FACULTAD DE
INGENIERIA
Y ARQUITECTURA

MECANICA DE SOLIDOS

VI Tarea

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MANIZALES

Estudio de las tensiones en un punto.

Círculo de Mohr en tres dimensiones

Docente: Diego Andrés Álvarez Marín
Pablo César Teixeira Hoyos. *pcteixeirah@unal.edu.co*
CC. 1125348583

---

6) 2.9.6 (círculo de Mohr en 3D)

* Estudiar los videos adjuntos.

* Estudiar y subir a esta plataforma los apuntes de los videos y de la lectura de la sección 2.9.6 del main.pdf

---

2.9.6. Círculo de Mohr en tres dimensiones

En el caso de un estado de esfuerzos tridimensional, también es posible hacer una representación gráfica de Mohr. Esta representación la trataremos en la presente sección.

$$\sigma_n = \sigma_x\alpha^2 + \sigma_y\beta^2 + \sigma_z\gamma^2 + 2\tau_{xy}\alpha\beta + 2\tau_{yz}\beta\gamma + 2\tau_{zx}\gamma\alpha$$

Eq. 2.24

Para una matriz de tensiones $\underline{\underline{\sigma}}$ dada y para un plano descrito por un vector normal $\hat{\boldsymbol{n}}$, podemos calcular las longitudes de la componente normal σ_n y de la componente tangencial τ_n de $\boldsymbol{q}$ utilizando las ecuaciones (2.24) y (2.26) respectivamente.

Supongamos que queremos trabajar en $\mathbb{R}^3$, no obstante, en vez de trabajar en el espacio que tiene como base los vectores $\hat{\boldsymbol{i}}, \hat{\boldsymbol{j}}, \hat{\boldsymbol{k}}$, trabajaremos en uno que tiene como base los vectores propios correspondientes a las direcciones principales[29] $\hat{\boldsymbol{n}}_1, \hat{\boldsymbol{n}}_2, \hat{\boldsymbol{n}}_3$ (he aquí la importancia del porqué estos vectores deben satisfacer el producto cruz $\hat{\boldsymbol{n}}_1 \times \hat{\boldsymbol{n}}_2 = \hat{\boldsymbol{n}}_3$, puesto que deben formar un sistema de coordenadas de la mano derecha). Aquí hemos supuesto que los valores propios respectivos están ordenados de modo tal que $\sigma_1 > \sigma_2 > \sigma_3$.

Base Ortogonal

Calcularemos entonces la magnitud de los esfuerzos normales σ_n y tangenciales τ_n asociados haciendo en las ecuaciones (2.24) y (2.26) $\sigma_x = \sigma_1$, $\sigma_y = \sigma_2$, $\sigma_z = \sigma_3$, $\tau_{xy} = \tau_{yz} = \tau_{xz} = 0$; por lo tanto,

$$\boxed{\begin{aligned} \sigma_n &= \sigma_1\alpha^2 + \sigma_2\beta^2 + \sigma_3\gamma^2 \qquad (2.65) \\ \tau_n^2 &= (\sigma_1\alpha)^2 + (\sigma_2\beta)^2 + (\sigma_3\gamma)^2 - \sigma_n^2 \qquad (2.66) \end{aligned}}$$

[29] En este caso, decimos que el material está sometido a un estado de *esfuerzos triaxiales*, ya que no existe una fuerza cortante aplicada.

Eq. 2.26

$$|\tau_n| = \left[(\sigma_x\alpha + \tau_{xy}\beta + \tau_{zx}\gamma)^2 + (\tau_{xy}\alpha + \sigma_y\beta + \tau_{yz}\gamma)^2 + (\tau_{zx}\alpha + \tau_{yz}\beta + \sigma_z\gamma)^2 - \sigma_n^2\right]^{1/2}$$

