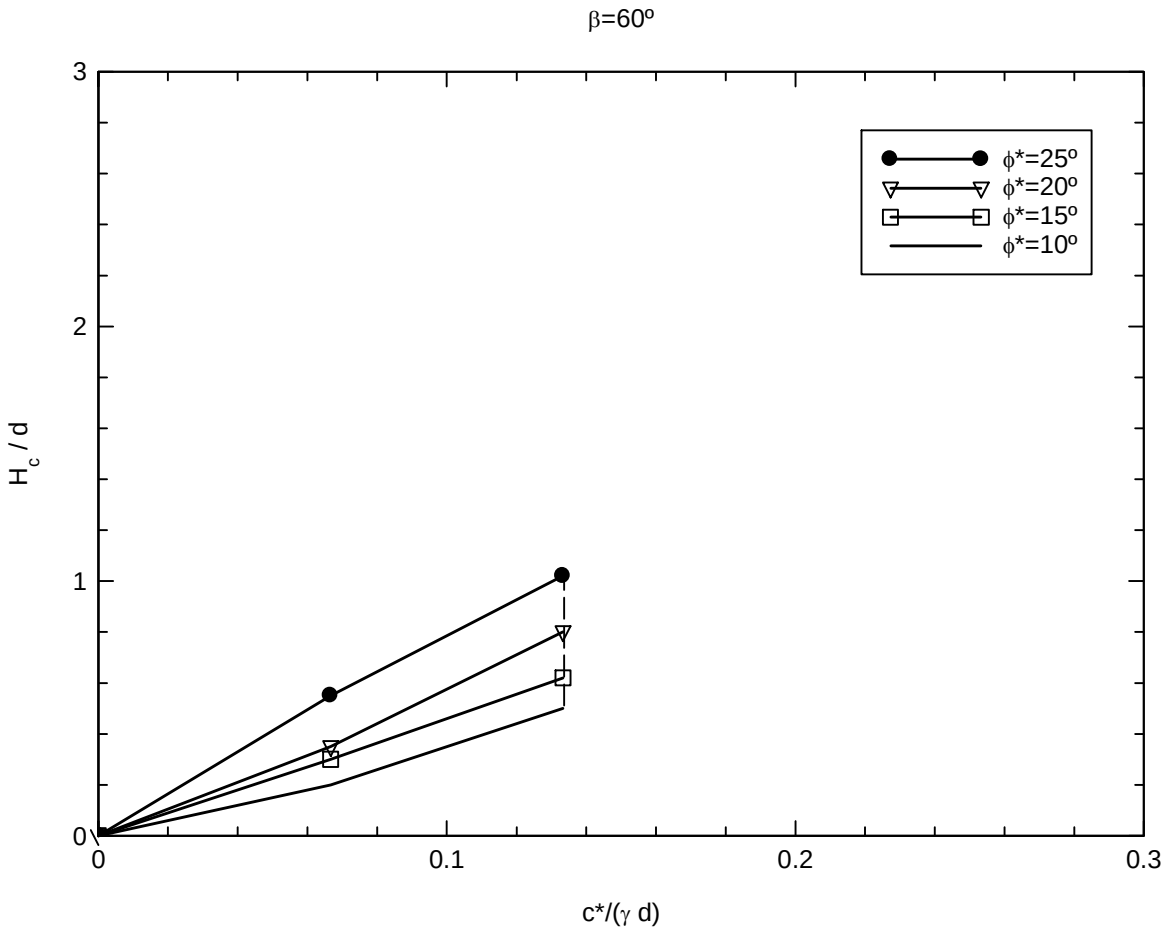


Figura A.20. Reducción de altura por cohesión (β = 60º). Talud con filtración paralela

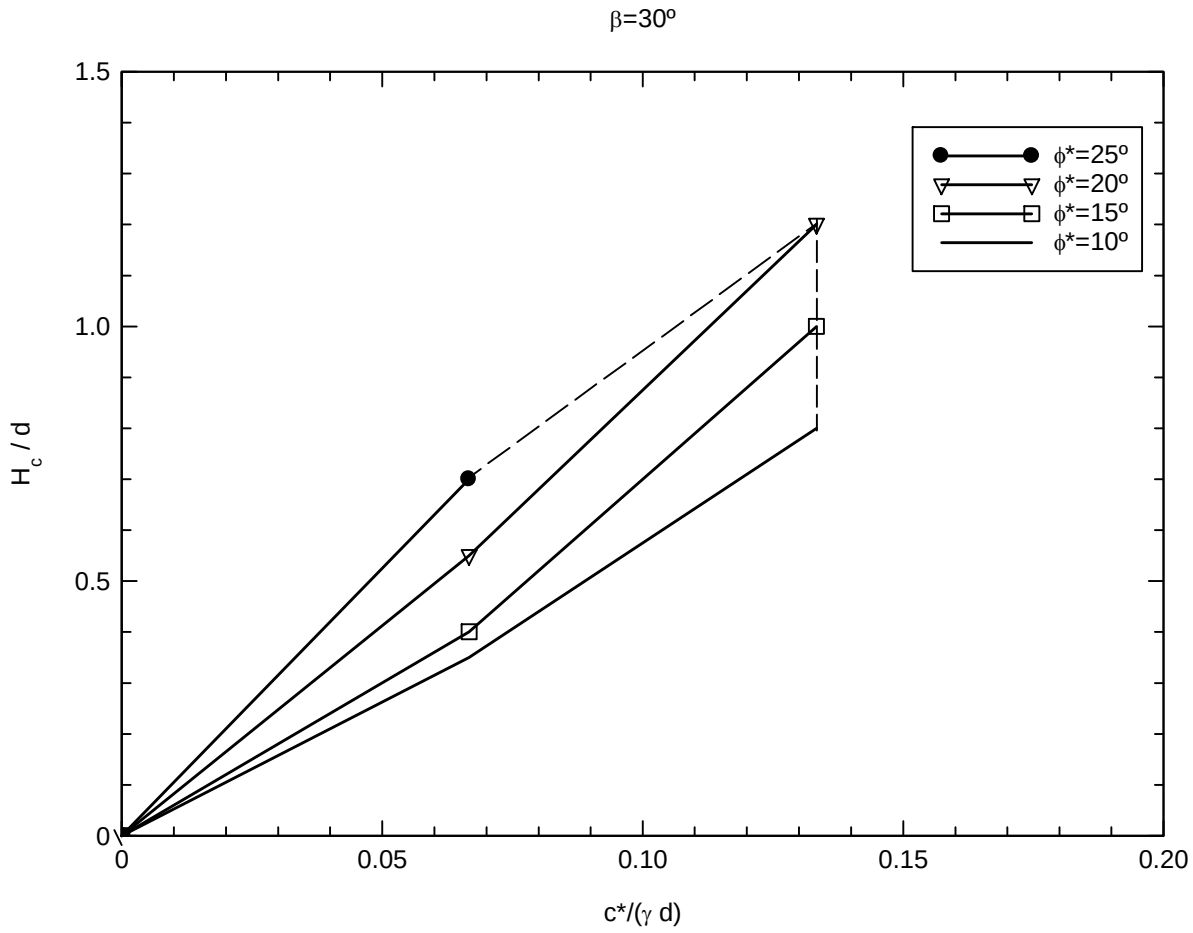


Figura A.21. Reducción de altura por cohesión (β = 30º). Talud con filtración horizontal

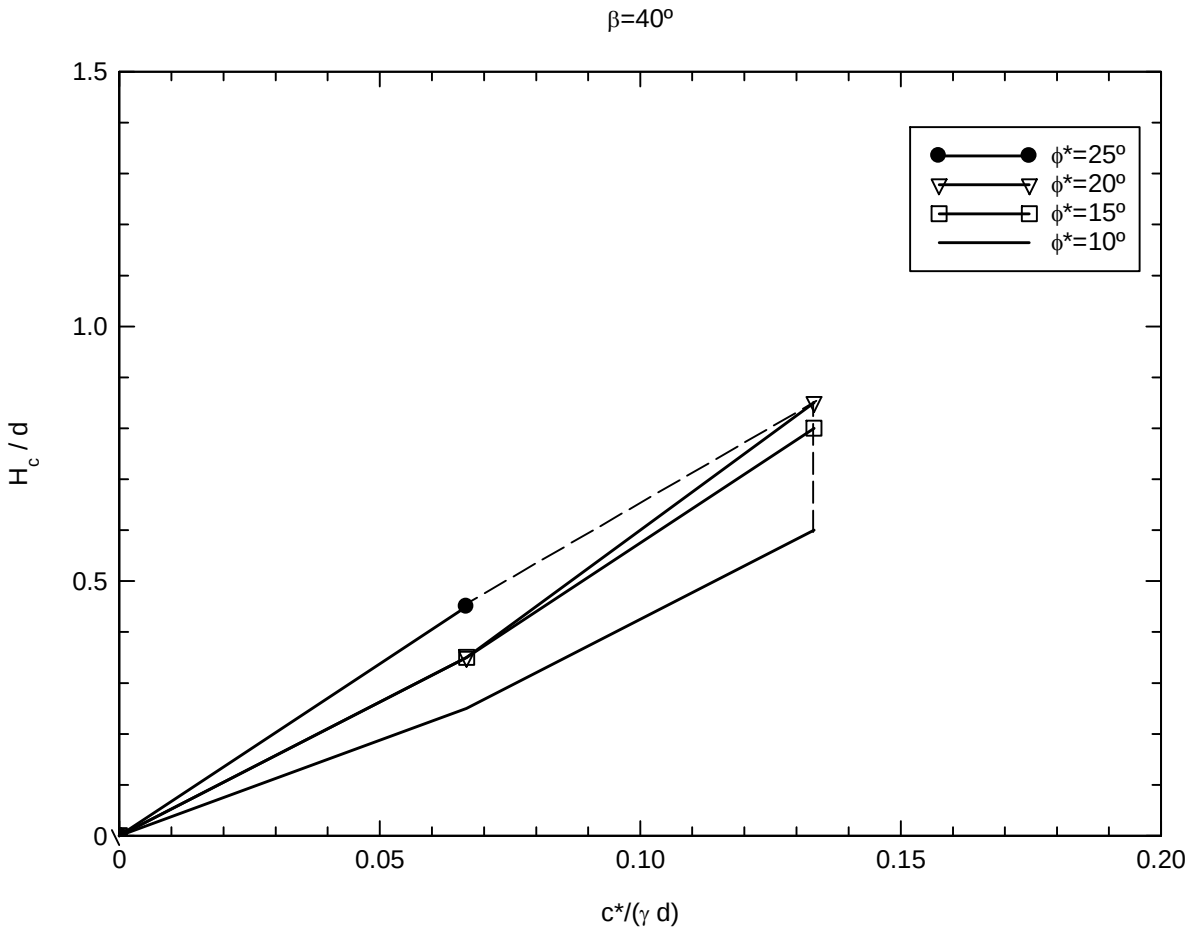


Figura A.22. Reducción de altura por cohesión (β = 40º). Talud con filtración horizontal

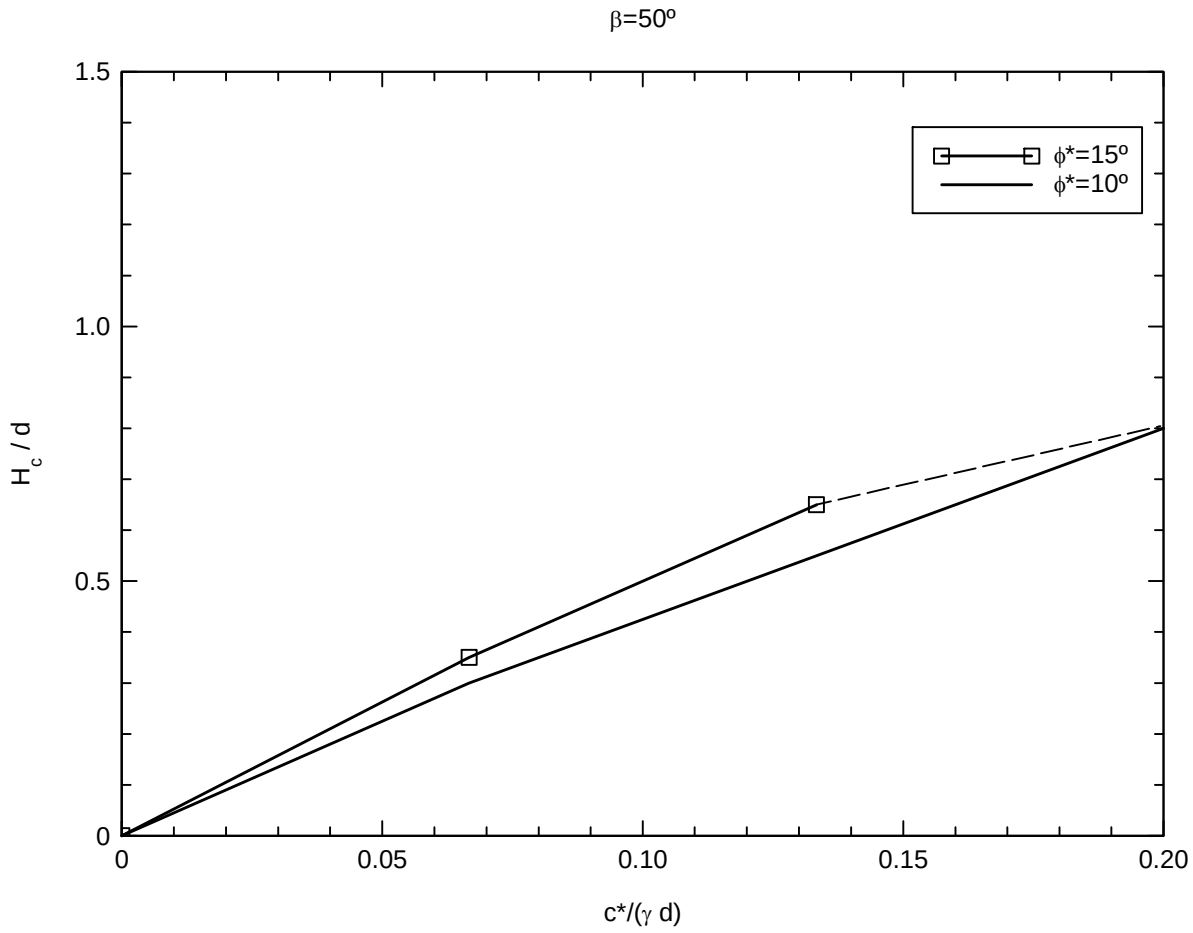


Figura A.23. Reducción de altura por cohesión (β = 50º). Talud con filtración horizontal

APÉNDICE B. RESULTADOS DE APLICACIÓN DE LOS MODELOS PRESIÓN-CURVATURA A LOS ENSAYOS DE CARGA DISTRIBUIDA SOBRE LA MALLA

B.1. DEFORMADA DE LA MALLA TECCO G-65 SOMETIDA A CARGA UNIFORME EN SU SUPERFICIE

En el Capítulo 3 se presentan unos modelos de presión-curvatura para la malla de refuerzo, con los que se trata de obtener la deformada de una malla y el valor de la fuerza axil que está soportando, en función de sus características resistentes y la geometría, para el caso de una presión actuando sobre su superficie.

Dichos modelos se presentan para el caso de deformada cilíndrica de la malla, y para el de deformada esférica, correspondiendo cada uno de dichos casos a una diferente puesta en obra del sistema de refuerzo. Dada la similitud del caso de deformada cilíndrica con el caso del ensayo de laboratorio de carga distribuida sobre la superficie de la malla TECCO G-65, se compara en dicho Capítulo los resultados de los ensayos de laboratorio con los que se obtendrían, para las condiciones de dichos ensayos, con la aplicación del correspondiente modelo.

En este Apéndice se presentan los gráficos correspondientes a los resultados que se obtienen con respecto a la deformada que adquiere la malla debido a la acción de la carga distribuida en su superficie, obtenidos aplicando el modelo teórico citado anteriormente. Cada uno de los resultados obtenidos se acompaña del correspondiente al ensayo de laboratorio.

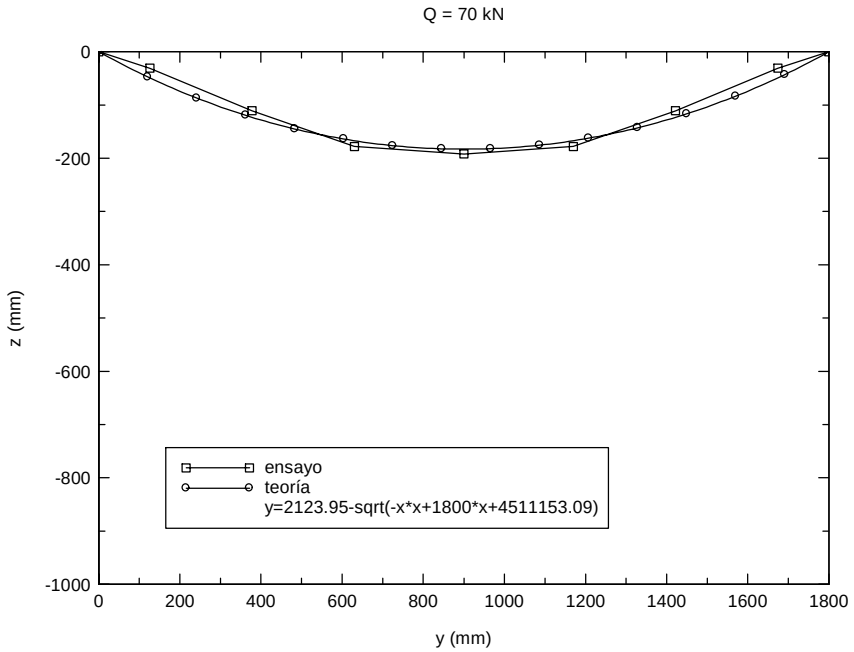


Figura B.1. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados teóricos (Q = 70 kN)

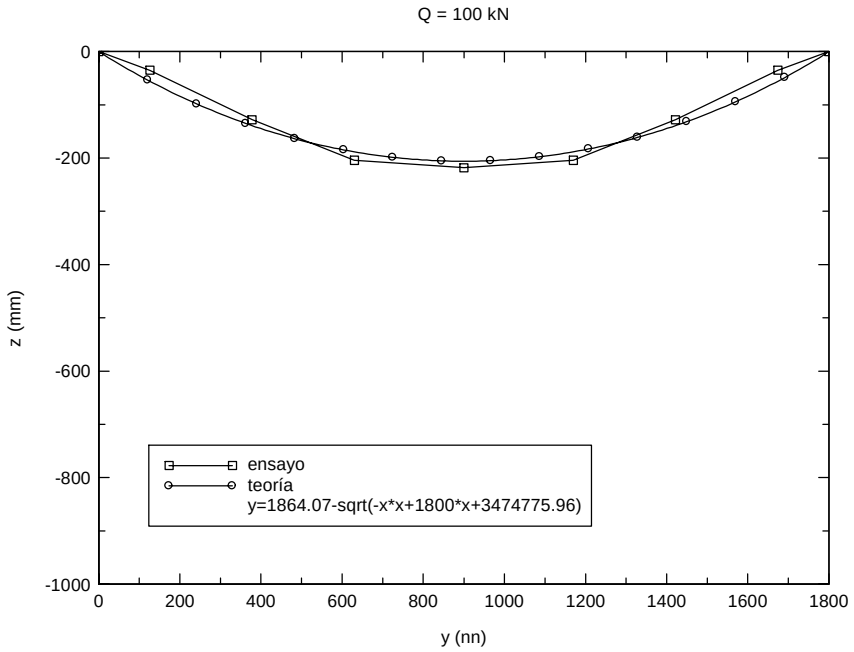


Figura B.2. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados teóricos (Q = 100 kN)

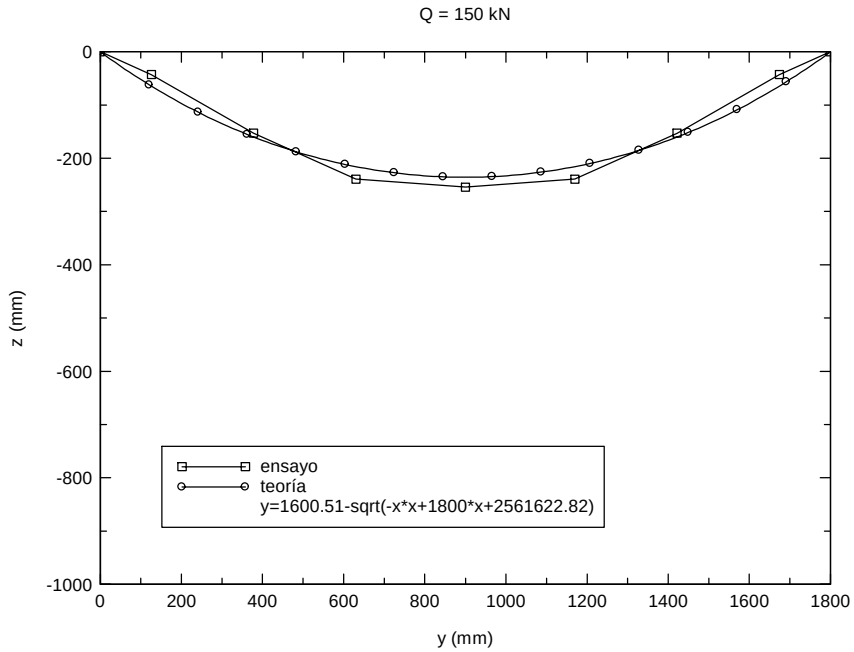


Figura B.3. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados teóricos (Q = 150 kN)

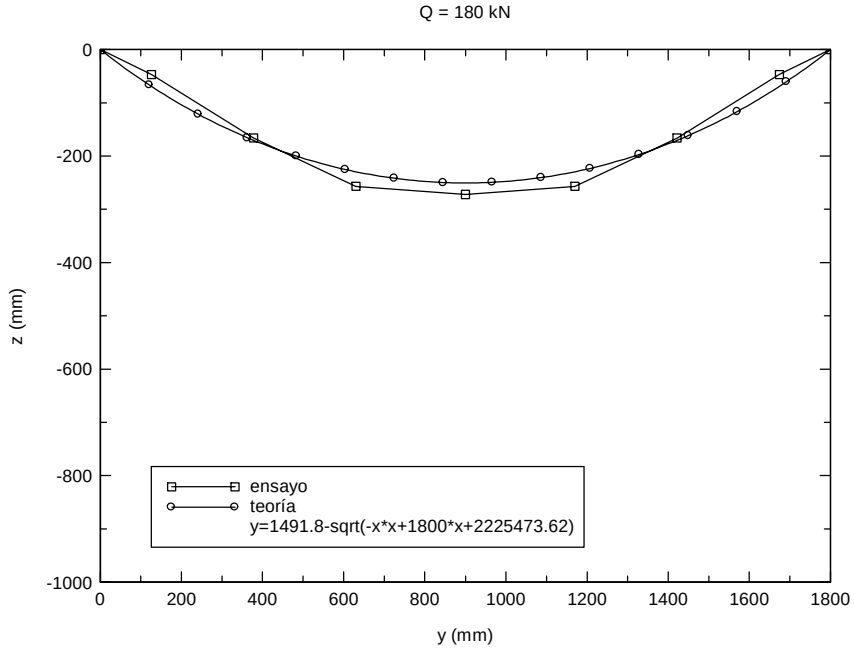


Figura B.4. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados teóricos (Q = 180 kN)

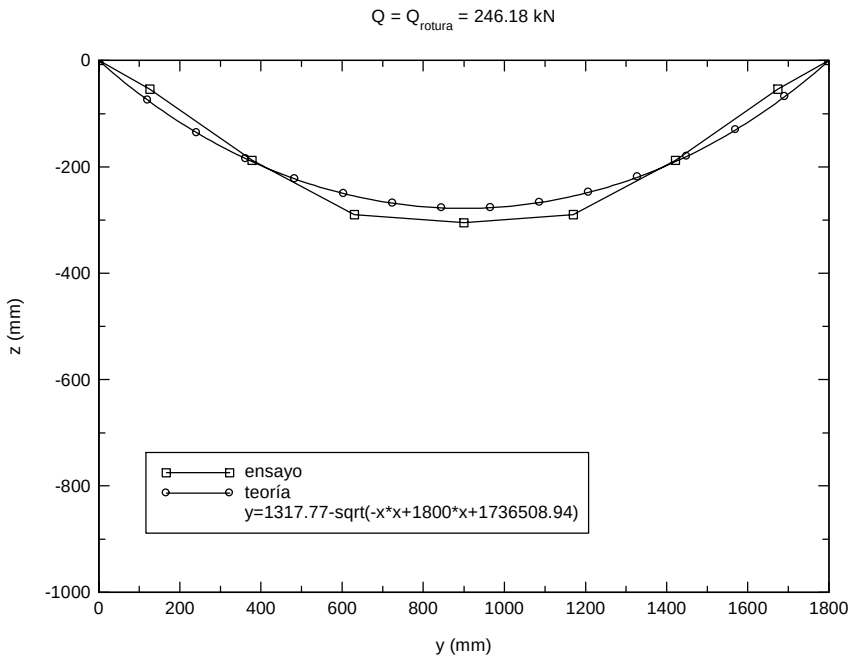


Figura B.5. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados teóricos ($Q = Q_{rot} = 246.18$ kN)

APÉNDICE C. RESULTADOS DEL ANÁLISIS DE LAS CELDAS ELEMENTALES

C.1. INTRODUCCIÓN

En el Capítulo 4 se analiza, mediante el empleo de los programas CRISP 90 y PLAXIS, una serie de celdas elementales.

En este apéndice se presentan los gráficos correspondientes a los resultados obtenidos sobre la deformada que adquiere la malla y la fuerza axil a la que se ve sometida. Dichos resultados se presentan para cada una de las celdas elementales analizadas, y para cada unos de los programas empleados.

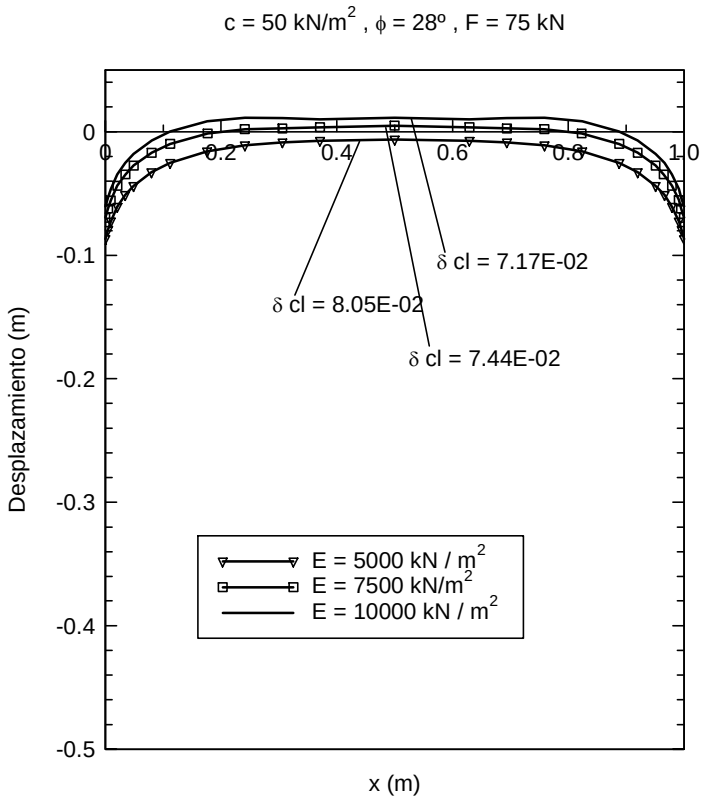
Para analizar los resultados que se presentan, es preciso tener en cuenta el criterio de signos considerado en cada programa con respecto a las tensiones, tanto en el suelo como en los elementos que representan la malla de refuerzo.

En el programa CRISP90 se consideran positivas las compresiones en el suelo, y negativas las tracciones en los elementos ‘bar’ empleados para representar la malla.

En el programa PLAXIS se consideran negativas las compresiones en el suelo, y positivas las tracciones en los elementos ‘geotextile’ empleados para representar la malla.

C.2. EFECTO DEL PRETENSADO DE LOS ANCLAJES. RESULTADOS DEL PROGRAMA CRISP90

En las figuras siguientes se presentan los resultados obtenidos, con el programa CRISP90, del análisis de la celda elemental correspondiente al estudio del efecto del pretensado de los anclajes. Los resultados se refieren a la influencia del módulo de elasticidad, cohesión, y espesor de suelo considerado, en la deformada de la malla y la fuerza axil a la que se ve sometida.



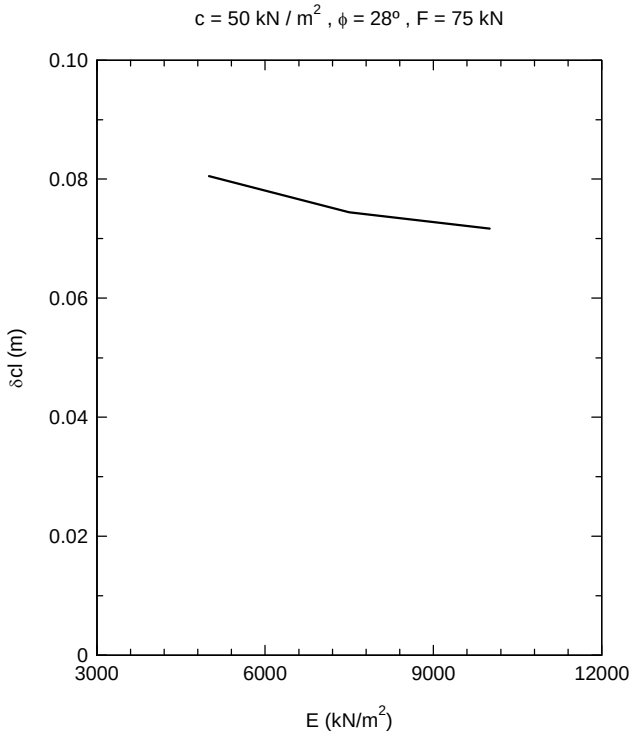
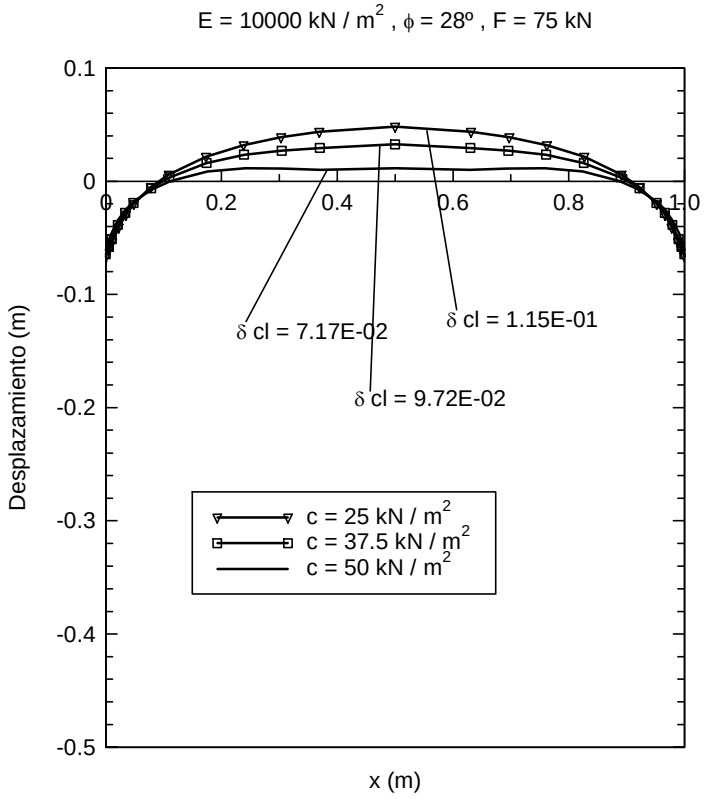


Figura C.1. Efecto de pretensado de anclajes. Deformada de la malla en función del módulo de elasticidad del suelo (CRISP90)



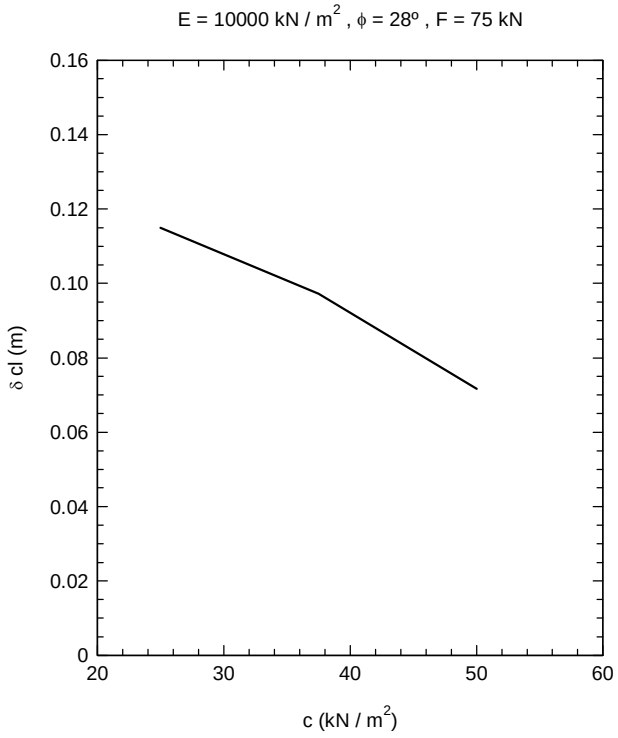


Figura C.2. Efecto de pretensado de anclajes. Deformada de la malla en función de la cohesión del suelo (CRISP90)

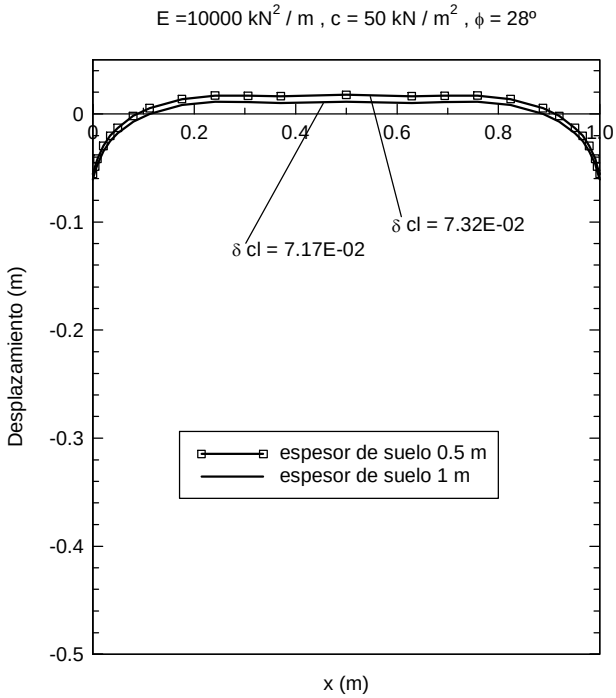


Figura C.3. Efecto de pretensado de anclajes. Deformada de la malla en función del espesor de suelo (CRISP90)

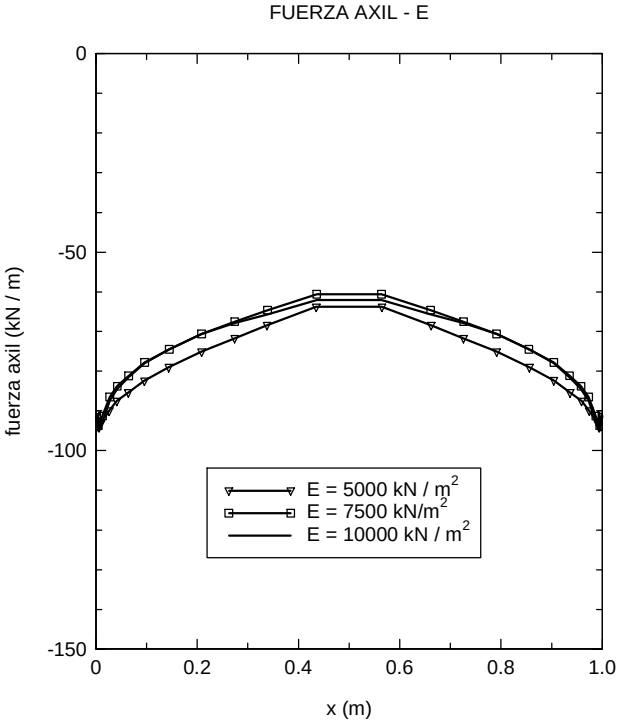


Figura C.4. Efecto de pretensado de anclajes. Fuerza axil en la malla en función del módulo de elasticidad del suelo (CRISP90)

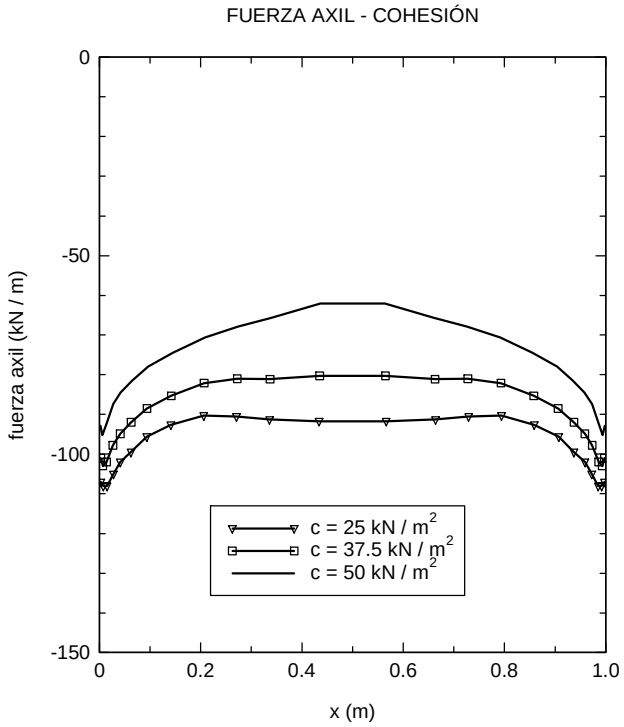


Figura C.5. Efecto de pretensado de anclajes. Fuerza axil en la malla en función de la cohesión del suelo (CRISP90)

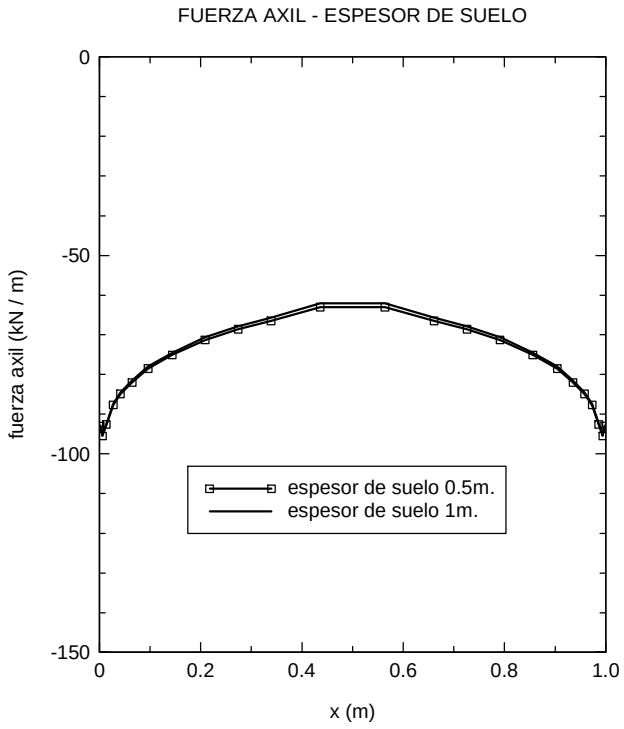


Figura C.6. Efecto de pretensado de anclajes. Fuerza axil en la malla en función del espesor de suelo (CRISP90)

C.3. EFECTO DEL PRETENSADO DE LOS ANCLAJES. RESULTADOS DEL PROGRAMA PLAXIS

En las figuras siguientes se presentan los resultados obtenidos, con el programa PLAXIS, del análisis de la celda elemental correspondiente al estudio del efecto del pretensado de los anclajes. Los resultados se refieren a la influencia del módulo de elasticidad, el ángulo de rozamiento interno, y la cohesión, en la deformada de la malla y la fuerza axil a la que se ve sometida. También se presentan resultados correspondientes a las tensiones normales y tangenciales del suelo situado en distintas seccione bajo la malla.

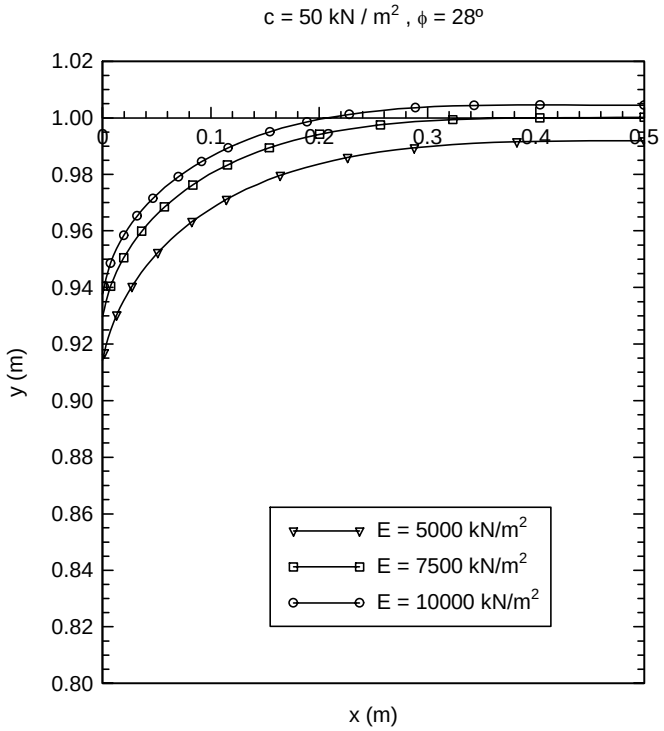


Figura C.7. Efecto de pretensado de anclajes. Deformada de la malla en función del módulo de elasticidad del suelo (PLAXIS)

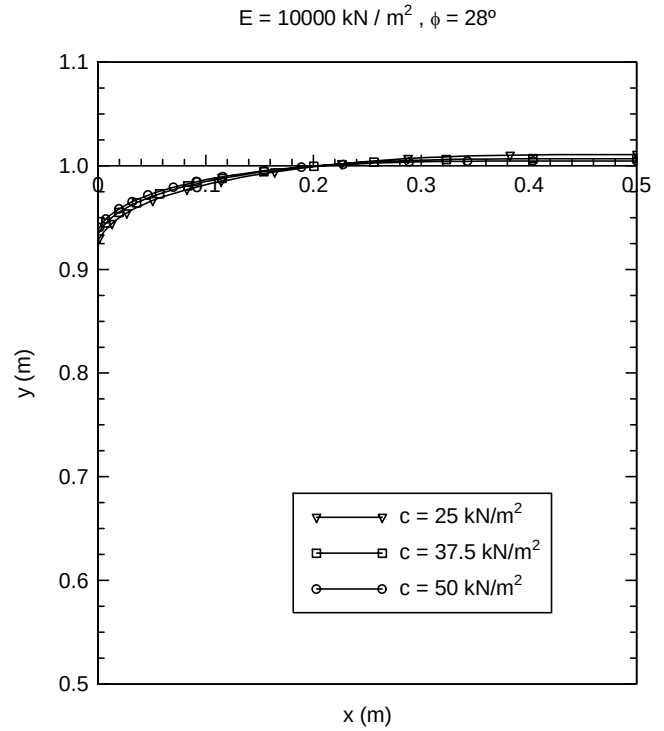


Figura C.8. Efecto de pretensado de anclajes. Deformada de la malla en función de la cohesión del suelo (PLAXIS)

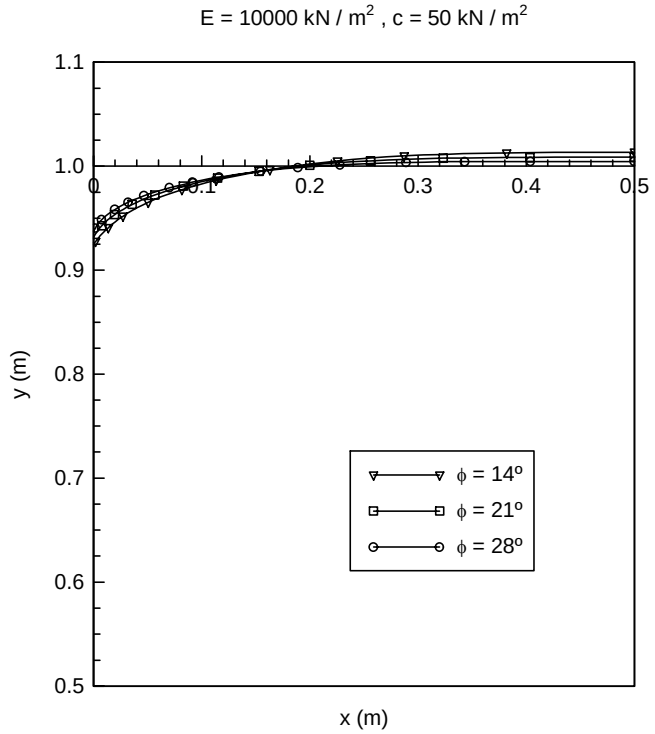


Figura C.9. Efecto de pretensado de anclajes. Deformada de la malla en función del ángulo de rozamiento del suelo (PLAXIS)

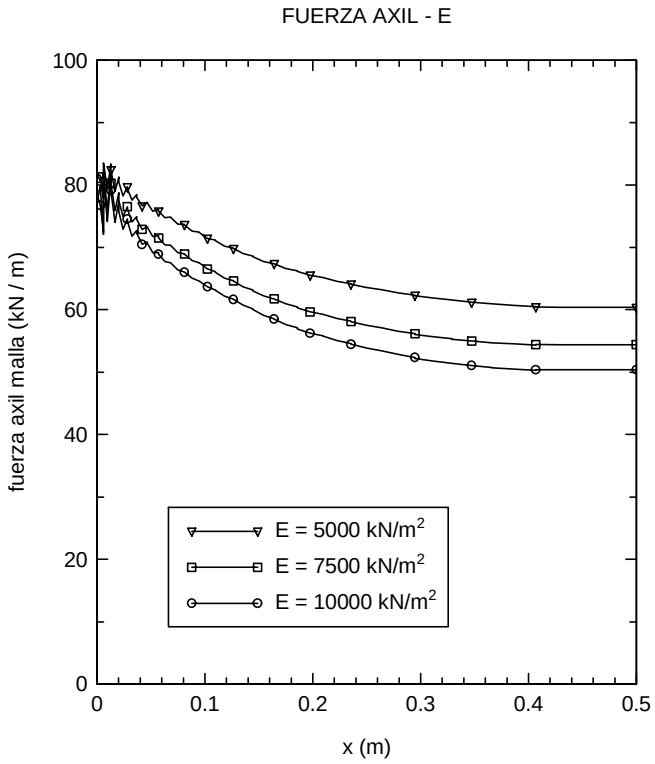


Figura C.10. Efecto de pretensado de anclajes. Fuerza axil en la malla en función del módulo de elasticidad del suelo (PLAXIS)

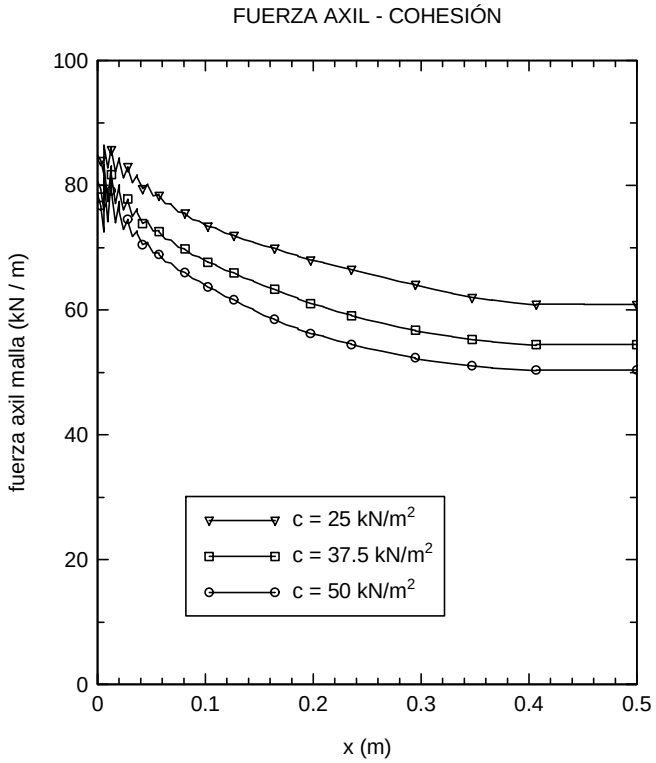


Figura C.11. Efecto de pretensado de anclajes. Fuerza axil en la malla en función de la cohesión del suelo (PLAXIS)