A partir de las ecuaciones (2.65) y (2.66), teniendo en cuenta que

$$\alpha^2+\beta^2+\gamma^2=1, \tag{2.67}$$

y utilizando el siguiente código de MAXIMA:

```
eq1: sn = s1*a^2 + s2*b^2 + s3*g^2$
eq2: tn^2 = (s1*a)^2 + (s2*b)^2 + (s3*g)^2 - sn^2$
eq3: a^2 + b^2 + g^2 = 1$                                            */
solve([eq1,eq2,eq3], [a,b,g])$ /* se resuelve alpha, beta y gamma    */
factor(%^2)$    /* se calculan alpha^2, beta^2 y gamma^2 y se factorizan
/* Aqui apareceran varios resultados, en listas diferentes, algunos de */
    ellos repetidos, por lo que es necesario quitar los repetidos     */
flatten(%)$     /* se ponen todos los resultados en una sola lista    */
unique(%);      /* se quitan todos los resultados repetidos
```

el cual retorna:

```
                2      2
          2   tn   + sn   - s3 sn - s2 sn + s2 s3
(%o7) [a  = --------------------------------,
                   (s2 - s1) (s3 - s1)


       2      2
 2   tn   + sn   - s3 sn - s1 sn + s1 s3
b  = - ----------------------------------,
            (s2 - s1) (s3 - s2)


      2      2
 2   tn   + sn   - s2 sn - s1 sn + s1 s2
g  = ---------------------------------]
          (s3 - s1) (s3 - s2)
```

determinamos las cantidades no negativas α^2, β^2 y γ^2:

Solución del Sistema

$$\begin{aligned}
\alpha^2 &= \frac{\tau_n^2+(\sigma_n-\sigma_2)(\sigma_n-\sigma_3)}{(\sigma_1-\sigma_2)(\sigma_1-\sigma_3)} \geq 0\\
\beta^2 &= \frac{\tau_n^2+(\sigma_n-\sigma_3)(\sigma_n-\sigma_1)}{(\sigma_2-\sigma_3)(\sigma_2-\sigma_1)} \geq 0\\
\gamma^2 &= \frac{\tau_n^2+(\sigma_n-\sigma_1)(\sigma_n-\sigma_2)}{(\sigma_3-\sigma_1)(\sigma_3-\sigma_2)} \geq 0;
\end{aligned} \tag{2.68}$$

recuerde que estas coordenadas de α, β y γ están dadas con respecto a la base especificada por $\hat{\boldsymbol{n}}_1$, $\hat{\boldsymbol{n}}_2$ y $\hat{\boldsymbol{n}}_3$.

Como nota, observe que las ecuaciones (2.68) también se pueden deducir al resolver el sistema de ecuaciones:

$$\begin{pmatrix}\sigma_1^2 & \sigma_2^2 & \sigma_3^2\\ \sigma_1 & \sigma_2 & \sigma_3\\ 1 & 1 & 1\end{pmatrix}\begin{pmatrix}\alpha^2\\ \beta^2\\ \gamma^2\end{pmatrix}=\begin{pmatrix}\sigma_n^2+\tau_n^2\\ \sigma_n\\ 1\end{pmatrix}$$

Sistema de ecuaciones
Esfuerzos principales

(%i14)
```
/**Definimos nuestro sistema de ecuaciones,
 * compuesto por 2.65, 2.66, 2.67
*/
eq1: sn = s1·a^2 + s2·b^2 + s3·g^2$
eq2: tn^2 = (s1·a)^2 + (s2·b)^2 + (s3·g)^2 - sn^2$
eq3: a^2 + b^2 + g^2 = 1$

/**Resolvemos para alpha, beta y gamma */
solve(
    [eq1, eq2, eq3],
    [a, b, g]
)$
factor(%^2)$ /**Se calculan y factorizan los cuadrados*/

/**Aquí aparecerán varios resultados en listas
 * diferentes, algunos de ellos repetidos, siendo
 * necesario quitarlos
*/
flatten(%)$ /**Se ponen los resultados en una lista */
transpose(
    unique(%)
); /**Se quitan las soluciones repetidas */
```

(%o14)
$$\begin{pmatrix} a^2 = \dfrac{tn^2 + sn^2 - s3\, sn - s2\, sn + s2\, s3}{(s2 - s1)\,(s3 - s1)} \\ b^2 = -\dfrac{tn^2 + sn^2 - s3\, sn - s1\, sn + s1\, s3}{(s2 - s1)\,(s3 - s2)} \\ g^2 = \dfrac{tn^2 + sn^2 - s2\, sn - s1\, sn + s1\, s2}{(s3 - s1)\,(s3 - s2)} \end{pmatrix}$$

el cual resulta al escribir en forma matricial las ecuaciones (2.66), (2.65) y (2.67), respectivamente.

Como $\sigma_1 > \sigma_2 > \sigma_3$, y $\alpha^2 \geq 0$, $\beta^2 \geq 0$ y $\gamma^2 \geq 0$, entonces los numeradores de (2.68) satisfacen:

Sistema Eqs.