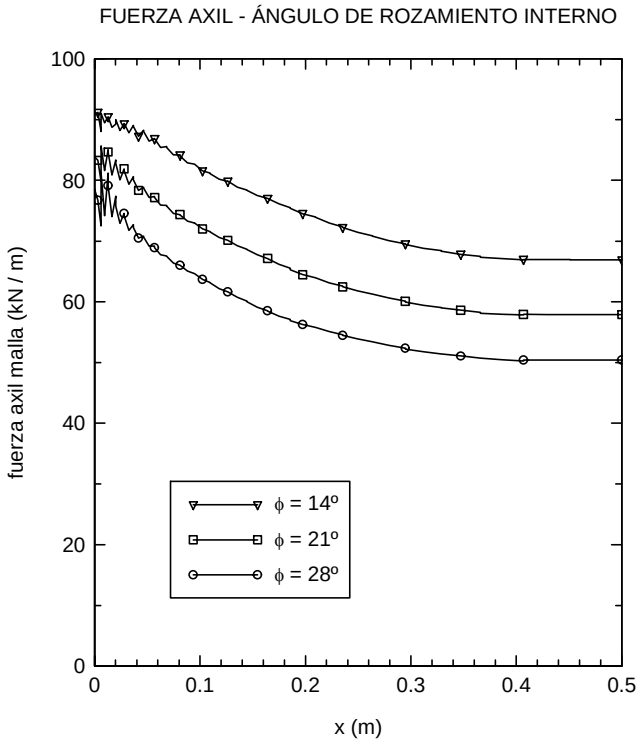


Figura C.12. Efecto de pretensado de anclajes. Fuerza axil en la malla en función del ángulo de rozamiento del suelo (PLAXIS)

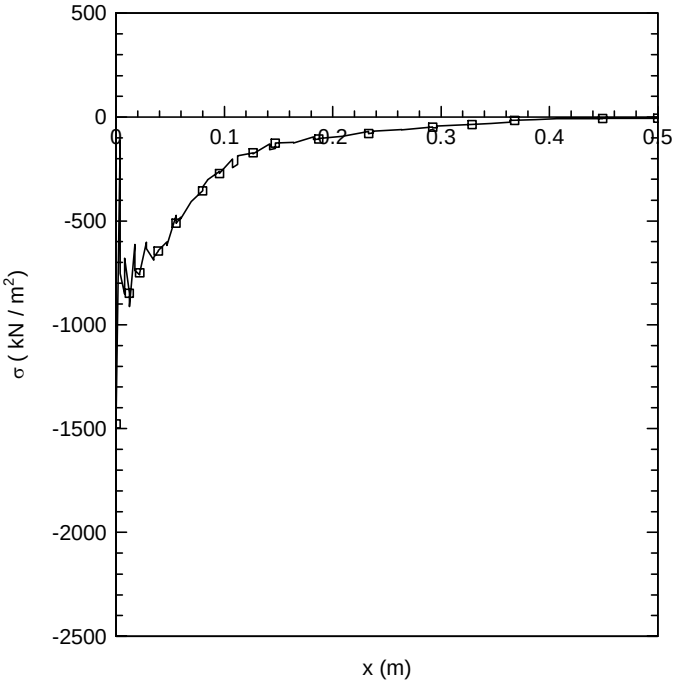


Figura C.13. Efecto de pretensado de anclajes. Tensiones normales en una sección de suelo bajo la malla (PLAXIS)

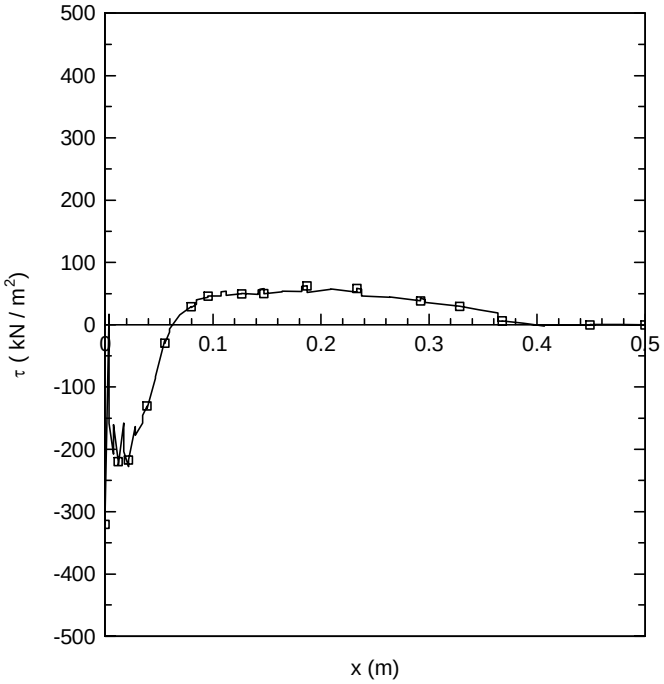


Figura C.14. Efecto de pretensado de anclajes. Tensiones tangenciales en una sección de suelo bajo la malla (PLAXIS)

C.4. EFECTO DE LA INESTABILIDAD SUPERFICIAL. RESULTADOS DEL PROGRAMA PLAXIS

En las figuras siguientes se presentan los resultados obtenidos, con el programa PLAXIS, del análisis de la celda elemental correspondiente al estudio de la pérdida de estabilidad superficial. Los resultados se refieren a la influencia de la cohesión, el ángulo de rozamiento interno, y el ángulo de dilatancia; en la deformada de la malla, y en la fuerza axil a la que se ve sometida.

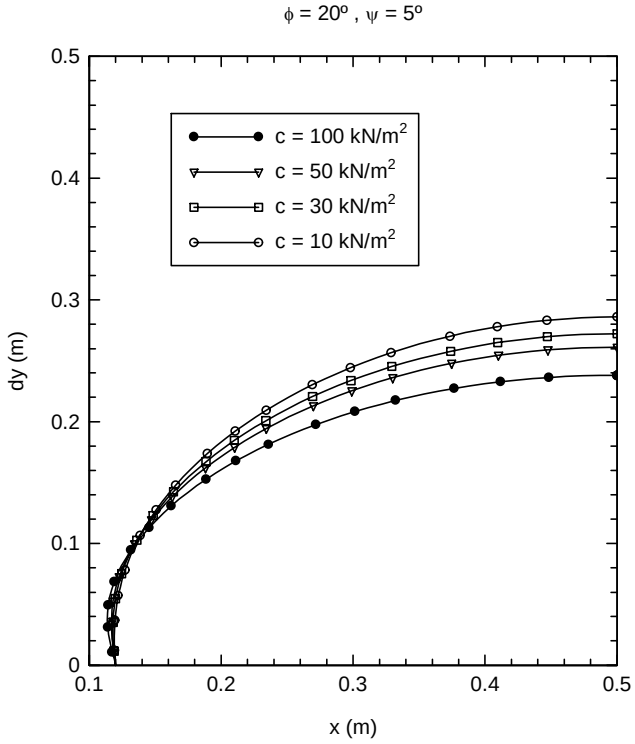


Figura C.15. Desplazamiento lateral impuesto. Influencia de la cohesión en la deformada de la malla (PLAXIS)

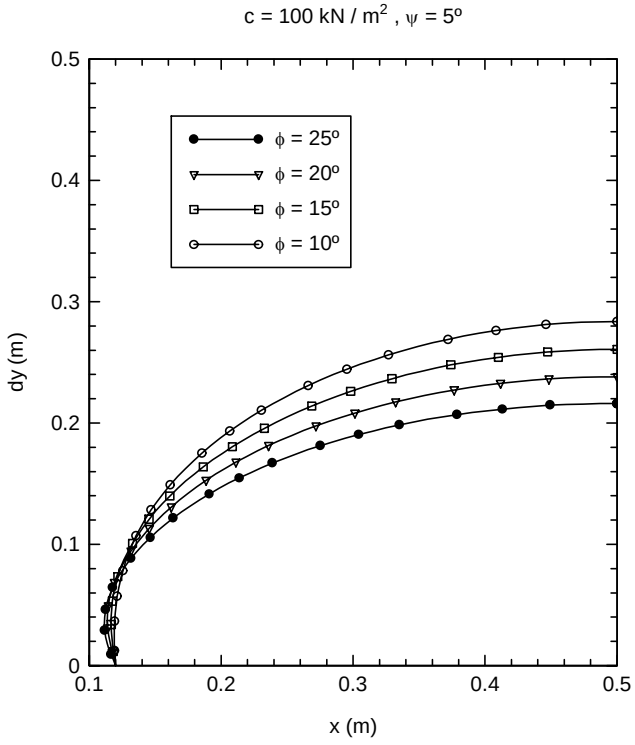


Figura C.16. Desplazamiento lateral impuesto. Influencia del ángulo de rozamiento en la deformada de la malla (PLAXIS)

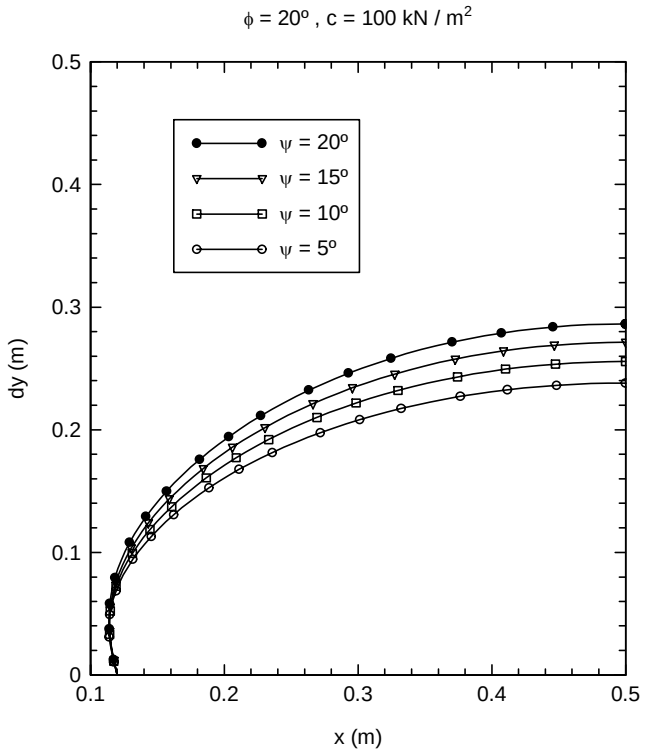


Figura C.17. Desplazamiento lateral impuesto. Influencia del ángulo de dilatancia en la deformada de la malla (PLAXIS)

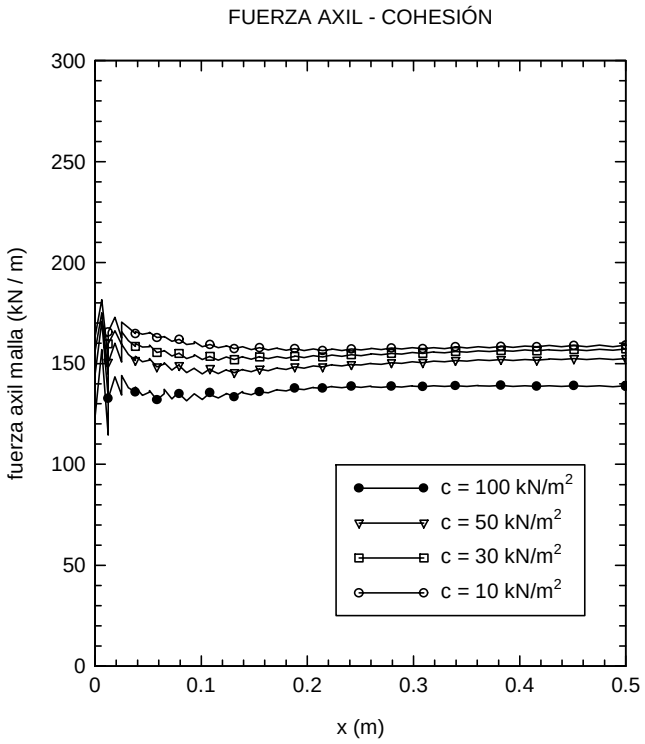


Figura C.18. Desplazamiento lateral impuesto. Fuerza axil en la malla en función de la cohesión del suelo (PLAXIS)

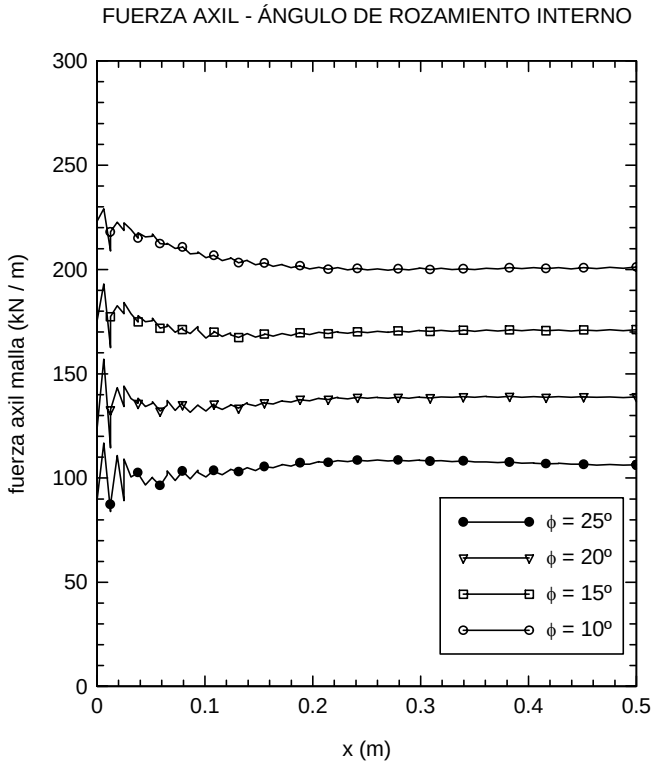


Figura C.19. Desplazamiento lateral impuesto. Fuerza axil en la malla en función del ángulo de rozamiento del suelo (PLAXIS)

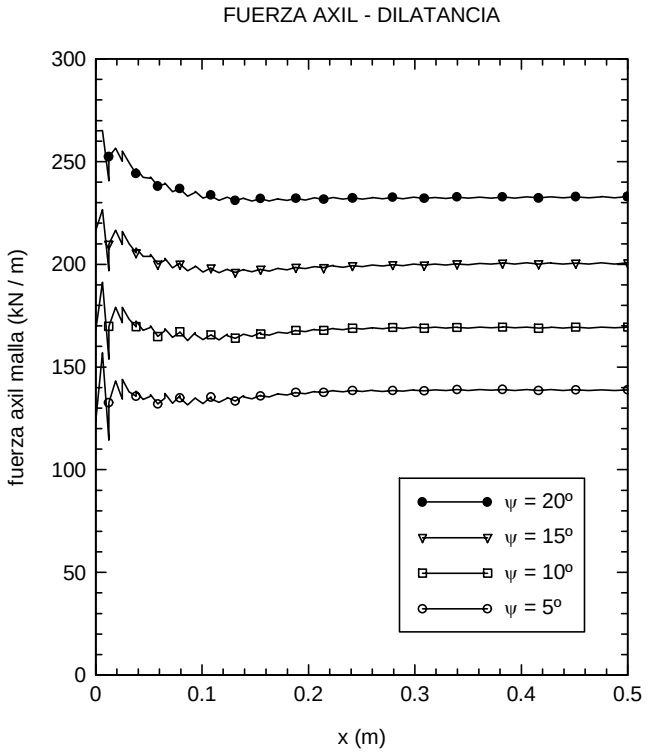


Figura C.20. Desplazamiento lateral impuesto. Fuerza axil en la malla en función del ángulo de dilatancia del suelo (PLAXIS)

C.5. EFECTO CONJUNTO DEL PRETENSADO DE ANCLAJES E INESTABILIDAD SUPERFICIAL. RESULTADOS DEL PROGRAMA PLAXIS

Se presentan los gráficos correspondientes a los resultados que se obtienen con respecto a la deformada que adquiere la malla y la fuerza axil en ella, debidos al efecto conjunto del pretensado de anclajes y la inestabilidad superficial.

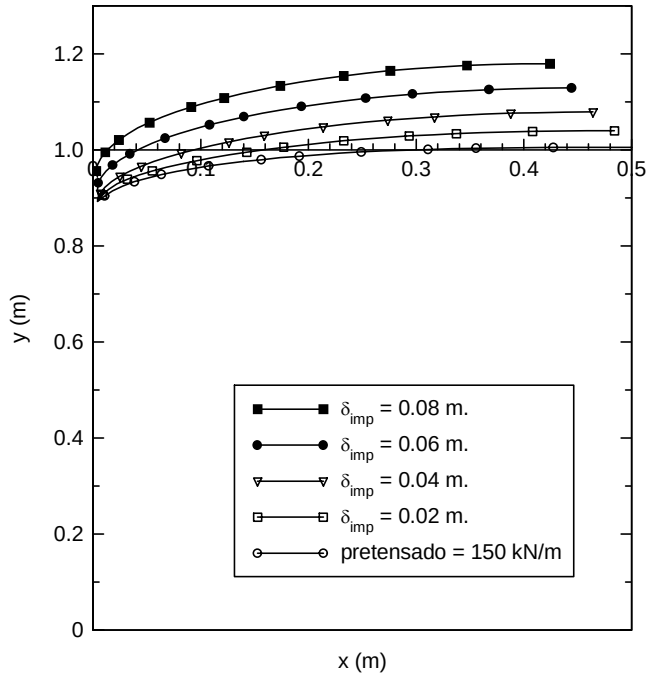


Figura C.21. Efecto pretensado-inestabilidad. Deformada de la malla en función del desplazamiento impuesto (PLAXIS)

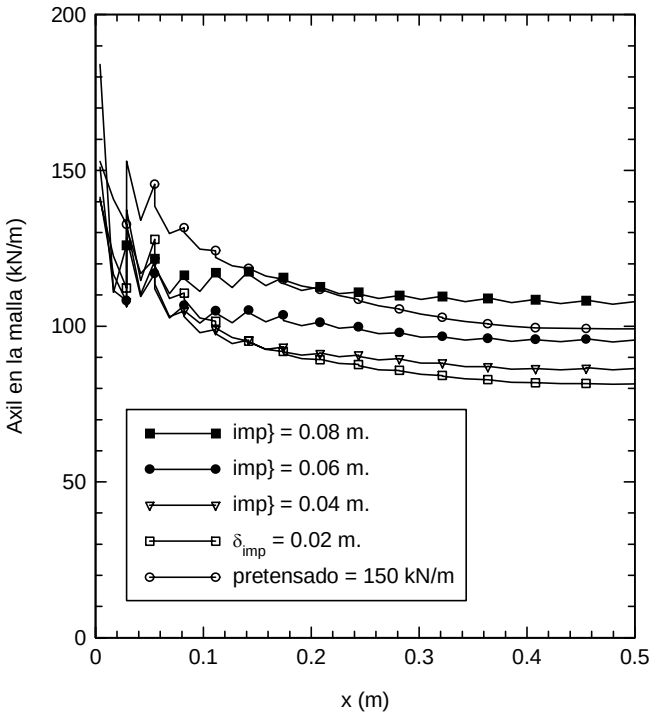


Figura C.22. Efecto pretensado-inestabilidad. Fuerza axil en la malla en función del desplazamiento impuesto (PLAXIS)

C.6. MALLA DE REFUERZO SOMETIDA A PRESIONES EN SUPERFICIE

Se presentan los resultados sobre la deformada de la malla, en función del espesor de suelo considerado, obtenidos con el empleo del programa CRISP90.

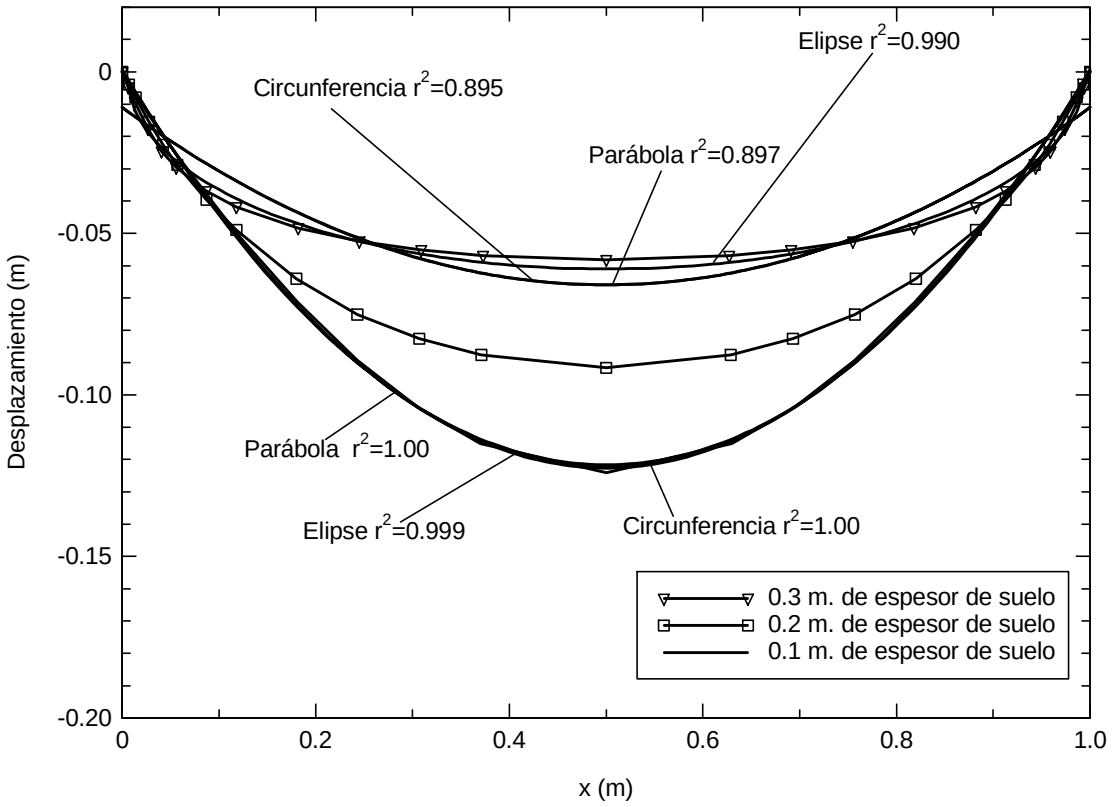
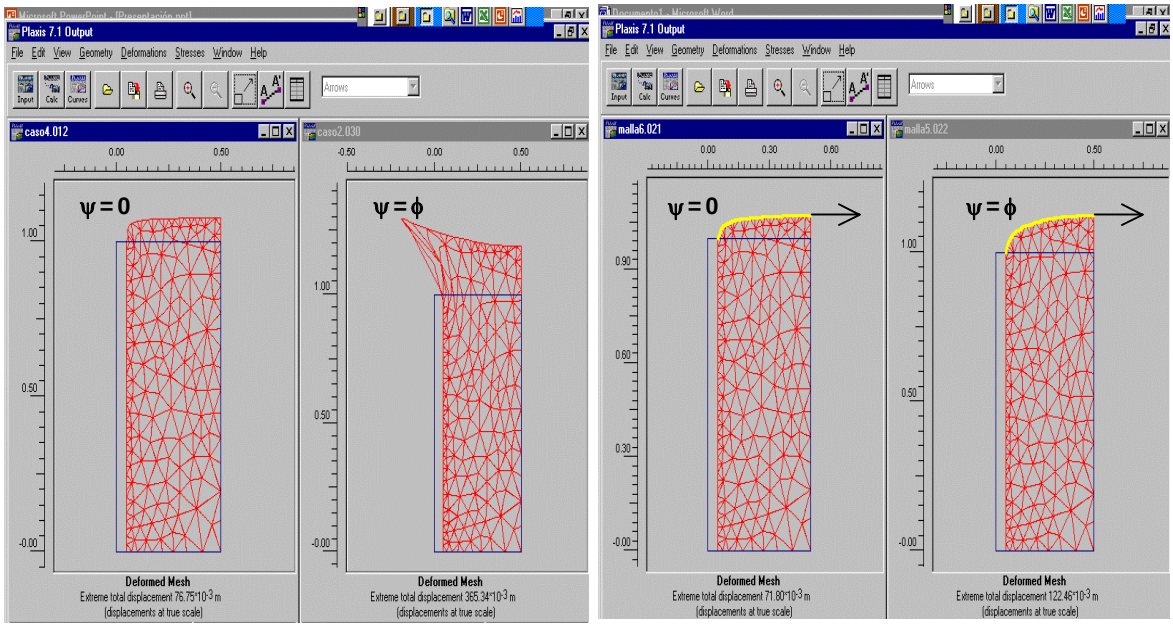


Figura C.23. Malla sometida a presión en superficie. Deformada de la malla de refuerzo en función del espesor de suelo (CRISP90)

C.7. EFECTO DE LA DILATANCIA DEL SUELO

Se presentan los resultados relacionados con el análisis con el programa PLAXIS, del efecto de la dilatancia del suelo.



SIN MALLA CON MALLA (Suelo confinado)

Figura C.24. Efecto de la dilatancia. Deformada de la malla (PLAXIS)

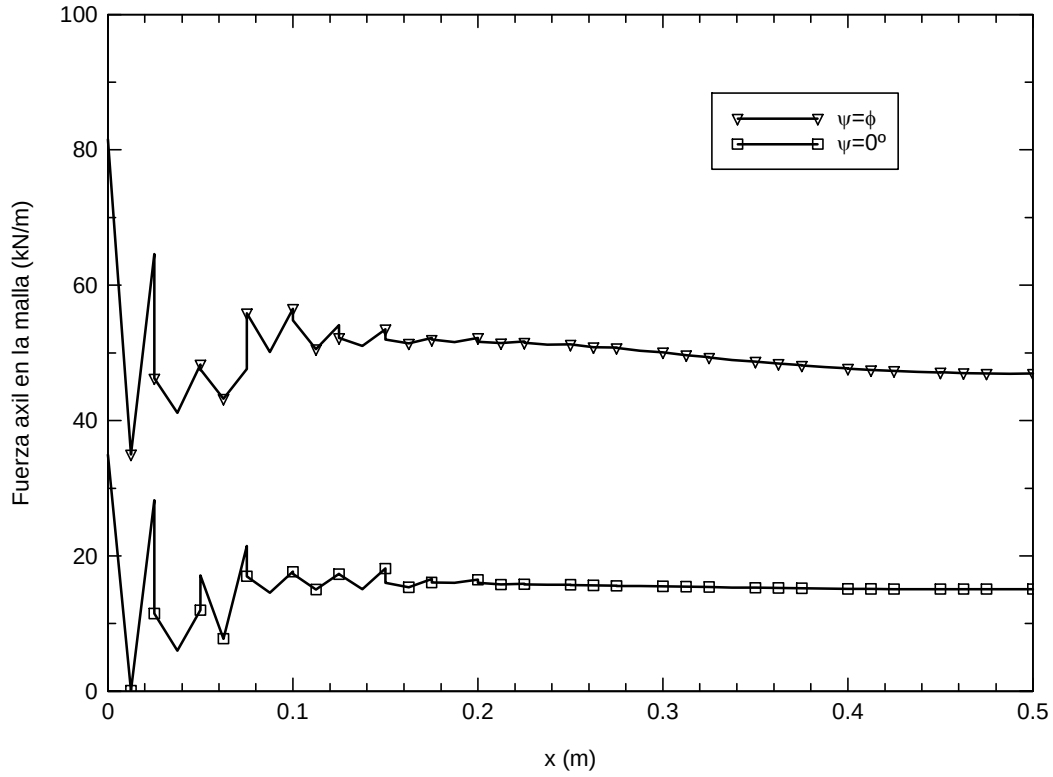


Figura C.25. Efecto de la dilatancia. Fuerza axil en la malla de refuerzo (PLAXIS)

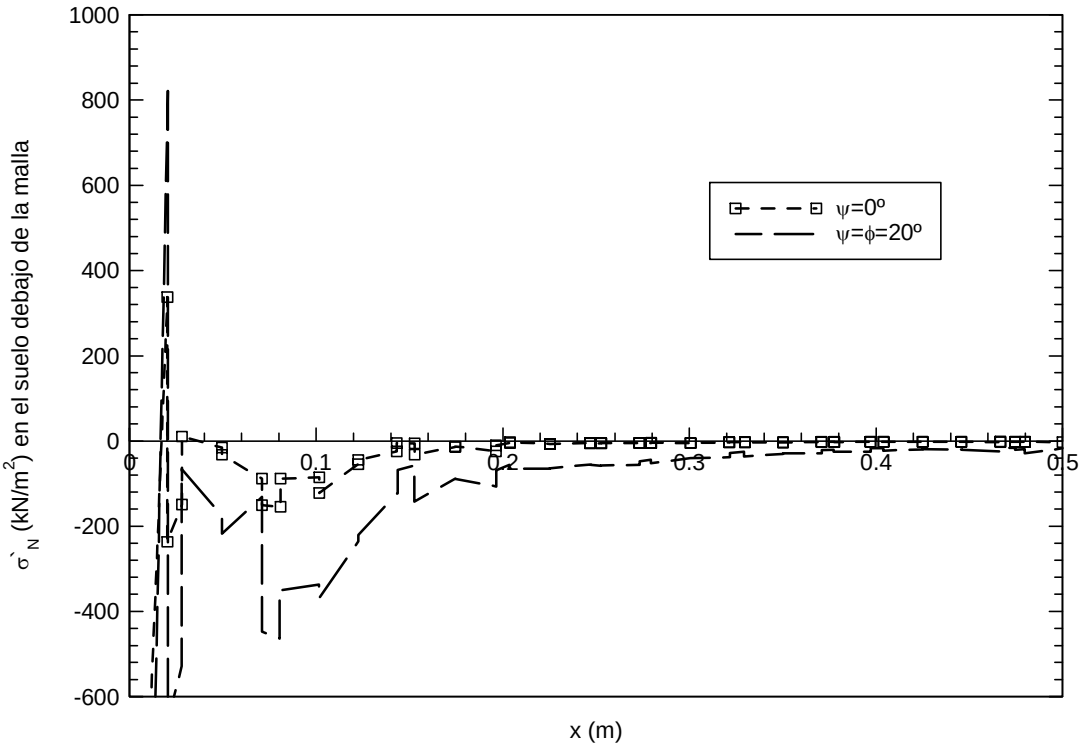


Figura C.26. Efecto de la dilatancia. Tensiones en el suelo bajo la malla de refuerzo (PLAXIS)

C.8. RESULTADOS DE LA REPRODUCCIÓN DEL ENSAYO DE CARGA DISTRIBUIDA

Se presentan los gráficos correspondientes a los resultados que se obtienen con respecto a la deformada que adquiere la malla debido a la acción de la carga distribuida en su superficie. Cada uno de los resultados obtenidos se acompaña del correspondiente al ensayo de laboratorio.

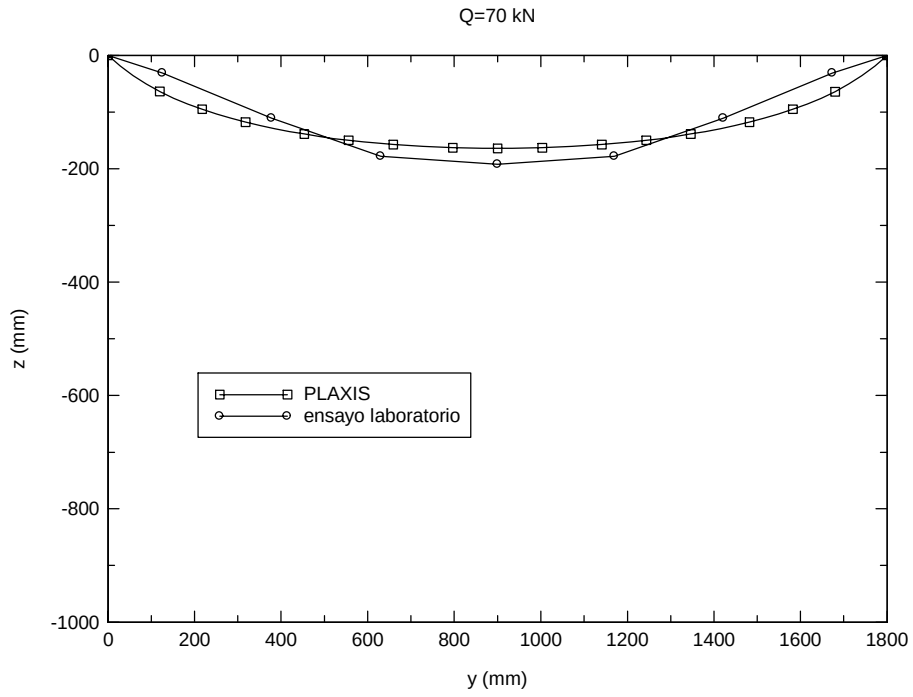


Figura C.27. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados numéricos (Q = 70 kN)

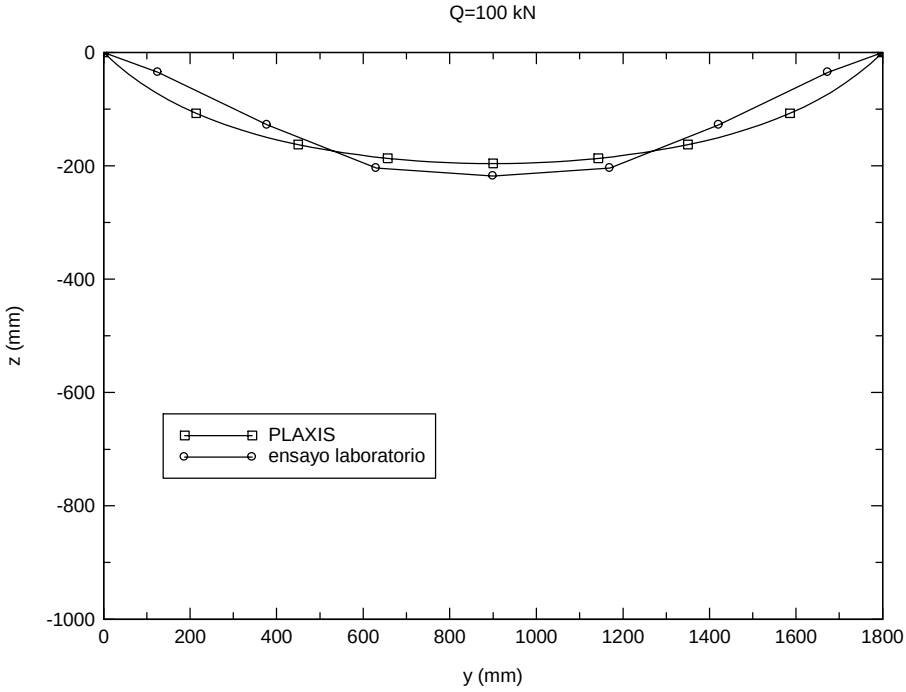


Figura C.28. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados numéricos (Q = 100 kN)

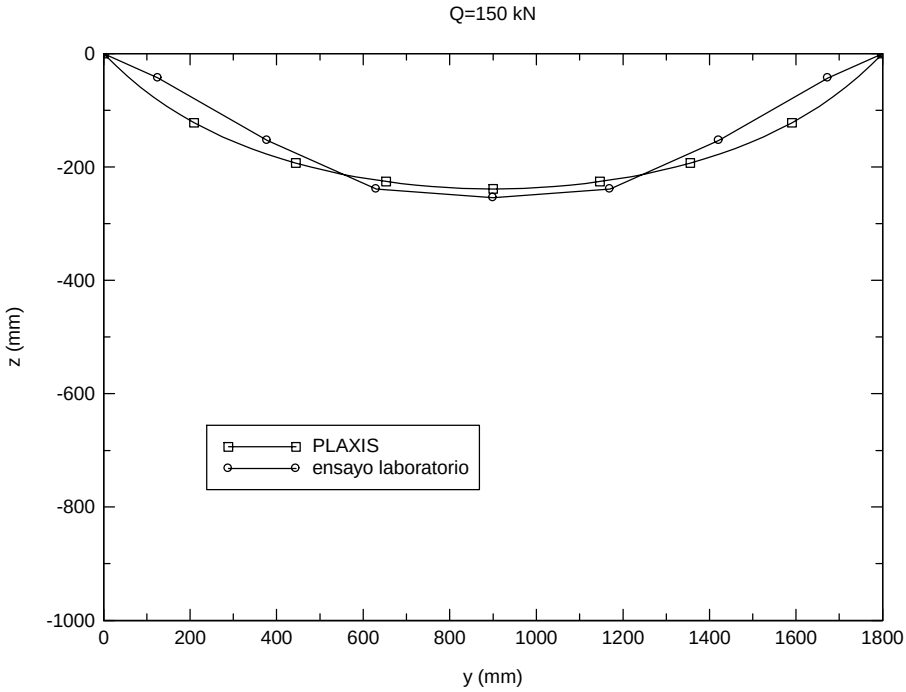


Figura C.29. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados numéricos (Q = 150 kN)

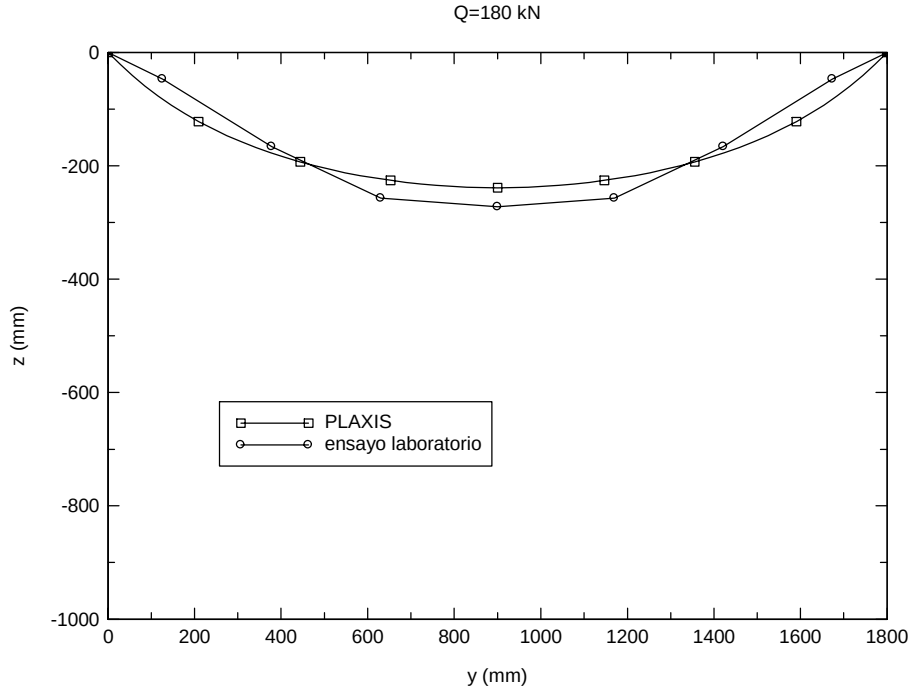


Figura C.30. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados numéricos (Q = 180 kN)

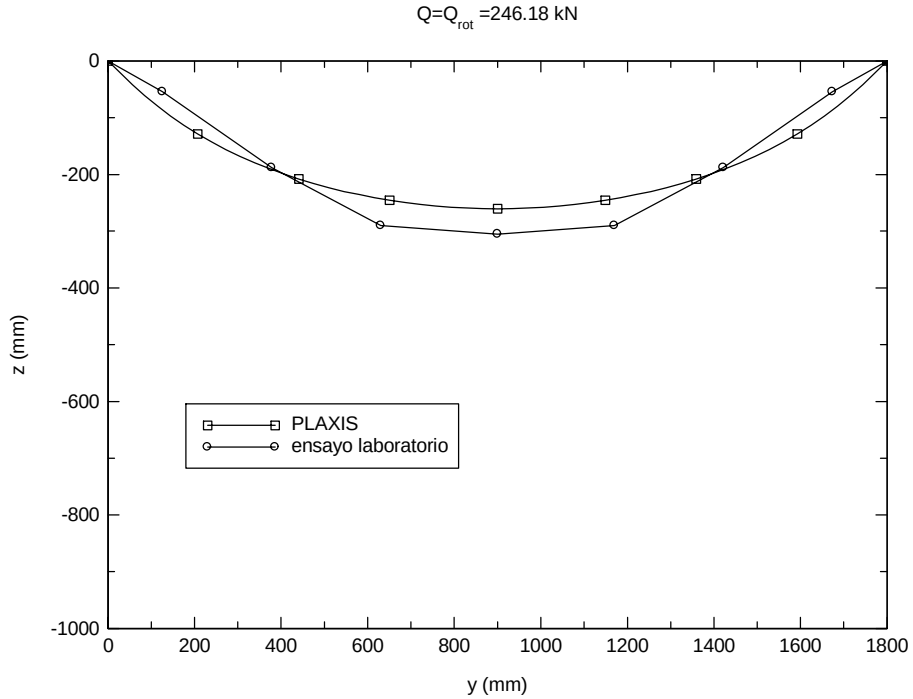


Figura C.31. Deformada de la malla de refuerzo sometida a carga distribuida sobre su superficie. Resultados numéricos (Q = Q_{rot} = 246.18 kN)

APÉNDICE D. DIAGRAMAS DE FLUJO DEL PROGRAMA OXFEM

D.1. DIAGRAMA DE FLUJO DEL PROGRAMA OXFEM

En la Figura D.1 se presenta el diagrama de flujo correspondiente al programa OXFEM. En él se presentan las subrutinas más importantes y representativas, las cuales se describen de forma muy breve a continuación.

- maxdim.- establece las máximas dimensiones para varias variables empleadas por el programa
- values.- establece valores de varias variables empleadas por el programa
- utils.- establece el valor de varias constantes empleadas por el programa
- var_restart.- definición de variables globales del programa
- not_restart.- definición de variables globales del programa
- var_alloc.- definición de variables dinámicas
- startup.- escribe en el fichero de salida los encabezados correspondientes al análisis
- initialise.- inicializa las variables necesarias para el cálculo
- allocate_arrays.- asignación de matrices empleadas por el programa
- input.- lee el fichero de entrada de datos
- echo_input_data.- escribe los datos del fichero de entrada a otro fichero
- standard_set_up.- establece varios vectores y matrices antes del comienzo del cálculo
- proset.- genera la matriz que contiene la propiedades de los elementos
- codes.- genera las matrices que relacionan el número de un grado de libertad con el número de nodo, y al revés
- initia.- calcula las tensiones iniciales

- body_forces.- calcula la fuerzas correspondientes al peso propio
- external_loads.- establece las cargas exteriores para cada fase de cálculo
- distrib_loads.- establece las cargas distribuidas para cada fase de cálculo
- excavation.- establece las fuerzas debidas a la excavación o adicción de elementos para cada fase de cálculo
- prescribed_disps.- establece los desplazamientos impuestos
- bound_conds.- establece las condiciones de contorno
- change_properties.- cambia las propiedades de los elementos si es necesario
- surface_loads.- calcula el vector de fuerzas nodales equivalentes
- frontl.- ensambla las matrices de rigidez K de los elementos, y resuelve $\Delta F = K \cdot \Delta\delta$ obteniendo el incremento de desplazamientos
- elcod.- calcula las coordenadas de los nodos de un elemento
- stiffness.- calcula la matriz de rigidez de un elemento
- gauss.- calcula las funciones y pesos de integración
- shape.- calcula las funciones de forma
- bmatrix.- calcula la matriz B relacionada con la geometría del problema
- dmorg.- selecciona la subrutina adecuada para calcular la matriz de rigidez de un elemento, según el modelo de comportamiento del elemento
- update.- actualiza el valor de las tensiones y de las fuerzas
- strain.- calcula los incrementos de deformación correspondientes a los incrementos de desplazamiento

- stress.- calcula las tensiones correspondientes a un incremento de las deformaciones

- elast.- calcula el incremento de tensiones dentro del rango elástico

- plast.- llama a la subrutina correspondiente según el modelo de comportamiento, la cual calcula el incremento de tensiones en el rango plástico

- inter.- llama a la subrutina correspondiente según el modelo de comportamiento, la cual calcula la intersección con la superficie de plastificación

- yout.- llama a la subrutina correspondiente según el modelo de comportamiento, la cual comprueba si el punto se encuentra en una zona válida del espacio de tensiones o no

- dyfun.- llama a la subrutina correspondiente según el modelo de comportamiento, la cual calcula el incremento del valor de la función de plastificación

- numint.- calcula los incrementos de tensiones mediante la ecuación $\dot{\sigma} = D \cdot \dot{\varepsilon}$, aplicando el método de Runge-Kutta

- force.- calcula las fuerzas nodales equivalentes a las tensiones

- check_converge.- comprueba la convergencia de la solución

- coord_update.- actualiza las coordenadas en análisis con cambio de geometría

- step_output.- escribe resultados correspondientes a un escalón de carga en los ficheros de salida

- reset_displacements.- pone los desplazamientos a cero si así está indicado en la fase de cálculo

- estimate_end.- estima el tiempo restante hasta el final del análisis

- stage_output.- escribe resultados correspondientes a cada fase del análisis en los ficheros de salida

- end_output.- salida y limpieza de las variables empleadas en el cálculo

- deallocate_arrays.- desasignación de matrices empleadas en el programa


program OXFEM
use maxdim
use values
use utils
use var_ restart ! módulos empleados por el programa
use not_restart
use var_alloc
use var_derived
call startup
analysis: do ianalysis = 1, nanalysis ! para todos los análisis
call initialise
call allocata_arrays
call INPUT
call echo_input_data
call standarad_set_up
call PROSET
call CODES
call INITIA
call body_forces
SREG = 1
SSTEP = 1
stage: do IREG=SREG, NREG ! para todas las fases de cálculo
if (SSTEP == 1) then ! si es el primer escalón de carga
EXLOD, EXCLOD, PRES, DODLOD, SURLOD, SCHLOD, grdol NFIX, EXFLAG = 0
call external_loads
call distrib_loads
call excavation
call prescribed_disps
call bound_conds
call change_properties
else
end if
step: do IST = SSTEP, ISTEP ! para todos los escalones de carga
DDELTA = 0 DREACT = 0 ! inicialización de desplazamientos y reacciones
MULT = 1.0/REAL(ISTEP)
call surface_loads
set_up_forces: do LGDOF=1 , NGDOF ! para todos los puntos de Gauss
FUBAL = FCURR = DEXLOD= DPRES = ! cálculo de fuerzas
end do set_up_forces
call FORNTL
do IELE =1 , NELE _ ! para todos los elementos
call ELCOD
call STIFFNESS
call GAUSS
do IGAU = 1, NGAU ! para todos los puntos de Gauss
call SHAPE
call BMATRIX
call DMORG
end do
ensamblaje de las matrices de rigidez de los elementos
(1)
(2)
(3)
(4)
(5)
(6)
(7)

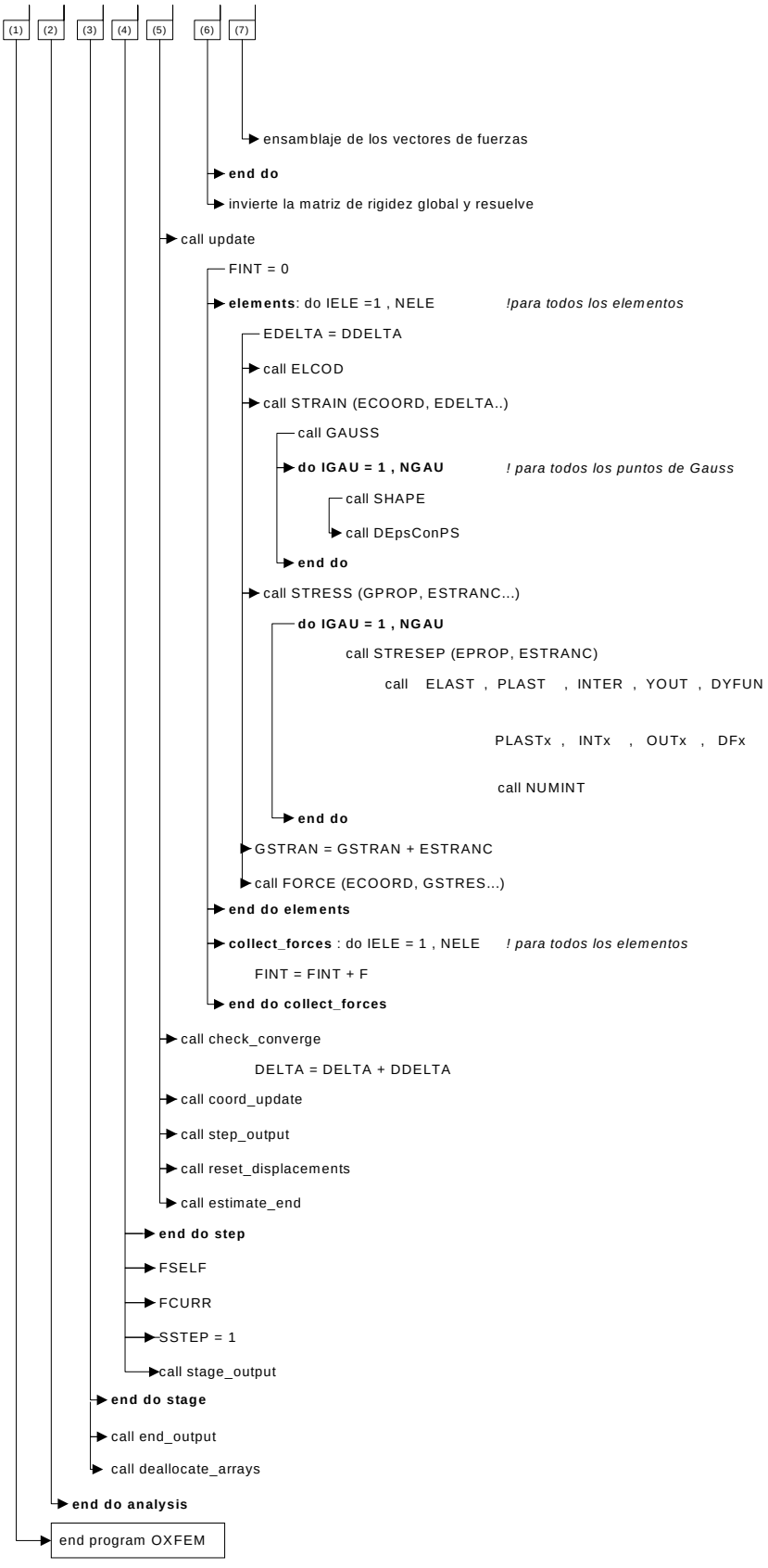


Figura D.1. Diagrama de flujo del programa OXFEM

D.2. SUBRUTINAS MODIFICADAS Y CREADAS PARA INTRODUCCIÓN DEL MODELO MATSUOKA-NAKAI GENERALIZADO EN EL PROGRAMA OXFEM

En la Figura D.2 se presenta de nuevo el diagrama de flujo correspondiente al diagrama OXFEM, indicando en este caso mediante subrayado las subrutinas que ha sido necesario modificar para introducir el modelo de comportamiento Matsuoka-Nakai generalizado, y mediante sombreado las nuevas subrutinas que ha sido necesario crear y que se describen brevemente a continuación.