I $\tau_n^2 + (\sigma_n - \sigma_2)(\sigma_n - \sigma_3) \geq 0,\quad$ ya que $\quad \sigma_1 - \sigma_2 > 0 \quad$ y $\quad \sigma_1 - \sigma_3 > 0$

II $\tau_n^2 + (\sigma_n - \sigma_3)(\sigma_n - \sigma_1) \leq 0,\quad$ ya que $\quad \sigma_2 - \sigma_3 > 0 \quad$ y $\quad \sigma_2 - \sigma_1 < 0$

III $\tau_n^2 + (\sigma_n - \sigma_1)(\sigma_n - \sigma_2) \geq 0,\quad$ ya que $\quad \sigma_3 - \sigma_1 < 0 \quad$ y $\quad \sigma_3 - \sigma_2 < 0$

$\to f(\sigma_2, \sigma_3)$

1: Eq I

Buscamos reescribir el sistema de una forma físicamente coherente

Expandiendo la primera desigualdad y sumando $\boxed{\frac{1}{4}(\sigma_2+\sigma_3)^2 - \sigma_2\sigma_3}$ a ambos lados de esta, obtenemos:

$$\sigma_n^2 - \sigma_n(\sigma_2+\sigma_3) + \frac{1}{4}(\sigma_2+\sigma_3)^2 + \tau_n^2 + \sigma_2\sigma_3 - \sigma_2\sigma_3 \geq \frac{1}{4}(\sigma_2+\sigma_3)^2 - \sigma_2\sigma_3$$

$$\left[\sigma_n - \tfrac{1}{2}(\sigma_2+\sigma_3)\right]^2 + \tau_n^2 \geq \frac{1}{4}(\sigma_2^2 + 2\sigma_2\sigma_3 + \sigma_3^2) - \sigma_2\sigma_3$$

$$\left[\sigma_n - \tfrac{1}{2}(\sigma_2+\sigma_3)\right]^2 + \tau_n^2 \geq \frac{\sigma_2^2}{4} - \frac{\sigma_2\sigma_3}{2} + \frac{\sigma_3^2}{4}$$

$$\left[\sigma_n - \tfrac{1}{2}(\sigma_2+\sigma_3)\right]^2 + \tau_n^2 \geq \left[\tfrac{1}{2}(\sigma_2-\sigma_3)\right]^2;$$

pasos análogos se pueden realizar con las otras dos desigualdades (se utiliza $\frac{1}{4}(\sigma_3 + \sigma_1)^2 - \sigma_3\sigma_1$ con la segunda desigualdad y $\frac{1}{4}(\sigma_1+\sigma_2)^2 - \sigma_1\sigma_2$ con la tercera), resultando:[30]

Gráfica diagrama círculos de Mohr 3D

1. $\left[\sigma_n - \tfrac{1}{2}(\sigma_2+\sigma_3)\right]^2 + \tau_n^2 \geq \left[\tfrac{1}{2}(\sigma_2-\sigma_3)\right]^2 \quad$ (ver circunferencia C_1) $\quad$ (2.69)

2. $\left[\sigma_n - \tfrac{1}{2}(\sigma_1+\sigma_3)\right]^2 + \tau_n^2 \leq \left[\tfrac{1}{2}(\sigma_1-\sigma_3)\right]^2 \quad$ (ver circunferencia C_2) $\quad$ (2.70)

3. $\underbrace{\left[\sigma_n - \tfrac{1}{2}(\sigma_1+\sigma_2)\right]^2}_{(x-x_0)^2} + \underbrace{\tau_n^2}_{(y-y_0)^2} \geq \underbrace{\left[\tfrac{1}{2}(\sigma_1-\sigma_2)\right]^2}_{r^2} \quad$ (ver circunferencia C_3) $\quad$ (2.71)

estas desigualdades definen respectivamente las ecuaciones de los tres círculos de Mohr para el esfuerzo, C_1, C_2 y C_3, con radios

$$R_1 = \tfrac{1}{2}(\sigma_2 - \sigma_3) \qquad R_2 = \tfrac{1}{2}(\sigma_1 - \sigma_3) \qquad R_3 = \tfrac{1}{2}(\sigma_1 - \sigma_2),$$

y cuyos centros tienen respectivamente las coordenadas

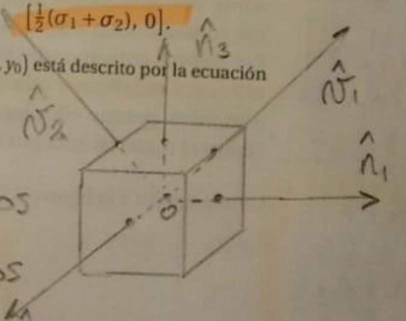
$$\left[\tfrac{1}{2}(\sigma_2+\sigma_3), 0\right] \qquad \left[\tfrac{1}{2}(\sigma_1+\sigma_3), 0\right] \qquad \left[\tfrac{1}{2}(\sigma_1+\sigma_2), 0\right].$$

[30] Recuerde que un círculo de radio r y con centro en el punto (x_0, y_0) está descrito por la ecuación

$$(x-x_0)^2 + (y-y_0)^2 = r^2$$

→ Esquema del sólido infinitesimal

En la figura los vectores unitarios que definen los planos en los que se presentan los esfuerzos normales ($\hat{n}$) y cortantes ($\hat{\nu}$) principales.



→Es fácil determinar los esfuerzos normales y cortantes correspondientes al estado de cortante máximo, tal como se sigue en el programa de Maxima adjunto

Básicamente se reemplazan dichos esfuerzos en las ecuaciones 2.68

Obteniendo:

$\alpha^2 = \frac{1}{2}$

$\beta^2 = 0$

$\gamma^2 = \frac{1}{2}$

→Lo que nos lleva a dos posibles vectores que cumplen con dicha condición:

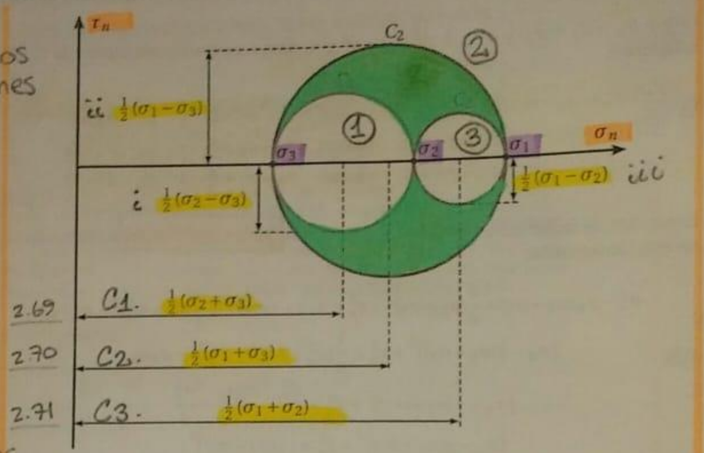


Figura 2.22: *Círculo de Mohr en tres dimensiones. Aquí la región en verde junto con los bordes de las tres circunferencias C_1, C_2 y C_3 constituyen el conjunto de puntos (σ_n, τ_n) que representan el estado de esfuerzos normal y cortante que satisfacen las desigualdades (2.69), (2.70) y (2.71); todo estado de esfuerzo en un sólido tridimensional debe necesariamente yacer sobre la zona mencionada.*

$\hat{v}_1 = \left[\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right]$ ∧ $\hat{v}_2 = \left[\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}\right]^T$, con respecto a la base: $\{\hat{n}_1, \hat{n}_2, \hat{n}_3\}$

→Dichos vectores describen los planos:

$x + z = 0$

$x - z = 0$

(respectivamente)

Estas desigualdades muestran que todos los puntos de esfuerzo admisible (σ_n, τ_n) yacen, ya sea sobre las circunferencias o dentro del área sombreada encerrada por los círculos, tal y como se muestra en la Figura 2.22. Los puntos de esfuerzo (σ_n, τ_n) que satisfacen la desigualdad (2.69) del círculo C_1 se ubican sobre la circunferencia o fuera del círculo C_1. Los puntos de esfuerzo (σ_n, τ_n) que satisfacen la desigualdad (2.70) para el círculo C_2 yacen sobre la circunferencia o dentro del círculo C_2; finalmente, los puntos de esfuerzo (σ_n, τ_n) que satisfacen la ecuación del círculo C_3, desigualdad (2.71), se localizan ya sea sobre la circunferencia o fuera del círculo C_3.