- mtymd.- calcula la matriz de rigidez de un elemento cuyo modelo de comportamiento es el de Matsuoka-Nakai generalizado

- plastm.- calcula el incremento de tensiones en el rango plástico para elementos con modelo de comportamiento Matsuoka-Nakai generalizado

- intm.- calcula la intersección con la superficie de plastificación para elementos con modelo de comportamiento Matsuoka-Nakai generalizado

- outm.- comprueba si un punto de Gauss se encuentra en una zona válida del espacio de tensiones o no, para elementos con modelo de comportamiento Matsuoka-Nakai generalizado

- dfm.- calcula el incremento del valor de la función de plastificación para elementos con modelo de comportamiento Matsuoka-Nakai generalizado

- yfunm.- calcula el valor de la función de plastificación en un punto de Gauss, para elementos con modelo de comportamiento Matsuoka-Nakai generalizado

- matchkm1 y matchkm2. comprueban que los puntos de Gauss se encuentran en la zona válida del espacio de tensiones

- correctmc.- corrige las tensiones a la superficie de plastificación para elementos con modelo de comportamiento Matsuoka-Nakai generalizado

- vectorv.- calcula la dirección empleada en la corrección de tensiones para elementos con modelo de comportamiento Matsuoka-Nakai generalizado

- cosineteta.- calcula en ángulo que forma el eje correspondiente a $\sigma_1=\sigma_2=\sigma_3$, con el vector resultante de unir el vértice de la superficie de plastificación con un punto de gauss

- cosinetetamax.- calcula el ángulo de abertura máximo de la superficie de plastificación

- choosebi.- elige la solución correcta en la corrección de tensiones a la superficie de plastificación


program OXFEM
use maxdim
use values
use utils
use var_ restart ! módulos empleados por el programa
use not_restart
use var_alloc
use var derived
call startup
analysis: do ianalysis = 1, nanalysis ! para todos los análisis
call initialise
call allocate_arrays
call INPUT
call echo_input_data
call standarad_set_up
call PROSET
call CODES
call INITIA
call body_forces
SREG = 1
SSTEP = 1
stage: do IREG=SREG, NREG ! para todas las fases de cálculo
if (SSTEP == 1) then ! si es el primer escalón de carga
EXLOD, EXCLOD, PRES, DODLOD, SURLOD, SCHLOD, grdol NFIX, EXFLAG = 0
call external_loads
call distrib_loads
call excavation
call prescribed_disps
call bound_conds
call change_properties
else
end if
step: do IST = SSTEP, ISTEP ! para todos los escalones de carga
DDELTA = 0 DREACT = 0 ! inicialización de desplazamientos y reacciones
MULT = 1.0/REAL(ISTEP)
call surface_loads
set_up_forces: do LGDOF=1 , NGDOF ! para todos los puntos de Gauss
FUBAL = FCURR = DEXLOD= DPRES = ! cálculo de fuerzas
end do set_up_forces
call FORNTL
do IELE =1 , NELE ! para todos los elementos
call ELCOD
call STIFFNESS
call GAUSS
do IGAU = 1, NGAU ! para todos los puntos de Gauss
call SHAPE
call BMATRIX
call DMORG
call mtymd
end do
ensamblaje de las matrices de rigidez de los elementos
(1)
(2)
(3)
(4)
(5)
(6)
(7)

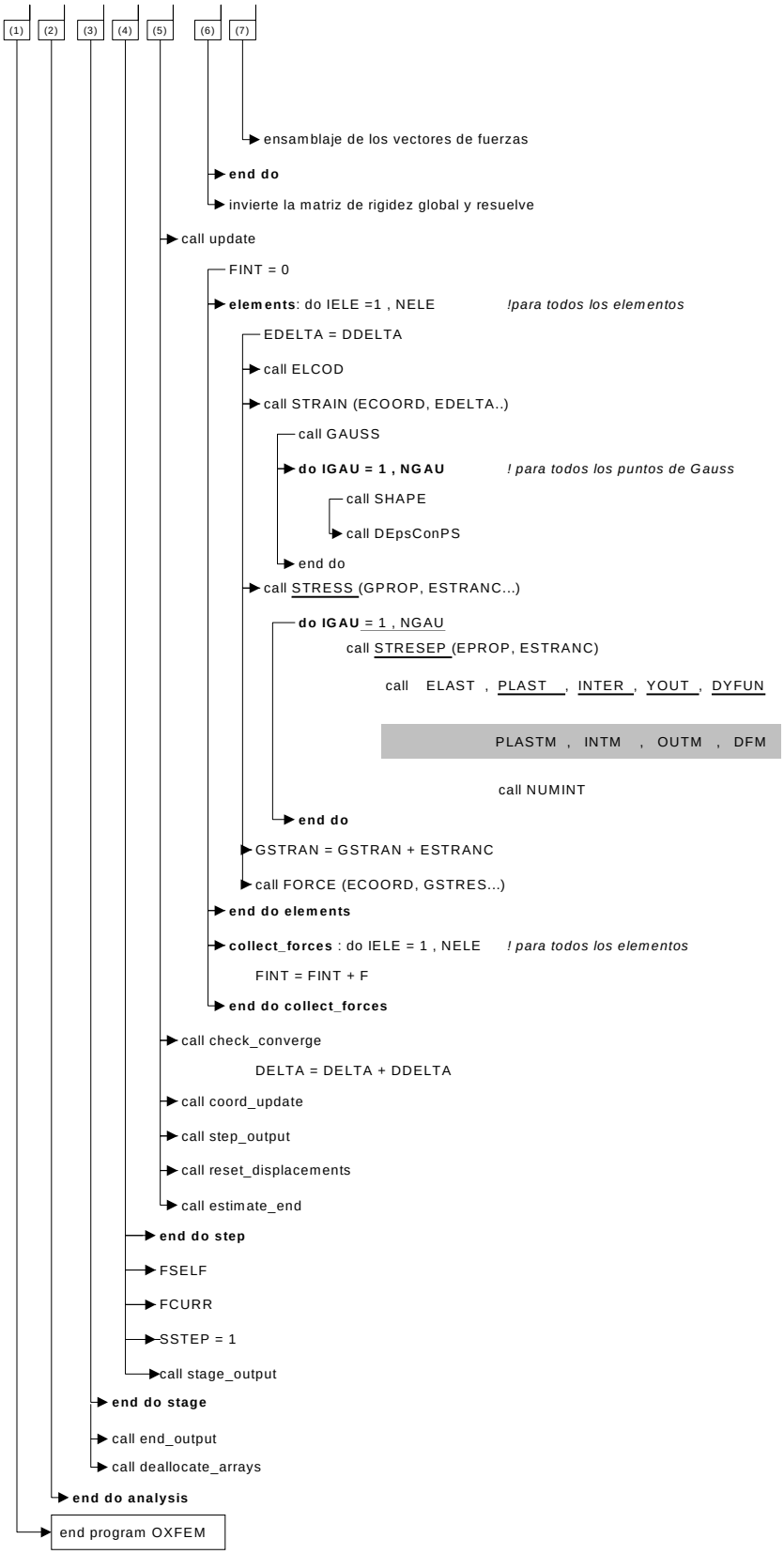


Figura D.2. Subrutinas modificadas y creadas para introducir el modelo de Matsuoka-Nakai genralizado en OXFEM

D.3. SUBRUTINAS MODIFICADAS Y CREADAS PARA INTRODUCCIÓN DE LA FORMULACIÓN DE LA DEGRADACIÓN EN EL PROGRAMA OXFEM

En la Figura D.3 se presenta un diagrama de flujo del programa OXFEM en el cual aparecen subrayadas las subrutinas que ha sido necesario modificar para introducir la formulación correspondiente a la degradación, y sombreada la que ha sido necesario crear llamada degradation. Esta subrutina llama a su vez a otras, también relacionadas con el cálculo de la degradación, para al final obtener el incremento de fuerzas en un escalón de carga, debidas al cambio del valor de la cohesión en los puntos de Gauss. A continuación se describen todas ellas.

- degradation.- es la subrutina general en la que al final se obtiene el incremento de fuerzas en un escalón de carga, debidas al cambio del valor de la cohesión en los puntos de Gauss

- stresdegra.- emplea el método de Runge-Kutta para la obtención del incremento de tensiones en un punto de Gauss, correspondiente al cambio del valor de la cohesión en un escalón de carga

- fixstresdegra.- emplea el método de Runge-Kutta para la obtención del incremento de tensiones en un punto de Gauss, correspondiente al cambio del valor de la cohesión en un escalón de carga, pero a diferencia de la anterior lo hace para un número fijo de pasos impuestos en el método de Runge-Kutta

- strsdegracal.- esta subrutina es llamada por las dos anteriores para calcular el incremento de tensiones en un punto de Gauss, correspondiente al cambio del valor de la cohesión, pero no del escalón de carga completo, sino de cada uno de los pasos en que se divide el método de Runge-Kutta

- origstrs.- corrige las tensiones de un punto de Gauss al vértice de la superficie de plastificación

- check materia.- comprueba que el tipo de material es susceptible de ser degradado


program OXFEM
use maxdim
use values
use utils
use var_ restart ! módulos empleados por el programa
use not_restart
use var_alloc
use var derived
call startup
analysis: do ianalysis = 1, nanalysis ! para todos los análisis
call initialise
call allocate_arrays
call INPUT
call echo_input_data
call standarad_set_up
call PROSET
call CODES
call INITIA
call body_forces
SREG = 1
SSTEP = 1
stage: do IREG=SREG, NREG ! para todas las fases de cálculo
if (SSTEP == 1) then ! si es el primer escalón de carga
EXLOD, EXCLOD, PRES, DODLOD, SURLOD, SCHLOD, grdol NFIX, EXFLAG = 0
call external_loads
call distrib_loads
call excavation
call prescribed_disps
call bound_conds
call change_properties
else
end if
step: do IST = SSTEP. ISTEP ! para todos los escalones de carga
DDELTA = 0 DREACT = 0 ! inicialización de desplazamientos y reacciones
MULT = 1.0/REAL(ISTEP)
call surface_loads
if IREG =INT(DEGRAD (IREG)) then ! si es una fase de degradación
call DEGRADATION
end if
set_up_forces: do LGDOF=1 , NGDOF ! para todos los puntos de Gauss
FUBAL = FCURR = DEXLOD= DPRES = ! cálculo de fuerzas
end do set_up_forces
call FORNTL
do IELE =1 , NELE_ ! para todos los elementos
call ELCOD
call STIFFNESS
call GAUSS
do IGAU = 1, NGAU ! para todos los puntos de Gauss
call SHAPE
call BMATRIX
call DMORG call mtymd
end do
ensamblaje de las matrices de rigidez de los elementos
(1)
(2)
(3)
(4)
(5)
(6)
(7)

Figura D.3. Subrutinas modificadas y creadas para introducir la formulación de la degradación en OXFEM

APÉNDICE E. RESULTADOS DE CARGA DE HUNDIMIENTO. MODELO MATSUOKA-NAKAI GENERALIZADO

E.1. RESULTADOS

En el Capítulo 5 se ha presentado el análisis de carga de hundimiento de una cimentación con objeto de verificar el funcionamiento del modelo de Matusoka-Nakai generalizado implementado en el programa OXFEM.

En este apéndice se presentan los gráficos correspondientes a los resultados obtenidos para los distintos casos analizados correspondientes a diferentes valores de los parámetros resistentes del suelo.

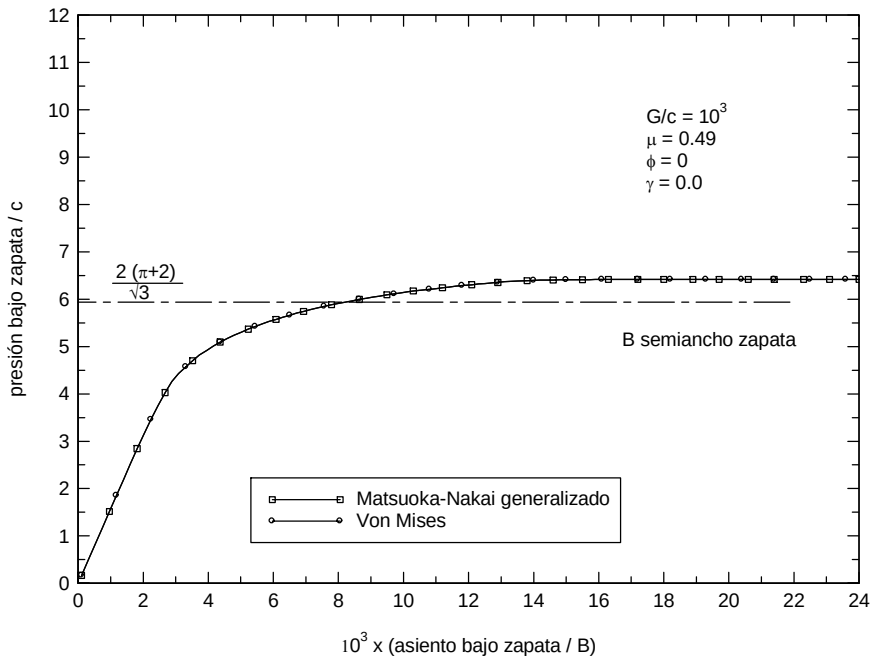


Figura E.1. Carga de hundimiento (Comparación Matsuoka-Nakai generalizado con $\phi = 0$ y Von Mises)

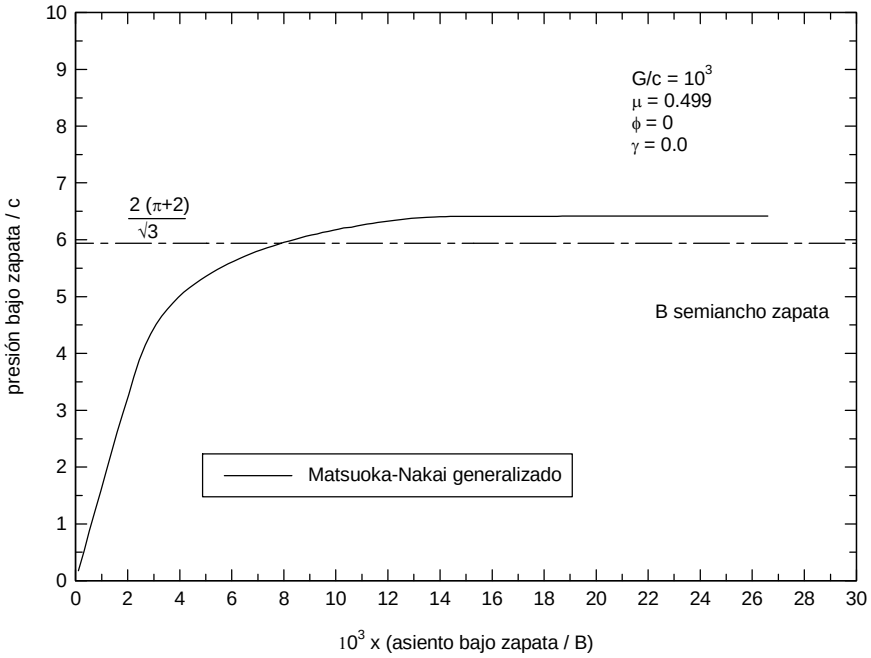


Figura E.2. Carga de hundimiento ($\phi = 0$)

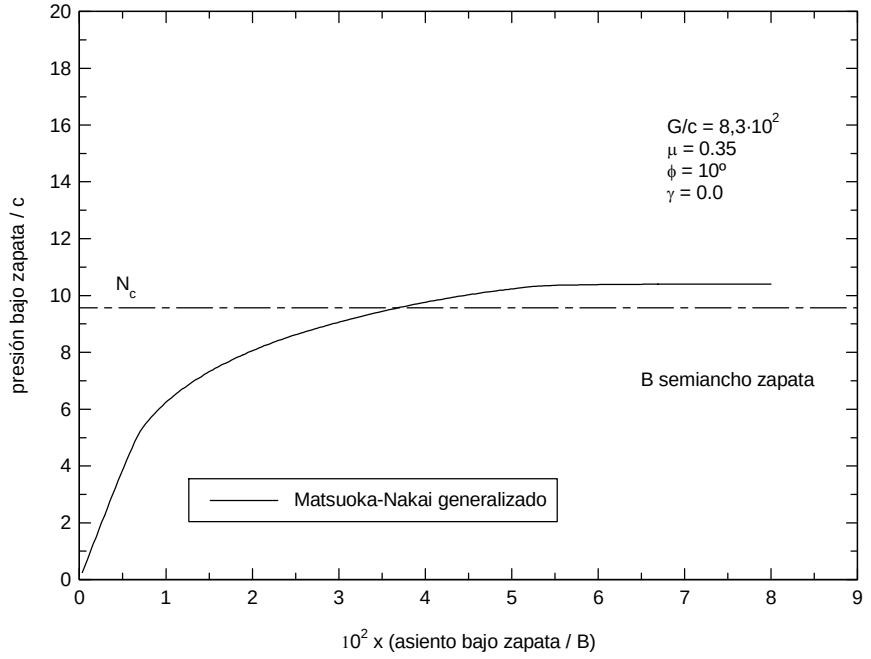


Figura E.3. Carga de hundimiento ($\phi = 10º$)

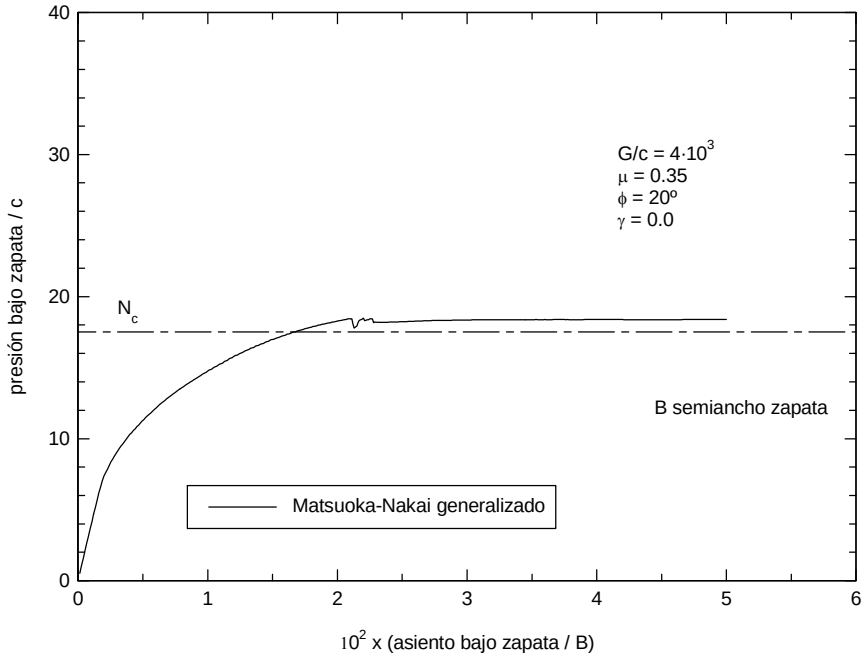


Figura E.4. Carga de hundimiento (ϕ = 20º)

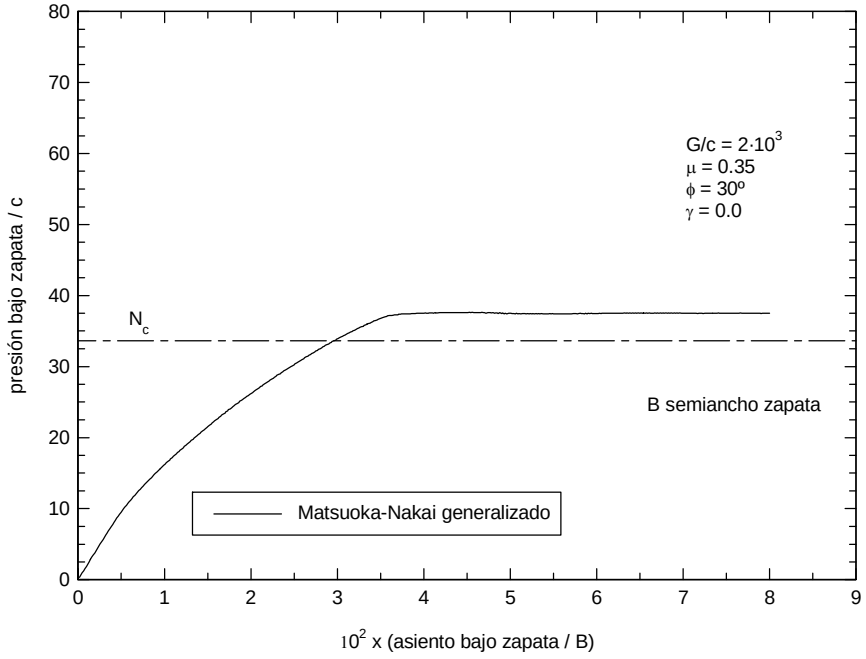


Figura E.5. Carga de hundimiento (ϕ = 30º)

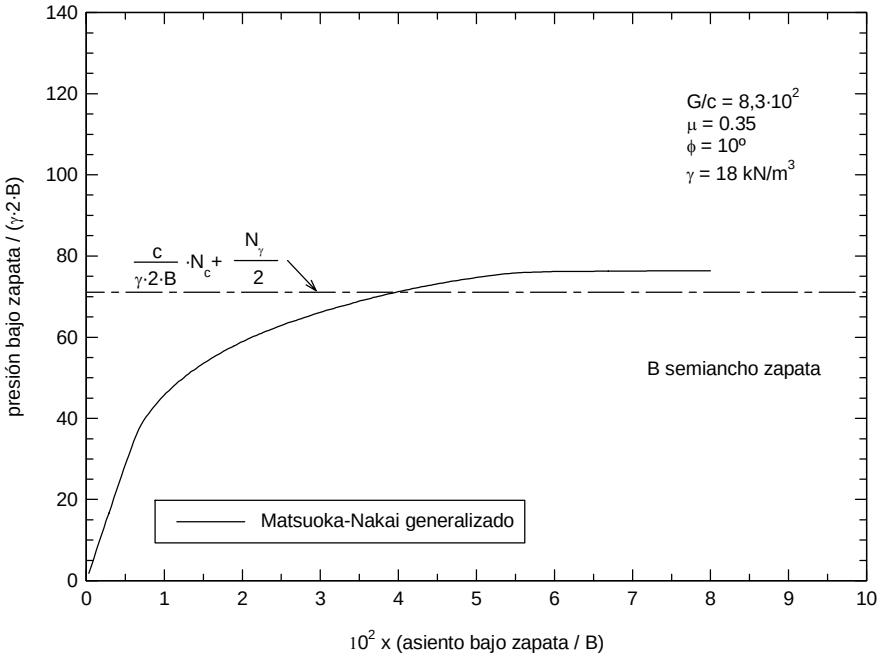


Figura E.6. Carga de hundimiento (c, ϕ, γ)

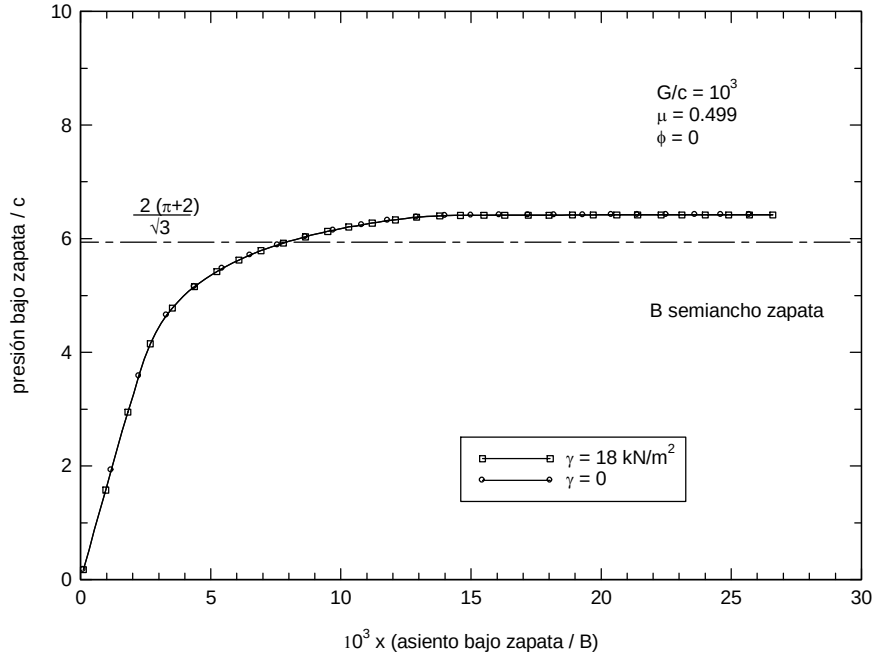


Figura E.7. Carga de hundimiento (c, γ)

APÉNDICE F. RESULTADOS DEL ANÁLISIS DE DEGRADACIÓN EN UNA CIMENTACIÓN SUPERFICIAL

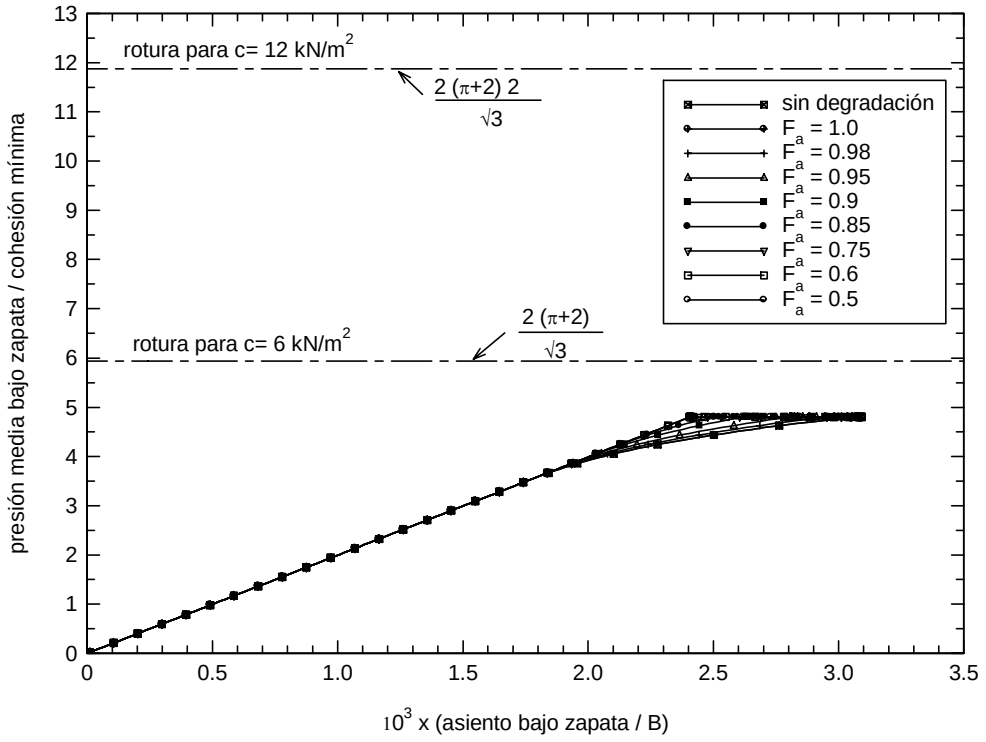
F.1. INTRODUCCIÓN

En el Capítulo 5 se ha presentado el análisis con el programa OXFEM de una cimentación superficial sometida en primer lugar a una fuerza vertical, y posteriormente a una degradación del terreno bajo el plano de cimentación.

Dicho análisis se realizó con objeto de comprobar el funcionamiento de la formulación desarrollada para representar la degradación del terreno en un caso más sencillo que el correspondiente a un talud. El análisis se ha realizado para dos casos, uno con ángulo de rozamiento nulo, y otro con ángulo de rozamiento igual a 15º. En el Capítulo 5 se han descrito las condiciones en cuanto a fuerza aplicada en la zona de la zapata, y en cuanto a las características de la degradación impuesta al terreno bajo ella para cada caso.

En este Apéndice se presentan los gráficos correspondientes a los resultados obtenidos. Por un lado se presentan las curvas carga-desplazamiento correspondientes a todo el proceso (aplicación de carga y fase de degradación del terreno), dadas de forma adimensional; y por otro, las curvas correspondientes al asiento que se va produciendo bajo el plano de cimentación a medida que avanzan los frentes de degradación, dadas también en forma adimensional. En estas últimas gráficas, los valores de las variables 'd1' y 'd2' representan la distancia recorrida hasta el momento por el primer y segundo frente de degradación respectivamente.

F.2. RESULTADOS CORRESPONDIENTES AL CASO $\phi = 0^o$



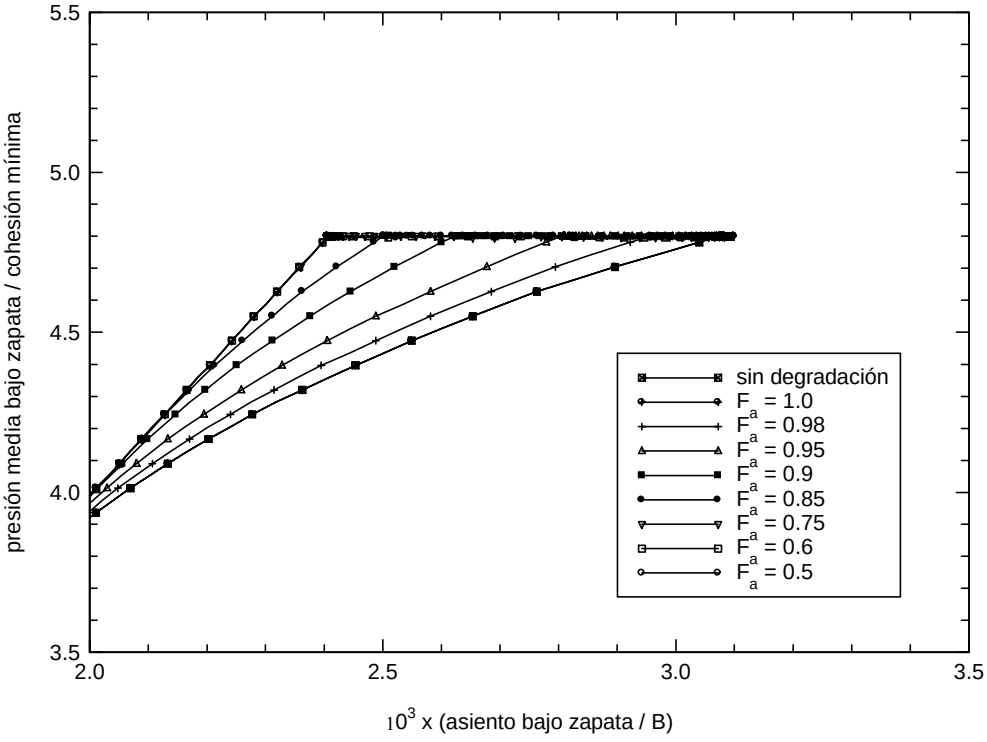
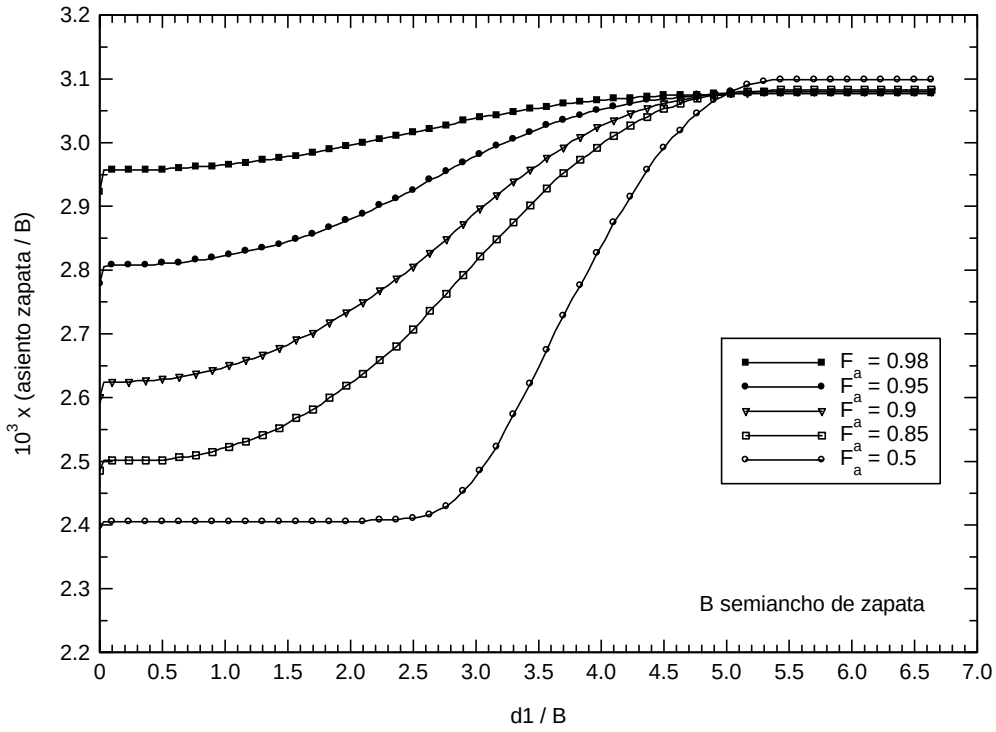


Figura F.1. Resultados de degradación en el caso de una cimentación superficial ($\phi = 0$)



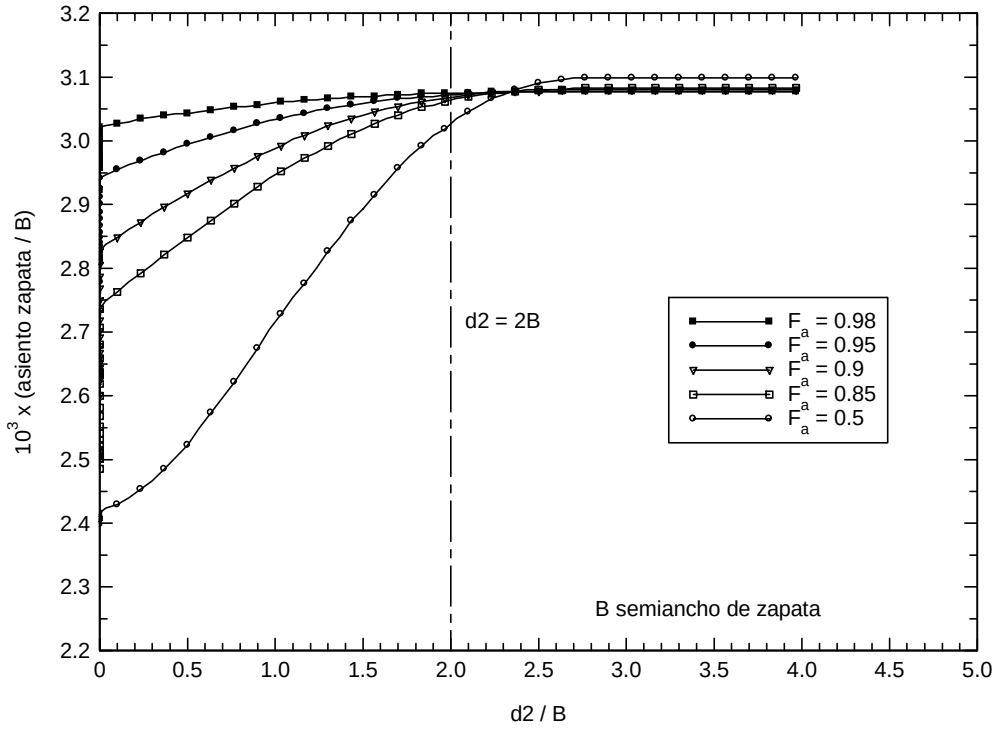
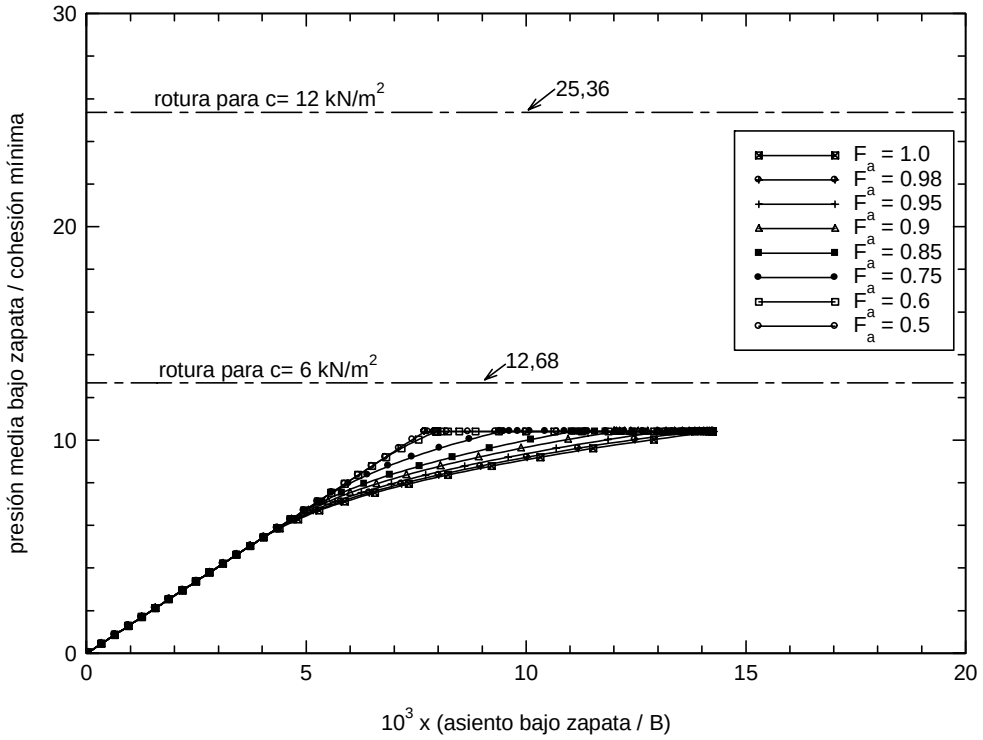


Figura F.2. Relación entre asiento bajo cimentación y frentes de degradación en el caso de una cimentación superficial ($\phi = 0$)

F.3. RESULTADOS CORRESPONDIENTES AL CASO ϕ = 15º



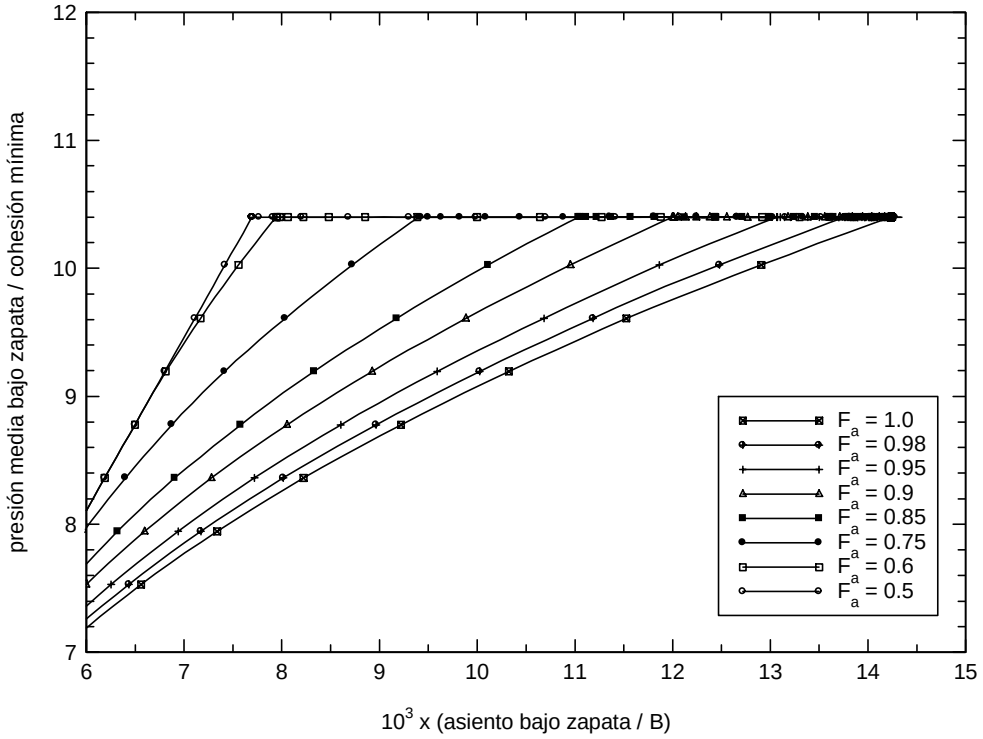


Figura F.3. Resultados de degradación en el caso de una cimentación superficial (ϕ =15º)

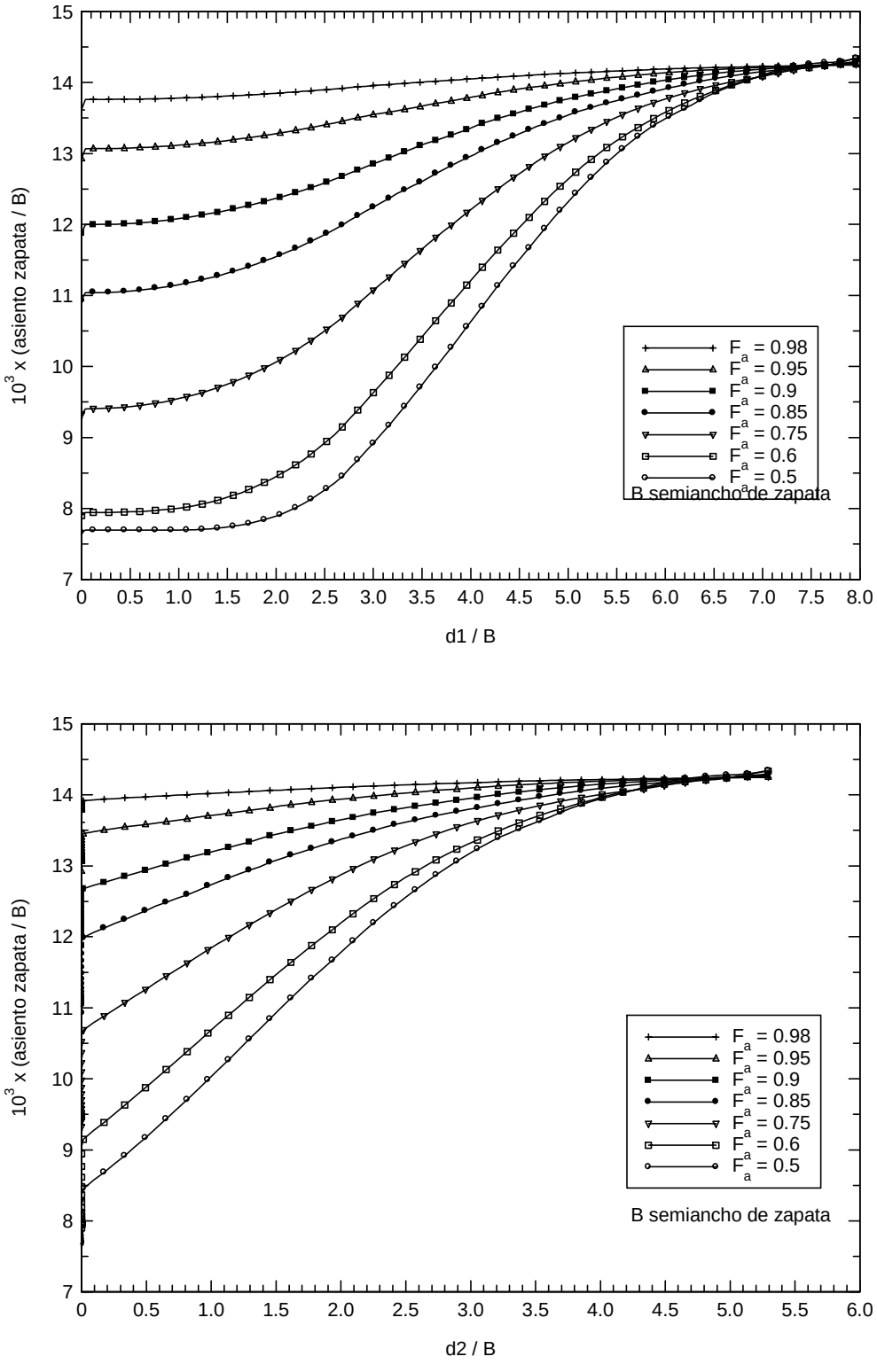


Figura F.4. Relación entre asiento bajo cimentación y frentes de degradación en el caso de una cimentación superficial (ϕ = 15º)