→Dado que:

$$\hat{n}_1 + \hat{n}_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\|\hat{n}_1 + \hat{n}_3\| = \sqrt{2}$$

$$\Downarrow$$

$$\hat{v}_1 = \frac{\hat{n}_1 + \hat{n}_3}{\|\hat{n}_1 + \hat{n}_3\|}$$

$$\hat{v}_2 = \frac{\hat{n}_1 - \hat{n}_3}{\|\hat{n}_1 - \hat{n}_3\|}$$

Observe que los círculos de Mohr están ubicados sobre el eje σ_n y se observa que sobre este existen tres puntos para los cuales el esfuerzo cortante $\tau_n = 0$; estos puntos corresponden a aquellas superficies que no están sujetas a esfuerzos cortantes. Los puntos correspondientes a σ_1 y σ_3 representan, respectivamente, los esfuerzos normales máximos y mínimos que se presentan en el material y son bastante importantes, porque ellos están directamente relacionados con los esfuerzos para los cuales el material falla por la acción del esfuerzo normal.

Con respecto a la Figura 2.22 se puede deducir que el esfuerzo cortante máximo está dado por:

$$\boxed{\tau_{máx} = \frac{\sigma_1 - \sigma_3}{2}} \Longrightarrow \sigma_n = \frac{1}{2}(\sigma_1 + \sigma_3)$$

correspondiente

```
(%i17) /**Se definen los esfuerzos normales y cortantes en el
        * punto de cortante máximo, en función de los esfuerzos
        * normales principales (s1, s2, s3)
       */
       sn: (s1 + s3)/2$
       tn: (s1 - s3)/2$

       /**Conociendo las expresiones para los cosenos directores
        * alpha, beta y gamma, dadas por 2.68, podemos calcular
        * las componentes del vector normal asociado a las
        * condiciones de cortante máximo
       */
       alpha2: factor((tn^2+sn^2-s3·sn-s2·sn+s2·s3)/((s2-s1)·(s3-s1)));
       beta2: factor(-(tn^2+sn^2-s3·sn-s1·sn+s1·s3)/((s2-s1)·(s3-s2)));
       gamma2: factor((tn^2+sn^2-s2·sn-s1·sn+s1·s2)/((s3-s1)·(s3-s2)));
```

(%o15) $\frac{1}{2}$

(%o16) 0

(%o17) $\frac{1}{2}$

A continuación, resolveremos los ejercicios planteados en el presente capítulo del libro valiéndonos de dos programas en Python. Con el primer código pretendemos obtener los vectores que describen los planos en los que se presentan los esfuerzos normal y cortante dados por el problema; para una matriz de tensiones conocida, y considerando que dichas direcciones están definidas con respecto a la base ortogonal de las direcciones principales. Con el segundo código pretendemos graficar el diagrama de círculos de Mohr para el caso tridimensional generalizado, obteniendo así mismo los esfuerzos y direcciones principales, el esfuerzo cortante máximo y el plano en el que este se presenta. Se presentan en ambos casos los resultados para los problemas particulares planteados en el libro, juntamente con el algoritmo que es válido en general y, por lo tanto, replicable.

```python
import numpy as np
'''
Considere un punto P sometido a los esfuerzos: sigma_x, sigma_y,
sigma_z, tao_xy, tao_xz, tao_yz
   ---> Encuentre las direcciones y los esfuerzos principales en dicho punto
   ---> Encuentre los vectores normalizados que definen los planos para los
   cuales se presenta el estado de esfuerzos: sigma_n, tao_n
'''
# ==============================================================================
def solution(U):
    #Find the eigenvalues and eigenvector of U(transpose).U
    e_vals, e_vecs = np.linalg.eig(np.dot(U.T, U))
    #Extract the eigenvector (column) associated with the
    # minimum eigenvalue
    return e_vecs[:, np.argmin(e_vals)]
# ==============================================================================
sigma_x = 1
sigma_y = 3
sigma_z = 0
tao_xy = 0
tao_xz = 0
tao_yz = 2
# ==============================================================================
#Determinamos el estado de esfuerzos requerido en el plano v:
sigma_n = 0.1
tao_n = 2
# ==============================================================================
#Definimos la matriz de tensiones sigma_matrix
sigma_matrix = np.zeros((3,3))
sigma_matrix[0,0] = sigma_x
sigma_matrix[0,1] = tao_xy
sigma_matrix[0,2] = tao_xz
sigma_matrix[1,0] = tao_xy
sigma_matrix[1,1] = sigma_y
sigma_matrix[1,2] = tao_yz
sigma_matrix[2,0] = tao_xz
sigma_matrix[2,1] = tao_yz
sigma_matrix[2,2] = sigma_z

#Definimos los coeficientes del polinomio característico
# asociado a sigma_matrix
# a*sigma_n**3+b*sigma_n**2+c*sigma_n+d = 0
char_poly = np.poly(sigma_matrix)

#Definimos las raíces del polinomio en sigma_n
poly_roots = np.roots(char_poly)
poly_roots = np.sort(poly_roots)[::-1] #Ordenamos de mayor a menor
```

```
#Resolvemos los sistemas de ecuaciones a fin de obtener los
# vectores asociados a las direcciones principales
# =============================================================================
# #Esfuerzo principal sigma_1
# =============================================================================
sigma_1 = poly_roots[0]
A1 = sigma_matrix
A1 = A1-sigma_1*np.identity(3)
result1 = solution(A1)
# =============================================================================
# #Esfuerzo principal sigma_2
# =============================================================================
sigma_2 = poly_roots[1]
A2 = sigma_matrix
A2 = A2-sigma_2*np.identity(3)
result2 = solution(A2)
# =============================================================================
# #Esfuerzo principal sigma_3
# =============================================================================
sigma_3 = poly_roots[2]
A3 = sigma_matrix
A3 = A3-sigma_3*np.identity(3)
result3 = solution(A3)
# =============================================================================
#Calculamos el valor absoluto de los cosenos directores correspondientes a los
# planos v según las ecuaciones 2.68
arg_1 = (tao_n**2+(sigma_n-sigma_2)*(sigma_n-sigma_3))/((sigma_1-sigma_2)*(sigma_1-sigma_3))
alpha = np.abs(np.sqrt(arg_1))

arg_2 = (tao_n**2+(sigma_n-sigma_3)*(sigma_n-sigma_1))/((sigma_2-sigma_3)*(sigma_2-sigma_1))
beta = np.abs(np.sqrt(arg_2))

arg_3 = (tao_n**2+(sigma_n-sigma_1)*(sigma_n-sigma_2))/((sigma_3-sigma_1)*(sigma_3-sigma_2))
gamma = np.abs(np.sqrt(arg_3))

#Definimos los 4 vectores asociados a las combinaciones de los cosenos
# directores calculados anteriormente, recuérdese que dichos vectores
# están definidos con respecto a la base ortogonal de las direcciones principales
v1 = np.transpose([alpha, beta, gamma])
v2 = np.transpose([alpha, beta, -gamma])
v3 = np.transpose([alpha, -beta, gamma])
v4 = np.transpose([alpha, -beta, -gamma])
# =============================================================================
print("Esfuerzos principales: \n", poly_roots," [Pa]")
print("Direcciones principales: \n", result1, " (Principal mayor) \n",
          result2, "(Principal media) \n", result3, "(Principal menor) \n")
print("Dirección 1 correspondiente a sigma_n, tao_n: \n", v1)
print("Dirección 2 correspondiente a sigma_n, tao_n: \n", v2)
print("Dirección 3 correspondiente a sigma_n, tao_n: \n", v3)
print("Dirección 4 correspondiente a sigma_n, tao_n: \n", v4)
```

```
In [2]: runfile('C:/Users/Pablo Teixeira/Documents/Civil Pro/
SOLID_MECHANICS/Homework/TALLER_6/TALLER6_Pack/untitled1.py', wdir='C:/
Users/Pablo Teixeira/Documents/Civil Pro/SOLID_MECHANICS/Homework/TALLER_6/
TALLER6_Pack')
Esfuerzos principales:
 [ 4.  1. -1.]  [Pa]
Direcciones principales:
 [0.         0.89442719 0.4472136 ]  (Principal mayor)
 [1. 0. 0.] (Principal media)
 [ 0.          0.4472136  -0.89442719] (Principal menor)

Dirección 1 correspondiente a sigma_n, tao_n:
 [0.44795833 0.21984843 0.86660256]
Dirección 2 correspondiente a sigma_n, tao_n:
 [ 0.44795833  0.21984843 -0.86660256]
Dirección 3 correspondiente a sigma_n, tao_n:
 [ 0.44795833 -0.21984843  0.86660256]
Dirección 4 correspondiente a sigma_n, tao_n:
 [ 0.44795833 -0.21984843 -0.86660256]
```