APÉNDICE G. CÓDIGO DE LAS SUBRUTINAS DESARROLLADAS

G.1. CÓDIGO DE LA SUBRUTINA PARA CÁLCULO DE LA MATRIZ DE RIGIDEZ CON CRITERIO DE PLASTIFICACIÓN DE MATSUOKA-NAKAI GENERALIZADO

```
SUBROUTINE MTYMD (INDEX,PROP,STRES,STATE,D)
!/***********************************************************************
!    This subroutine returns the material stiffness matrix
!    for a Matsuoka with cohesion material with constant elastic parameters
!    using a plastic potential to derive plastic strain increments.
!
!    If STATE(1) = Y = 0  then the elastic stiffness matrix is
!    returned.
!    If STATE(1) = Y = 1  then the elastic-plastic stiffness matrix is
!    returned.
!
!   CREATED BY   A. DA COSTA 9/02/2002
!***********************************************************************
 use utils
 use maxdim
 use values
 implicit none
 INTEGER :: Y,INDEX(MINFO)
 INTEGER  ::        I, J, K1
 real(kind=dp) :: ACC3
 real(kind=dp) :: STRES(MSTR), PROP(MPROP), D(MSTR,MSTR)
 real(kind=dp) :: G, PHI, PSI, K, STATE(MAXSTA), C
 real(kind=dp) :: SIGX, SIGY, SIGZ, TAU,V
 real(kind=dp) :: I1, I2, I3, ALF, ALG, NU, ETA, ID2
 real(kind=dp) :: A11, A22, A33, Q, R, THETA, CS,CS1,CS2,CS3
 real(kind=dp) :: DFDI1, DFDI2, DFDI3, DGDI1, DGDI2, DGDI3, DFDID2, DGDID2
 real(kind=dp) :: DI1D1, DI1D2, DI1D3, DI1D4, DI2D1, DI2D2
 real(kind=dp) :: DI2D3, DI2D4, DI3D1, DI3D2, DI3D3, DI3D4
 real(kind=dp) :: DID2D1, DID2D2, DID2D3, DID2D4
 real(kind=dp) :: A(5), B(5),W, ATDB, ATD
 real(kind=dp) :: ABT(5,5), ABTD(5,5), DABTD(5,5)
 PARAMETER (ACC3 = 1D-8)
```

```
 REAL (KIND=DP) , PARAMETER :: ACC = 0.0001
  D        = 0.0_dp
 Y   = STATE(1)
 G   = PROP(1)
 K   = PROP(2)
 PHI = PROP(3)
 PSI = PROP(4)
 C   = PROP(5)
 if (PHI/=0.0) then
   V   = C/DTAN(PHI)
 end if
 SIGX = STRES(1)
 SIGY = STRES(2)
 SIGZ = STRES(3)
 TAU  = STRES(4)
 D(1,1) =  K+4D0*G/3D0
 D(1,2) =  K-2D0*G/3D0
 D(1,3) =  D(1,2)
 D(2,1) =  D(1,2)
 D(2,2) =  D(1,1)
 D(2,3) =  D(1,2)
 D(3,1) =  D(1,2)
 D(3,2) =  D(1,2)
 D(3,3) =  D(1,1)
 D(4,4) =  G

! Include jaumann terms if required (2d only at present)
 if(INDEX(indxIJAU) == 1)then
   !call status('value of indxIJAU',indxIJAU)
   D(1,5) =  TAU
   D(2,5) =  -TAU
   D(4,5) = (SIGY - SIGX)/2.0_dp
 end if
 IF (Y == 1) THEN
   I1  = SIGX + SIGY + SIGZ
   I2  = SIGX*SIGY + SIGX*SIGZ + SIGZ*SIGY - TAU*TAU
   I3  = SIGZ*(SIGX*SIGY - TAU*TAU)
   ID2 = I2-(I1*I1)/3
```

```
  NU  = DTAN(PHI)
  ETA = DTAN(PSI)
  ALF = 9.0 + 8.0*NU**2
  ALG = 9.0 + 8.0*ETA**2
  IF (ETA == 0.0) THEN
   DGDI1 = 2.0*I1
   DGDI2 = -3.0
   DGDI3 = 0.0
   DGDID2 = 0.0
  ELSE
        IF (ETA == NU) THEN
              CS = C
        ELSE                IF                (((SIGX-V)<ACC).AND.((SIGY-V)<ACC).AND.((SIGZ-
V)<ACC).AND.((TAU)<ACC)) THEN
             CS = ETA*V
        ELSE

                    A11 = -ETA*I1
              A22 = 3*ID2/4+ETA*ETA*I2
              A33 = -ETA*(9*I3-I1*I2)/8-ETA*ETA*ETA*I3
              Q   = (3.0*A22 - A11**2)/9.0
              R   = (9.0*A11*A22 - 27.0*A33 - 2.0*A11**3)/54.0
              IF (Q < 0.0) THEN
              IF(DABS(R/DSQRT(-Q*Q*Q))>1.0)THEN
                          call status('solving  cubic ec.,error',0)
                    WRITE(chanCAL,'(" ROUTINE MTYMD - ATTEMPT TO", " INVERT
COSINE OUT OF RANGE")')
                    WRITE(chanCAL,'(" ARGUMENT=",G20.10)')(R/DSQRT(-Q*Q*Q))
                    !PROP(5)=REAL(Y)
                    !PROP(6)=REAL(INDEX(1))
                    CALL ERRORS(STRES,STRES,PROP,D(1,1))
                    STOP
              END IF
                    THETA = DACOS(R/DSQRT(-Q*Q*Q))/3.0
                    CS1     = 2.0*DSQRT(-Q)*DCOS(THETA)- A11/3.0
                    CS2     = 2.0*DSQRT(-Q)*DCOS(THETA + PI*2.0/3.0)- A11/3.0
                    CS3     = 2.0*DSQRT(-Q)*DCOS(THETA + PI*4.0/3.0)- A11/3.0
                    CS=MAX(CS1,CS2,CS3)
```

```
                    IF (CS  < 0.0) THEN
                    WRITE(chanCAL,'("Error-Selecting root at MTYMD")' )
                    DO I=1,4
                        WRITE(chanCAL,'("PROP(",I1,")=",G15.8)')PROP(I)
                    enddo
                    DO I=1,4
                        WRITE(chanCAL,'(" STRES(",I1,") = ",G15.8)')I,(I)
                    enddo
                    CALL ERSTOP
                    END IF
                ELSE
                WRITE(chanCAL,'(" Error in Q at MTYMD ")' )
                CALL ERSTOP
                END IF
            END IF
            DGDI1  = -ETA*I2+8*CS*CS*ETA
            DGDI2  = -ETA*I1-8*CS*ETA*ETA
            DGDI3  = 9*ETA+8*ETA*ETA*ETA
            DGDID2 = -6*CS
        END IF
        DFDI1  = -NU*I2+8*NU*C*C
        DFDI2  = -NU*I1-8*C*NU*NU
        DFDI3  = ALF*NU
        DFDID2 = -6*C
        DI1D1  = 1.0
        DI1D2  = 1.0
        DI1D3  = 1.0
        DI1D4  = 0.0
        DI2D1  = SIGY + SIGZ
        DI2D2  = SIGX + SIGZ
        DI2D3  = SIGY + SIGX
        DI2D4  = -2.0 *TAU
        DI3D1  = SIGY*SIGZ
        DI3D2  = SIGX*SIGZ
        DI3D3  = SIGY*SIGX - TAU*TAU
        DI3D4  = -2.0*SIGZ*TAU
        DID2D1 = SIGY+SIGZ-2*I1/3
        DID2D2 = SIGX+SIGZ-2*I1/3
```

```
DID2D3 = SIGX+SIGY-2*I1/3
DID2D4 = -2*TAU
A(1) = DFDI1*DI1D1 + DFDI2*DI2D1 + DFDI3*DI3D1 + DFDID2*DID2D1
A(2) = DFDI1*DI1D2 + DFDI2*DI2D2 + DFDI3*DI3D2 + DFDID2*DID2D2
A(3) = DFDI1*DI1D3 + DFDI2*DI2D3 + DFDI3*DI3D3 + DFDID2*DID2D3
A(4) = DFDI1*DI1D4 + DFDI2*DI2D4 + DFDI3*DI3D4 + DFDID2*DID2D4
A(5) = 0.0
B(1) = DGDI1*DI1D1 + DGDI2*DI2D1 + DGDI3*DI3D1 + DGDID2*DID2D1
B(2) = DGDI1*DI1D2 + DGDI2*DI2D2 + DGDI3*DI3D2 + DGDID2*DID2D2
B(3) = DGDI1*DI1D3 + DGDI2*DI2D3 + DGDI3*DI3D3 + DGDID2*DID2D3
B(4) = DGDI1*DI1D4 + DGDI2*DI2D4 + DGDI3*DI3D4 + DGDID2*DID2D4
B(5) = 0.0
DO I=1, 5
  DO J=1, 5
     ABT(I,J) = B(I)*A(J)
  enddo
enddo
ATDB = 0.0
DO  I=1, 5
  ATD  = 0.0
  DO J=1, 5
    ATD = ATD + A(J)*D(J,I)
  enddo
ATDB = ATDB + ATD*B(I)
enddo
DO I=1, 5
  DO J=1, 5
    W = 0.0
    DO K1=1, 5
      W = W + ABT(I,K1)*D(K1,J)
    enddo
    ABTD(I,J) = W
  enddo
enddo
DO I=1, 5
  DO J=1, 5
    W = 0.0
    DO K1=1, 5
```

```
        W = W + D(I,K1)*ABTD(K1,J)
       enddo
       DABTD(I,J) = W
     enddo
   enddo
   IF (ABS(ATDB) > ACC3) THEN
     DO I=1, 5
       DO J=1, 5
         D(I,J) = D(I,J) - DABTD(I,J)/ATDB
       enddo
     enddo
   END IF
 END IF

END subroutine mtymd
```