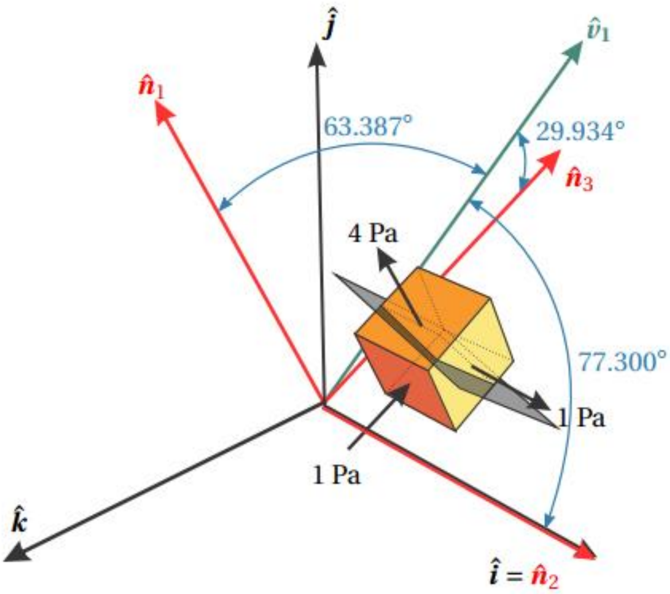


Figura 2.23: *Plano con vector normal* $\hat{v}_1 = [+0.44796,\ +0.21985,\ +0.86660]^T$ *definido con respecto al sistema de coordenadas formado por la base* $\{\hat{n}_1, \hat{n}_2, \hat{n}_3\}$.

```python
import numpy as np
import matplotlib.pyplot as plt
'''
Considere un punto P sometido a los esfuerzos: sigma_x, sigma_y,
sigma_z, tao_xy, tao_xz, tao_yz
   ---> Encuentre las direcciones y los esfuerzos principales en dicho punto
   ---> Encuentre los vectores normalizados que definen los planos en que se
   presentan los cortantes máximos, así como la magnitud de dichos esfuerzos
   ---> Grafique el diagrama de círculos de Mohr correspondiente a este
   estado de esfuerzos
'''
# ==============================================================================
def solution(U):
    #Find the eigenvalues and eigenvector of U(transpose).U
    e_vals, e_vecs = np.linalg.eig(np.dot(U.T, U))
    #Extract the eigenvector (column) associated with the
    # minimum eigenvalue
    return e_vecs[:, np.argmin(e_vals)]
# ==============================================================================
sigma_x = 1
sigma_y = 3
sigma_z = 0
tao_xy = 0
tao_xz = 0
tao_yz = 2
# ==============================================================================
#Definimos la matriz de tensiones sigma_matrix
sigma_matrix = np.zeros((3,3))
sigma_matrix[0,0] = sigma_x
sigma_matrix[0,1] = tao_xy
sigma_matrix[0,2] = tao_xz
sigma_matrix[1,0] = tao_xy
sigma_matrix[1,1] = sigma_y
sigma_matrix[1,2] = tao_yz
sigma_matrix[2,0] = tao_xz
sigma_matrix[2,1] = tao_yz
sigma_matrix[2,2] = sigma_z
```

```python
#Definimos los coeficientes del polinomio característico
# asociado a sigma_matrix
# a*sigma_n**3+b*sigma_n**2+c*sigma_n+d = 0
char_poly = np.poly(sigma_matrix)

#Definimos las raíces del polinomio en sigma_n
poly_roots = np.roots(char_poly)
poly_roots = np.sort(poly_roots)[::-1] #Ordenamos de mayor a menor
```

```python
#Resolvemos los sistemas de ecuaciones a fin de obtener los
# vectores asociados a las direcciones principales
# ==============================================================================
# #Esfuerzo principal sigma_1
# ==============================================================================
sigma_1 = poly_roots[0]
A1 = sigma_matrix
A1 = A1-sigma_1*np.identity(3)
result1 = solution(A1)
# ==============================================================================
# #Esfuerzo principal sigma_2
# ==============================================================================
sigma_2 = poly_roots[1]
A2 = sigma_matrix
A2 = A2-sigma_2*np.identity(3)
result2 = solution(A2)
# ==============================================================================
# #Esfuerzo principal sigma_3
# ==============================================================================
sigma_3 = poly_roots[2]
A3 = sigma_matrix
A3 = A3-sigma_3*np.identity(3)
result3 = solution(A3)
# ==============================================================================
#Determinamos el esfuerzo cortante máximo y los vectores que definen los
# planos en los que se presenta, según las expresiones conocidas
tao_max = (sigma_1-sigma_3)/2
v1_diag = (result1+result3)/np.linalg.norm(result1+result3)
v2_diag = (result1-result3)/np.linalg.norm(result1-result3)

# ==============================================================================
# DIAGRAMA DE CÍRCULOS DE MOHR 3D
# ==============================================================================

# ==============================================================================
#Definimos los parámetros de la circunferencia exterior C2