G.2. CÓDIGO DE LAS SUBRUTINAS RELACIONADAS CON LA DEGRADACIÓN DEL TERRENO

```
subroutine DEGRADATION (JREG, JSTEP, TNSTEP)
!*********************************************************************
!    This subroutine calculates the forces due to the degradation
!    JREG: current stage
!    JSTEP: current step
!    TNSTEP: total number of steps in this stage
!    FINT contains the forces due to degradation
!
!    Created by A. da Costa   18/09/02
!*********************************************************************

 use utils
 use maxdim
 use values
 use var_alloc
 use var_restart
 implicit none
 integer :: IELE,JREG, JSTEP, TNSTEP, INOD, NENOD, IDIM, NDIM, IGAU,
NGAU,NDOFN,err,ELFAM
 integer :: NEREC,IB,NANDR,IDOF,GDOF, IFAIL,NUMSTEP, IOUT,ERROR
          ,ERR2,ERR1,material,ex ,i,j
 real(kind=dp) :: Fa, Fb, CI, AF , z
 real(kind=dp), dimension(2) ::P1,P2,P3, P4, P5, P6,a1,a2, G
 real(kind=dp), dimension(3) ::Q1, Q2, Q3
 real(kind=dp), dimension(2) ::GP1, GP2, GP3, GP4, GP5, GP6
 real(kind=dp), dimension(MEDIM,MENOD) :: OCOORD
 real(kind=dp), dimension(MDIM,MAXGAU) :: GCORD
 real(kind=dp), dimension(MDIM,MENOD) :: ECOORD
 real(kind=dp), dimension(MAXSTA) :: STATE
 real(kind=dp) :: GP1GP2, GP2GP4, GP4GP3, GP3GP1, GP3GP4, GP4GP6, GP6GP5, GP5GP3,
GP6GP3
 real(kind=dp) :: MGP1, MGP2, MGP3, MGP4, MGP5, MGP6,L, T,TANALFA,Fj,Fi,P1P5,INCC,FUN
 real(kind=dp) :: A12, A24, A43, A31, A34, A46, A65, A53, A63, ANG1, ANG2, ANG3, A, B, C, F1, F2
```

```
real(kind=dp), parameter :: ACC = 0.00001_dp
real(kind=dp), allocatable, dimension(:,:) :: F
logical, allocatable, dimension(:) :: PRESENT
logical :: check_tie
call status('Entering degradation', 0)

allocate(F(MAXTYP,MAXEL), PRESENT(MAXEL), stat = err)

P1(1) = DEGRAD(JREG,2)
P1(2) = DEGRAD(JREG,3)
P2(1) = DEGRAD(JREG,4)
P2(2) = DEGRAD(JREG,5)
a1(1) = DEGRAD(JREG,6)
a1(2) = DEGRAD(JREG,7)
a2(1) = DEGRAD(JREG,8)
a2(2) = DEGRAD(JREG,9)
Fa = DEGRAD(JREG,10)
Fb = DEGRAD(JREG,11)
CI = DEGRAD(JREG,12)
AF = DEGRAD(JREG,13)

!Calcutaion of points P3, P4, P5 and P6 to define the front advance and the regions
if (JSTEP == 1) then
  P3 = P1
  P4 = P2
  P5 = P1
  P6 = P2
end if
TANALFA = (Fb-Fa)/AF
Fj = Fa
Fi = Fb
l = 1.0_dp/REAL(TNSTEP)
P1P5 = DSQRT((P1(1)-P5(1))*(P1(1)-P5(1))+(P1(2)-P5(2))*(P1(2)-P5(2)))
if (P1P5<AF) then
  Fj=Fb-P1P5*TANALFA
else   !points P3 and P4 moving
  P3(1) = P3(1)+a1(1)*L
  P3(2) = P3(2)+a1(2)*L
```

```
  P4(1) = P4(1)+a2(1)*L
  P4(2) = P4(2)+a2(2)*L
 end if
 P5(1) = P5(1)+a1(1)*L
 P5(2) = P5(2)+a1(2)*L
 P6(1) = P6(1)+a2(1)*L
 P6(2) = P6(2)+a2(2)*L

 do IELE=1, NELE
  !Check if the element has been excavated
  ex = 0
  do i=2, (MEXCAV+1)
   do j=1, (JREG-1) !until stage before current
            if (IELE == EXCAV(i,j)) then
              ex=1
            end if
           end do
  end do
  if(ex==1) then
   !for excavated elements do nothing
   continue
  else
    !Calculation of OCOORD containing the original coordinates of the nodes of the current element
    NDIM = GINDEX(indxNDIM,IELE)
    NENOD = GINDEX(indxNENOD,IELE)
    do INOD=1, NENOD !for all nodes of the element
     do IDIM=1, NDIM  !for all dimensions
              if (GINDEX(indxICOORDUP,1) == 1) then !case updating coordenates
                       OCOORD(IDIM,INOD)              =              NODES(IELEM(INOD,IELE),IDIM)-
DELTA(NMAP(IELEM(INOD,IELE),IDIM+1))
              else   !case no updating coordenates
                       OCOORD(IDIM,INOD) = NODES(IELEM(INOD,IELE),IDIM)
              end if
     end do
    end do
    !Calculation of gauss points coordinates
    call GCOR (GINDEX(1,IELE), OCOORD, GCORD)
    NGAU = GINDEX(indxNGAU,IELE)
```

```
do IGAU=1, NGAU
        call check_material (GINDEX(1,IELE),material)
        if (material==1) then
          !material mat_ELAS_FRIC_COH_MATS
          G(1) = GCORD(1,IGAU)
          G(2) = GCORD(2,IGAU)
   !Vectors calcultaion
   GP1(1) = P1(1)-G(1)
   GP1(2) = P1(2)-G(2)
   GP2(1) = P2(1)-G(1)
   GP2(2) = P2(2)-G(2)
   GP3(1) = P3(1)-G(1)
   GP3(2) = P3(2)-G(2)
   GP4(1) = P4(1)-G(1)
   GP4(2) = P4(2)-G(2)
   GP5(1) = P5(1)-G(1)
   GP5(2) = P5(2)-G(2)
   GP6(1) = P6(1)-G(1)
   GP6(2) = P6(2)-G(2)

   !Calculation of escalar products
   GP1GP2 = GP1(1)*GP2(1)+GP1(2)*GP2(2)
   GP2GP4 = GP2(1)*GP4(1)+GP2(2)*GP4(2)
   GP4GP3 = GP4(1)*GP3(1)+GP4(2)*GP3(2)
   GP3GP1 = GP3(1)*GP1(1)+GP3(2)*GP1(2)
   GP3GP4 = GP3(1)*GP4(1)+GP3(2)*GP4(2)
   GP4GP6 = GP4(1)*GP6(1)+GP4(2)*GP6(2)
   GP6GP5 = GP6(1)*GP5(1)+GP6(2)*GP5(2)
   GP5GP3 = GP5(1)*GP3(1)+GP5(2)*GP3(2)
        GP6GP3 = GP6(1)*GP3(1)+GP6(2)*GP3(2)

   !Calculation of modulus
   MGP1 = DSQRT(GP1(1)*GP1(1)+GP1(2)*GP1(2))
   MGP2 = DSQRT(GP2(1)*GP2(1)+GP2(2)*GP2(2))
   MGP3 = DSQRT(GP3(1)*GP3(1)+GP3(2)*GP3(2))
   MGP4 = DSQRT(GP4(1)*GP4(1)+GP4(2)*GP4(2))
   MGP5 = DSQRT(GP5(1)*GP5(1)+GP5(2)*GP5(2))
   MGP6 = DSQRT(GP6(1)*GP6(1)+GP6(2)*GP6(2))
```

```
        !Calculation of angles
if ((P1(1) == P2(1)).and.(P1(2) == P2(2))) then
          A12 = 0.0_dp
        else
                z=      GP1GP2/(MGP1*MGP2)
                if ((z>1.0_dp).or.(z<-1.0_dp)) then
                  call status ('element number',IELE)
                end if
                if ((1.0_dp<z).and.(z<1.00005_dp)) then
                   z=1.0_dp
                end if
                if ((-1.00005_dp<z).and.(z<-1.0_dp)) then
                  z=-1.0_dp
                end if
          A12 = DACOS(z)
        end if
        if ((P2(1) == P4(1)).and.(P2(2) == P4(2)))  then
          A24 = 0.0_dp
        else
          z=   GP2GP4/(MGP2*MGP4)
                if ((z>1.0_dp).or.(z<-1.0_dp)) then
                  call status ('element number',IELE)
                end if
                if ((1.0_dp<z).and.(z<1.00005_dp)) then
                   z=1.0_dp
                end if
                if ((-1.00005_dp<z).and.(z<-1.0_dp)) then
                  z=-1.0_dp
                end if
                A24 = DACOS(z)
        end if
        if ((P4(1) == P3(1)).and.(P4(2) == P3(2)))  then
          A43 = 0.0_dp
        else
          z=   GP4GP3/(MGP4*MGP3)
                if ((z>1.0_dp).or.(z<-1.0_dp)) then
                  call status ('element number',IELE)
```

```
          end if
          if ((1.0_dp<z).and.(z<1.00005_dp)) then
            z=1.0_dp
          end if
          if ((-1.00005_dp<z).and.(z<-1.0_dp)) then
           z=-1.0_dp
          end if
          A43 = DACOS(z)
end if
if ((P3(1) == P1(1)).and.(P3(2) == P1(2))) then
  A31 = 0.0_dp
else
  z=   GP3GP1/(MGP3*MGP1)
          if ((z>1.0_dp).or.(z<-1.0_dp)) then
           call status ('element number',IELE)
          end if
          if ((1.0_dp<z).and.(z<1.00005_dp)) then
            z=1.0_dp
          end if
          if ((-1.00005_dp<z).and.(z<-1.0_dp)) then
           z=-1.0_dp
          end if
          A31 = DACOS(z)
end if
if ((P3(1) == P4(1)).and.(P3(2) == P4(2)))  then
  A34=0.0_dp
else
  z=   GP3GP4/(MGP3*MGP4)
          if ((z>1.0_dp).or.(z<-1.0_dp)) then
           call status ('element number',IELE)
          end if
          if ((1.0_dp<z).and.(z<1.00005_dp)) then
            z=1.0_dp
          end if
          if ((-1.00005_dp<z).and.(z<-1.0_dp)) then
           z=-1.0_dp
          end if
          A34 = DACOS(z)
```

```
end if
if((P4(1) == P6(1)).and.(P4(2) == P6(2)))  then
  A46 = 0.0_dp
else
  z=   GP4GP6/(MGP4*MGP6)
          if ((z>1.0_dp).or.(z<-1.0_dp)) then
            call status ('element number',IELE)
          end if
          if ((1.0_dp<z).and.(z<1.00005_dp)) then
             z=1.0_dp
          end if
          if ((-1.00005_dp<z).and.(z<-1.0_dp)) then
            z=-1.0_dp
          end if
          A46 = DACOS(z)
end if
if ((P6(1) == P5(1)).and.(P6(2) == P5(2)))  then
  A65 = 0.0_dp
else
  z=   GP6GP5/(MGP6*MGP5)
          if ((z>1.0_dp).or.(z<-1.0_dp)) then
            call status ('element number',IELE)
          end if
          if ((1.0_dp<z).and.(z<1.00005_dp)) then
             z=1.0_dp
          end if
          if ((-1.00005_dp<z).and.(z<-1.0_dp)) then
            z=-1.0_dp
          end if
          A65 = DACOS(z)
end if
if ((P5(1) == P3(1)).and.(P5(2) == P3(2)))  then
  A53 = 0.0_dp
else
  z=   GP5GP3/(MGP5*MGP3)
          if ((z>1.0_dp).or.(z<-1.0_dp)) then
            call status ('element number',IELE)
          end if
```

```
                if ((1.0_dp<z).and.(z<1.00005_dp)) then
                  z=1.0_dp
                end if
                if ((-1.00005_dp<z).and.(z<-1.0_dp)) then
                  z=-1.0_dp
                end if
                A53 = DACOS(z)
        end if

!Check in wich zone is the gauss point
ANG1 = (A12+A24+A43+A31)/RAD
ANG2 = (A34+A46+A65+A53)/RAD
if (DABS(360.0_dp-ANG1)<ACC) then
  !Gauss point inside zone 1 so F2 equals to Fj
  F2 = Fj
  GPROP(6,IGAU,IELE) = GPROP(7,IGAU,IELE)
  GPROP(7,IGAU,IELE) = F2
  GPROP(5,IGAU,IELE) = CI*F2
        !If the point goes out of the new yield surface, then change ISTATE
        call YOUT(GINDEX(1,IELE),GSTRES(1,IGAU,IELE),GPROP(1,IGAU,IELE),IOUT)
        GSTATE(1,IGAU,IELE) = IOUT
else if (DABS(360.0_dp-ANG2)<ACC) then

  !Gauss point inside zone 2 so F2 between Fa and Fb, interpolation necesary
  GPROP(6,IGAU,IELE) = GPROP(7,IGAU,IELE)
  !Check inside which triangle is the gauus point
  if ((P6(1) == P3(1)).and.(P6(2) == P3(2)))  then
          A63 = 0.0_dp
        else
          z= GP6GP3/(MGP6*MGP3)
                if ((z>1.0_dp).or.(z<-1.0_dp)) then
                  call status ('element number',IELE)
                end if
                if ((1.0_dp<z).and.(z<1.00005_dp)) then
                  z=1.0_dp
                end if
                if ((-1.00005_dp<z).and.(z<-1.0_dp)) then
                  z=-1.0_dp
```

```
                end if
                    A63 = DACOS(z)
        end if
ANG3 = (A34+A46+A63)/RAD
if (DABS(360.0_dp-ANG3)<ACC) then
  !Gauss point inside triangle 1
  !Interpolation between P3, P4, P6
  Q1(1) = P3(1)
  Q1(2) = P3(2)
  Q1(3) = Fj
  Q2(1) = P4(1)
  Q2(2) = P4(2)
  Q2(3) = Fj
  Q3(1) = P6(1)
  Q3(2) = P6(2)
  Q3(3) = Fi
else
  !Gauss point inside triangle 2
  !Interpolation between P3, P6, P5
  Q1(1) = P3(1)
  Q1(2) = P3(2)
  Q1(3) = Fj
  Q2(1) = P5(1)
  Q2(2) = P5(2)
  Q2(3) = Fi
  Q3(1) = P6(1)
  Q3(2) = P6(2)
  Q3(3) = Fi
end if
!Interpolation
A = ((Q2(2)-Q1(2))*(Q3(3)-Q1(3))-(Q2(3)-Q1(3))*(Q3(2)-Q1(2)))
B = ((Q2(3)-Q1(3))*(Q3(1)-Q1(1))-(Q2(1)-Q1(1))*(Q3(3)-Q1(3)))
C = ((Q2(1)-Q1(1))*(Q3(2)-Q1(2))-(Q2(2)-Q1(2))*(Q3(1)-Q1(1)))
F2 = (A*(Q1(1)-G(1))+B*(Q1(2)-G(2)))/C+Q1(3)
GPROP(7,IGAU,IELE) = F2
GPROP(5,IGAU,IELE) = CI*F2
      !If the point goes out of the new yield surface, then change ISTATE
      call YOUT(GINDEX(1,IELE),GSTRES(1,IGAU,IELE),GPROP(1,IGAU,IELE),IOUT)
```

```
            GSTATE(1,IGAU,IELE) = IOUT
        end if
        call YOUT (GINDEX(1,IELE), GSTRES(1,IGAU,IELE), GPROP(1,IGAU,IELE), IOUT)
            if ((GPROP(6,IGAU,IELE) /= GPROP(7,IGAU,IELE)).and.(IOUT == 1)) then !add the
increment of stresses due to degrad
        !degradation at this gauss point

        !Calculation of the equivalent forces
            INCC = (GPROP(7,IGAU,IELE)-GPROP(6,IGAU,IELE))*CI
        T = 0.0_dp
        NUMSTEP = 100
            call STRESDEGRA (GINDEX(1,IELE), INCC, GPROP(1,IGAU,IELE), T,
GSTATE(1,IGAU,IELE), TOL, &
            & GSTRES(1,IGAU,IELE), IFAIL)
        if (IFAIL == 1) then

                call FIXSTRESDEGRA (GINDEX(1,IELE), INCC, GPROP(1,IGAU,IELE), T,
GSTATE(1,IGAU,IELE), NUMSTEP, &
            & GSTRES(1,IGAU,IELE), IFAIL)
        end if
            call MATCHKM2(GSTRES(1,IGAU,IELE),GPROP(1,IGAU,IELE),ERROR)
        if(ERROR == 1)then
!
!       ONE OR MORE PRINCIPAL STRESSES HAVE BECOME TOO POSITIVE, update the stresses
to the vertix
!
          call ORIGSTRS (GSTRES(1,IGAU,IELE),GPROP(1,IGAU,IELE))
        end if
            call YFUNM(GSTRES(1,IGAU,IELE),GPROP(1,IGAU,IELE),FUN)
        call MATCHKM1(GSTRES(1,IGAU,IELE),GPROP(1,IGAU,IELE),ERR1)
            if( (DABS(FUN) >ACC).or.(ERR1==1) )then
         ! Unacceptable output stresses from stresdegra/fixstresdegra
         ! correct stresses to the yield surface
            call                                                    CORRECTMC
(GINDEX(1,IELE),GPROP(1,IGAU,IELE),GSTRES(1,IGAU,IELE),GSTRES(1,IGAU,IELE))
          call YFUNM(GSTRES(1,IGAU,IELE),GPROP(1,IGAU,IELE),FUN)
                if( DABS(FUN) >ACC )then
                 call status('Subroutine CORRECTMC2: ERROR IN CORRECTED STRESSES',0)
```

```
            call ERTEXT('Subroutine CORRECTMC2: ERROR IN CORRECTED STRESSES')
            call ERTEXT('ERROR FUN:')
            call ERREAL(FUN)
            call ERSTOP
           end if

           call MATCHKM2(GSTRES(1,IGAU,IELE),GPROP(1,IGAU,IELE),ERR2)
           if(ERR2 ==1)then
            call status('Subroutine CORRECTMC2.(b) Error in corrected stresses, down the vertix',0)
            call ERTEXT('Subroutine CORRECTMC2: UNACCEPTABLE STRESSES')
            call ERSTOP
           end if
                end if
        end if
             end if
            end do     !gauss point
    !Calculation of the forces equivalents to the increment of stresses
    call ELCOD (IELEM(1,IELE), GINDEX(1,IELE), NODES, ECOORD)
    call check_alloc(err)
    call FORCE (GINDEX(1,IELE), ECOORD, GSTRES(1,1,IELE), F(1,IELE), IELEM(1,IELE), &
          & NODES, RT)
 end if
end do     !elements
FINT = 0.0_dp
do IELE = 1,NELE
   PRESENT(ABS(LISTEL(IELE))) = (LISTEL(IELE) > 0)
end do
collect_forces: do IELE = 1, NELE
   if(PRESENT(IELE))then
     NENOD  = GINDEX(indxNENOD,IELE)
     NDOFN  = GINDEX(indxNDOFN,IELE)
     ELFAM  = GINDEX(indxELFAM,IELE)
     NEREC  = GINDEX(indxNEREC,IELE)
     IB = 0
     if (check_tie(ELFAM)) then
       NANDR = NENOD + NEREC
     else
       NANDR = NENOD
```

```
      end if
      do INOD = 1,NANDR
       if (check_tie(ELFAM)) then
         NDOFN = NMAP(IELEM(INOD,IELE),1)
       end if
       do IDOF = 1, NDOFN
         GDOF      = NMAP(IELEM(INOD, IELE), IDOF + 1)
         FINT(GDOF) = FINT(GDOF) + F(IB + IDOF, IELE)
       end do
       IB = IB + NDOFN
      end do
    end if
 end do collect_forces
 deallocate(F, PRESENT, stat = err)
 call check_dealloc(err)
 call status('Leaving degradation',0)
 return
end subroutine DEGRADATION
```

```
subroutine STRESDEGRA(INDEX, INCC, PROP, T, STATE, TOL, SIG, IFAIL)
!*****************************************************************************
!    This subroutine updates the stresses SIG due to the variation of cohesion
!    because degradation. T is the time corresponding to the start
!    of the calculation.
!    SIG contains the stresses at time T on entry to the subroutine
!    and contains the stresses at time 1 on exit.
!    If the number of iterations exceeds a set limit( MAXCNT )
!    Then the subroutine returns to the driving program with
!    the original stresses and time. In this case IFAIL is 1
!    on exit, otherwise IFAIL is 0.
!
!   Created by A. da Costa   18/09/02
!*****************************************************************************

 use utils
 use maxdim
 use values
 implicit none
 integer :: I, EXIT, COUNT1, IFAIL,K,NSTR
 integer, parameter :: MAXCNT = 300
 integer, dimension(MINFO) :: INDEX
 real(kind=dp), dimension(MSTR) :: SIG, DS, DS1, DS2, STRS
 real(kind=dp), dimension(MSTR) :: DS3, DS4, DS5, DS6, RES, DSIG, SIG0
 real(kind=dp), dimension(MAXSTA) :: STATE
 real(kind=dp), dimension(MPROP) :: PROP
 real(kind=dp) :: TOL,DEN,NUM, T, T0, DT, RATIO, Q,INCC, CP,CC
 real(kind=dp), parameter :: LIMACC = 1.0e-10_dp

 NSTR = INDEX(indxNSTR)
 IFAIL = 0
 T0 = T
 DS = 0.0_dp
 SIG0(1:NSTR) = SIG(1:NSTR)
 if (T < 0.0_dp .or. T > 1.0_dp) then
  call status_real(' Subroutine STRESDEGRA: T out of bounds',T)
  call erstop
 end if
```

```
!   SET UP INITIAL INCREMENT SIZE
  DT = 1.0_dp - T
  CP = INCC * DT
  if ((T + DT) > 1.0_dp) then
    DT = 1.0_dp-T
    EXIT = 1
  else
    EXIT = 0
  end if
  COUNT1 = 0
  CC = PROP(5)*PROP(6)/PROP(7) !cohesion at the start of the integration
  mainloop : do
    COUNT1 = COUNT1 + 1
    if (COUNT1 > MAXCNT) then
      IFAIL = 1
      SIG(1:NSTR) = SIG0(1:NSTR)
      T = T0
      exit mainloop
    end if
!   4TH ORDER RUNGE-KUTTA INTEGRATION SCHEME WITH ERROR CORRECTION
    CP = INCC * DT
          CC = CC + CP/2.0_dp  !current cohesion
    DS(1:NSTR) = SIG(1:NSTR)
    call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
    DO K=1,5
      DS1(K) = STRS (K)*CP
    END DO
    DS(1:NSTR) = SIG(1:NSTR) + DS1(1:NSTR) * 0.5_dp
    call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
    DO K=1,5
      DS2(K) = STRS (K)*CP
    END DO
    DS(1:NSTR) = SIG(1:NSTR) + (DS1(1:NSTR) + DS2(1:NSTR)) * 0.25_dp
    call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
    DO K=1,5
      DS3(K) = STRS (K)*CP
    END DO
    DS(1:NSTR) = SIG(1:NSTR) + DS3(1:NSTR) * 2.0_dp     - DS2(1:NSTR)
```

```
call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
DO K=1,5
  DS4(K) = STRS (K)*CP
END DO
do I = 1,NSTR
  DS(I) = SIG(I)+(7.0_dp*DS1(I)+10.0_dp*DS2(I)+DS4(I))/27.0_dp
end do
call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
DO K=1,5
  DS5(K) = STRS (K)*CP
END DO
do I = 1,NSTR
  DS(I) = DS1(I)*28.0_dp-DS2(I)*125.0_dp+DS3(I)*546.0_dp
  DS(I) = SIG(I)+(DS(I)+DS4(I)*54.0_dp-DS5(I)*378.0_dp)/625.0_dp
end do
call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
DO K=1,5
  DS6(K) = STRS (K)*CP
END DO
do I = 1,NSTR
  DSIG(I) = (DS1(I)+4.0_dp*DS3(I)+DS4(I))/6.0_dp
end do
do I = 1,NSTR
  RES(I) = -42.0_dp*DS1(I)-224.0_dp*DS3(I)-21.0_dp*DS4(I)+162.0_dp*DS5(I)
  RES(I) = (RES(I)+125.0_dp*DS6(I))/336.0_dp
end do
NUM = 0
DEN = 0
do I = 1,NSTR
  NUM = NUM + RES(I) * RES(I)
  DEN = DEN + DSIG(I) * DSIG(I)
end do
if (SQRT(DEN) == 0.0_dp) then
  RATIO = TOL * 0.8_dp / 1.4_dp
else
  RATIO = SQRT(NUM) / SQRT(DEN)
end if
if (RATIO <= TOL) then
```

```
    T = T + DT
    SIG(1:NSTR) = SIG(1:NSTR) + DSIG(1:NSTR)
    if (EXIT == 1) then
      exit mainloop
    end if
    if (ABS(RATIO) < LIMACC) then
      Q = 5.0_dp
    else
      Q = 0.8_dp * TOL / RATIO
      if (Q > 1.4_dp) then
        Q = 1.4_dp
      end if
      if (Q < 1.0_dp) then
        Q = 1.0_dp
      end if
    end if
  else
    Q = 0.8_dp * TOL / RATIO
    if (Q < 0.1_dp) then
      Q = 0.1_dp
    end if
  end if
  DT = DT * Q
  if ((T + DT) > 1.0_dp) then
    DT = 1.0_dp - T
    EXIT = 1
  else
    EXIT = 0
  end if
 end do mainloop
end subroutine STRESDEGRA
```

```
subroutine FIXSTRESDEGRA (INDEX, INCC, PROP, T, STATE, NUMSTEP, SIG, IFAIL)
!**********************************************************************
! Update the stresses SIG  due to the variation of cohesion
! because degradation.
! This subroutine is based on a Fifth order Runge-Kutta approach with fixed
! time steps.
! T is the time corresponding to the start of the calculation.
! SIG contains the stresses at time T on entry to the
! subroutine and contains the stresses at time 1 on exit.
!
! Created by A. da Costa 19/09/02
!**********************************************************************
  use utils
  use maxdim
  use values
  implicit none

  integer ::  K, INDEX(MINFO),NSTR
  integer :: IFAIL,ISTEP,NUMSTEP
  real(kind=dp) :: CP,SIG(MSTR),T,INCC,CC
  real(kind=dp) :: PROP(MPROP)
  real(kind=dp) :: DS(MSTR), DS1(MSTR), DS2(MSTR), DS3(MSTR), STRS(MSTR)
  real(kind=dp) :: DS4(MSTR), DS5(MSTR), DS6(MSTR)
  real(kind=dp) :: DT, DSIG(MSTR), STATE(MAXSTA)

  NSTR = INDEX(indxNSTR)

 if (T < 0.0_dp .or. T > 1.0_dp) then
   call status_real('FIXSTRESDEGRA: Value of T out of range: ',T)
   call ERTEXT('FIXSTRESDEGRA: T out of range')
   call ERDBLE(T)
   call ERSTOP
  else if (T == 1.0_dp) then
    return
  end if
  DT = (1.0_dp - T) / REAL(NUMSTEP)
!
! 5TH ORDER RUNGE-KUTTA INTEGRATION SCHEME WITH FIXED TIME STEPS.
```

```
!
  CC = PROP(5)*PROP(6)/PROP(7)  !cohesion at the start of the integration
  CP = DT*INCC
  do ISTEP = 1,NUMSTEP
    CC = CC + CP/2          !current cohesion
           DS(1:NSTR) = SIG(1:NSTR)
    call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
    DO K=1,5
      DS1(K) = STRS (K)*CP
    END DO
    DS(1:NSTR) = SIG(1:NSTR) + 0.5_dp*DS1(1:NSTR)
    call STRSDEGRACAL (INDEX, PROP,CC, DS, STRS)
    DO K=1,5
      DS2(K) = STRS (K)*CP
    END DO
    DS(1:NSTR) = SIG(1:NSTR) + 0.25_dp*(DS1(1:NSTR) + DS2(1:NSTR))
    call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
    DO K=1,5
      DS3(K) = STRS (K)*CP
    END DO
    DS(1:NSTR) = SIG(1:NSTR) - DS2(1:NSTR) + 2.0_dp*DS3(1:NSTR)
    call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
    DO K=1,5
      DS4(K) = STRS (K)*CP
    END DO
    DS(1:NSTR) = SIG(1:NSTR) + (7.0_dp*DS1(1:NSTR) + 10.0_dp*DS2(1:NSTR) +   &
    &                    DS4(1:NSTR)) / 27.0_dp
    call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
    DO K=1,5
      DS5(K) = STRS (K)*CP
    END DO
    DS(1:NSTR) = SIG(1:NSTR) + (DS1(1:NSTR)*28.0_dp - DS2(1:NSTR)*125.0_dp + &
    &                    DS3(1:NSTR)*546.0_dp + DS4(1:NSTR)*54.0_dp - &
    &                    DS5(1:NSTR)*378.0_dp) / 625.0_dp
    call STRSDEGRACAL (INDEX, PROP, CC, DS, STRS)
    DO K=1,5
      DS6(K) = STRS (K)*CP
    END DO
```

```
    DSIG(1:NSTR) = (14.0_dp*DS1(1:NSTR) + 35.0_dp*DS4(1:NSTR) +          &
    &          162.0_dp*DS5(1:NSTR) + 125.0_dp*DS6(1:NSTR)) / 336.0_dp
    SIG(1:NSTR) = SIG(1:NSTR) + DSIG(1:NSTR)
  end do
  return
end subroutine FIXSTRESDEGRA
```

```
subroutine STRSDEGRACAL (INDEX,PROP,CC,STRES,SIG)
!**********************************************************************
!      This subroutine calculates the first factor in the equation of
!      stresses increment due to degradation, This factor is a vector
!      STRES contains the stresses at the considered point
!      SIG contains the first factor (vector) in the equation of
!      stresses increment due to degradation
!
!           Created by A. DA COSTA 20/09/02 for soil degradation
!**********************************************************************
 use maxdim
 use values
 use utils
 implicit none
 real (kind=dp) :: STRES(MSTR),PROP(MPROP),SIG(MSTR),CC
 integer :: INDEX(MINFO) , I, J
 real (kind=dp) :: G,K,PHI,PSI,C,SIGX, SIGY, SIGZ, TAU ,z
 real (kind=dp) :: CS,CS1,CS2,CS3, NU, ETA, ALF, ALG,I1, I2, I3,ID2
 real (kind=dp) :: DFDI1, DFDI2, DFDI3, DGDI1, DGDI2, DGDI3, DFDID2, DGDID2
 real (kind=dp) :: DI1D1, DI1D2, DI1D3, DI1D4, DI2D1, DI2D2
 real (kind=dp) :: DI2D3, DI2D4, DI3D1, DI3D2, DI3D3, DI3D4
 real (kind=dp) :: DID2D1, DID2D2, DID2D3, DID2D4,DFDC
 real (kind=dp) :: D(5,5),A(5), B(5), ATDB, ATD, DDG(5),DDG2(5)
 real (kind=dp) :: A11, A22, A33, Q, R, THETA

 G   = PROP(1)
 K   = PROP(2)
 PHI = PROP(3)
 PSI = PROP(4)
 C   = CC

 SIGX = STRES(1)
 SIGY = STRES(2)
 SIGZ = STRES(3)
 TAU  = STRES(4)

 DO I=1,5
   DO J=1,5
```

```
    D(I,J)=0.0_dp
  ENDDO
ENDDO
!*****************************************
!CALCULATION OF THE ELASTIC STIFFNESS MATRIX
D(1,1) =  K+4.0_dp*G/3.0_dp
D(1,2) =  K-2.0_dp*G/3.0_dp
D(1,3) =  D(1,2)
D(2,1) =  D(1,2)
D(2,2) =  D(1,1)
D(2,3) =  D(1,2)
D(3,1) =  D(1,2)
D(3,2) =  D(1,2)
D(3,3) =  D(1,1)
D(4,4) =  G

!Include Jauman terms if required (2D only)
If (INDEX(indxIJAU) == 1) then
  D(1,5)=TAU
  D(2,5)=-TAU
  D(4,5)=(SIGY-SIGX)/2.0_dp
end if
!*********************************************
!CALCULATION OF THE  INVARIANTS
I1  = SIGX + SIGY + SIGZ
I2  = SIGX*SIGY + SIGX*SIGZ + SIGZ*SIGY - TAU*TAU
I3  = SIGZ*(SIGX*SIGY - TAU*TAU)
ID2 = I2-(I1*I1)/3.0_dp

NU  = DTAN(PHI)
ETA = DTAN(PSI)
ALF = 9.0_dp + 8.0_dp*NU**2
ALG = 9.0_dp + 8.0_dp*ETA**2
!*****************************************
!CALCULATION OF DGDI
IF (ETA == 0.0_dp) THEN
   DGDI1 = 2.0_dp*I1
   DGDI2 = -3.0_dp
```

```
   DGDI3 = 0.0_dp
   DGDID2 = 0.0_dp
 ELSE
 IF (ETA == NU) THEN
   CS = C
 ELSE
                A11 = -ETA*I1
                A22 = 3.0_dp*ID2/4+ETA*ETA*I2
                A33 = -ETA*(9.0_dp *I3-I1*I2)/8.0_dp -ETA*ETA*ETA*I3
                Q   = (3.0_dp *A22 - A11**2)/9.0_dp
                R   = (9.0_dp *A11*A22 - 27.0_dp *A33 - 2.0_dp *A11**3)/54.0_dp
                IF (Q < 0.0_dp) THEN
                IF(DABS(R/DSQRT(-Q*Q*Q))>1.0_dp)THEN
                        WRITE(chanCAL,'(" ROUTINE ISTRESS- ATTEMPT TO", " INVERT
COSINE OUT OF RANGE")')
                        WRITE(chanCAL,'(" ARGUMENT=",G20.10)')(R/DSQRT(-Q*Q*Q))
                        CALL ERRORS(STRES,STRES,PROP,D(1,1))
                        STOP
                END IF
                        z= R/DSQRT(-Q*Q*Q)
                                if ((z>1.0).or.(z<-1.0)) then
                                  call status_real ('value costheta ec cubic',z)
                                end if
                                THETA = DACOS(R/DSQRT(-Q*Q*Q))/3.0_dp
                        CS1     = 2.0_dp *DSQRT(-Q)*DCOS(THETA)- A11/3.0_dp
                        CS2          = 2.0_dp  *DSQRT(-Q)*DCOS(THETA  +  PI*2.0_dp  /3.0_dp)-
A11/3.0_dp
                        CS3          = 2.0_dp  *DSQRT(-Q)*DCOS(THETA  +  PI*4.0_dp  /3.0_dp)-
A11/3.0_dp
                        CS=MAX(CS1,CS2,CS3)
                        IF (CS  < 0.0_dp) THEN
                        WRITE(chanCAL,'("Error-Selecting root at ISTRESS")' )
                        DO I=1,4
                                WRITE(chanCAL,'("PROP(",I1,")=",G15.8)')PROP(I)
                        enddo
                        DO I=1,4
                                WRITE(chanCAL,'(" STRES(",I1,") = ",G15.8)')I,(I)
                        enddo
```

```
                    CALL ERSTOP
                    END IF
                ELSE
                WRITE(chanCAL,'(" Error in Q at ISTRESS")' )
                CALL ERSTOP
                END IF
        END IF
        DGDI1  = -ETA*I2+8.0_dp *CS*CS*ETA
        DGDI2  = -ETA*I1-8.0_dp *CS*ETA*ETA
        DGDI3  = 9.0_dp *ETA+8.0_dp *ETA*ETA*ETA
        DGDID2 = -6.0_dp *CS
END IF
!*************************************
!CALCULATION OF DFDI
DFDI1  = -NU*I2+8.0_dp *NU*C*C
DFDI2  = -NU*I1-8.0_dp *C*NU*NU
DFDI3  = ALF*NU
DFDID2 = -6.0_dp *C
!*************************************
!CALCULATION OF DIDSIG
DI1D1  = 1.0_dp
DI1D2  = 1.0_dp
DI1D3  = 1.0_dp
DI1D4  = 0.0_dp
DI2D1  = SIGY + SIGZ
DI2D2  = SIGX + SIGZ
DI2D3  = SIGY + SIGX
DI2D4  = -2.0_dp *TAU
DI3D1  = SIGY*SIGZ
DI3D2  = SIGX*SIGZ
DI3D3  = SIGY*SIGX - TAU*TAU
DI3D4  = -2.0_dp *SIGZ*TAU
DID2D1 = SIGY+SIGZ-2.0_dp *I1/3.0_dp
DID2D2 = SIGX+SIGZ-2.0_dp *I1/3.0_dp
DID2D3 = SIGX+SIGY-2.0_dp *I1/3.0_dp
DID2D4 = -2.0_dp *TAU
!*************************************
!CALCULATION OF DFDSIG
```

```
A(1) = DFDI1*DI1D1 + DFDI2*DI2D1 + DFDI3*DI3D1 + DFDID2*DID2D1
A(2) = DFDI1*DI1D2 + DFDI2*DI2D2 + DFDI3*DI3D2 + DFDID2*DID2D2
A(3) = DFDI1*DI1D3 + DFDI2*DI2D3 + DFDI3*DI3D3 + DFDID2*DID2D3
A(4) = DFDI1*DI1D4 + DFDI2*DI2D4 + DFDI3*DI3D4 + DFDID2*DID2D4
A(5) = 0.0_dp
!**************************************
!CALCULATION OF DGDSIG
B(1) = DGDI1*DI1D1 + DGDI2*DI2D1 + DGDI3*DI3D1 + DGDID2*DID2D1
B(2) = DGDI1*DI1D2 + DGDI2*DI2D2 + DGDI3*DI3D2 + DGDID2*DID2D2
B(3) = DGDI1*DI1D3 + DGDI2*DI2D3 + DGDI3*DI3D3 + DGDID2*DID2D3
B(4) = DGDI1*DI1D4 + DGDI2*DI2D4 + DGDI3*DI3D4 + DGDID2*DID2D4
B(5) = 0.0_dp
!**************************************
!CALCULATION OF DFDISG*D          AND   DFDSIG*D*DGDSIG
ATDB = 0.0_dp
DO  I=1, 5
  ATD  = 0.0_dp
  DO J=1, 5
    ATD = ATD + A(J)*D(J,I)
  enddo
  ATDB = ATDB + ATD*B(I)
enddo
!**************************************
!CALCULATION OF D*DGDSIG
DO I=1,5
   DDG(I)=0.0_dp
ENDDO
DO I=1,5
   DO J=1,5
      DDG(I)=DDG(I)+D(I,J)*B(J)
   ENDDO
ENDDO
!**************************************
!CALCULATION OF DFDC
DFDC=-6.0_dp *ID2+8.0_dp *(-3.0_dp *C*C+2.0_dp *C*NU*I1-NU*NU*I2)
DO I=1,5
   DDG2(I)=DDG(I)*DFDC
ENDDO
```

```
  !*************************************
  !CALCULATION OF VECTOR SIG
  DO I=1,5
    SIG(I) = 0.0_dp
  ENDDO
  DO I=1,5
    SIG(I) = -DDG2(I)/ATDB
  ENDDO
END SUBROUTINE STRSDEGRACAL
```

```
subroutine ORIGSTRS (STRES,PROP)
!*********************************************************
! This subroutine update the stresses values to the yield funtion vertex
!
!  Created by A. DA COSTA 20/09/02 for soil degradation
!*********************************************************************

 use utils
 use maxdim
 use values
 use var_alloc
 use var_restart
 implicit none
 real(kind=dp) :: C,PHI
 real(kind=dp), dimension(MSTR) ::STRES
 real(kind=dp), dimension(MPROP) ::PROP

 C = PROP(5)
 PHI = PROP(3)

 STRES(1) = C/DTAN(PHI)
 STRES(2) = C/DTAN(PHI)
 STRES(3) = C/DTAN(PHI)
 STRES(4) = 0.0_dp
end subroutine ORIGSTRS
```

```
subroutine check_material (INDEX,material)
!*****************************************************************
!     This subroutine check that the material is possible
!     to be degradated
!
!  Created by A. DA COSTA 20/09/02 for soil degradation
!*****************************************************************

 use utils
 use maxdim
 use values
 use var_alloc
 use var_restart
 implicit none

 integer :: INDEX(MINFO),MP,material

 MP=INDEX(indxMP)
 if (MP==matELAS_FRIC_COH_MATS) then
   material=1
 else
   material=0
 end if
end subroutine
```