delta = 0.005 #Accuracy
r2 = (sigma_1-sigma_3)/2   #Radio de la semicircunferencia MOHR
h2 = (sigma_1+sigma_3)/2 #Centro de la semicircunferencia MOHR

sigma_mohr2 = np.arange(sigma_3, sigma_1+delta, delta) #Esfuerzos normales MOHR
n2_parameter = np.size(sigma_mohr2)
ang2_parameter = np.linspace(0, 2*np.pi, n2_parameter+1)

sigma_c2 = r2*np.cos(ang2_parameter)+h2
tao_c2 = r2*np.sin(ang2_parameter)
# ==============================================================================
#Definimos los parámetros de la circunferencia interior C1
r1 = (sigma_2-sigma_3)/2   #Radio de la semicircunferencia MOHR
h1 = (sigma_2+sigma_3)/2 #Centro de la semicircunferencia MOHR

sigma_mohr1 = np.arange(sigma_3, sigma_2+delta, delta) #Esfuerzos normales MOHR
n1_parameter = np.size(sigma_mohr1)
ang1_parameter = np.linspace(0, 2*np.pi, n1_parameter+1)

sigma_c1 = r1*np.cos(ang1_parameter)+h1
tao_c1 = r1*np.sin(ang1_parameter)
```

```python
# ==============================================================================
#Definimos los parámetros de la circunferencia interior C3
r3 = (sigma_1-sigma_2)/2    #Radio de la semicircunferencia MOHR
h3 = (sigma_1+sigma_2)/2 #Centro de la semicircunferencia MOHR

sigma_mohr3 = np.arange(sigma_2, sigma_1+delta, delta) #Esfuerzos normales MOHR
n3_parameter = np.size(sigma_mohr3)
ang3_parameter = np.linspace(0, 2*np.pi, n3_parameter+1)

sigma_c3 = r3*np.cos(ang3_parameter)+h3
tao_c3 = r3*np.sin(ang3_parameter)
# ==============================================================================
#Definimos los puntos en la gráfica que corresponden a los esfuerzos
# principales (normales) y cortantes máximos
point_sigma1 = [sigma_1, 0]
point_sigma2 = [sigma_2, 0]
point_sigma3 = [sigma_3, 0]
point_taomax = [h2, tao_max]
point_taomin = [h2, -tao_max]
# ==============================================================================
print("Esfuerzos principales: \n", poly_roots," [Pa]")
print("Esfuerzo cortante máximo: \n", tao_max," [Pa]")
print("Direcciones principales: \n", result1, " (Principal mayor) \n",
          result2, "(Principal media) \n", result3, "(Principal menor) \n")
print("Dirección 1 correspondiente a tao_max: \n", v1_diag)
print("Dirección 2 correspondiente a tao_max: \n", v2_diag)
# ==============================================================================
plt.axis('scaled')
plt.ylim(-r2-0.3, r2+0.7)
plt.xlim(sigma_3-0.3, sigma_1+0.7)
plt.grid()

plt.plot(sigma_c2, tao_c2)
plt.plot(sigma_c1, tao_c1)
plt.plot(sigma_c3, tao_c3)

plt.plot(point_sigma1[0],point_sigma1[1],'go')
plt.annotate('S1', point_sigma1, xytext=(4,6),textcoords='offset pixels')
plt.plot(point_sigma2[0],point_sigma2[1],'go')
plt.annotate('S2', point_sigma2, xytext=(4,6),textcoords='offset pixels')
plt.plot(point_sigma3[0],point_sigma3[1],'go')
plt.annotate('S3', point_sigma3, xytext=(4,6),textcoords='offset pixels')
plt.plot(point_taomax[0],point_taomax[1],'go')
plt.annotate('Tao', point_taomax, xytext=(4,6),textcoords='offset pixels')
plt.plot(point_taomin[0],point_taomin[1],'go')
plt.annotate('Tao', point_taomin, xytext=(4,6),textcoords='offset pixels')

plt.show()
# ==============================================================================
```

```
In [1]: runfile('C:/Users/Pablo Teixeira/Documents/Civil Pro/SOLID_MECHANICS/
Homework/TALLER_6/TALLER6_Pack/Circulo_Mohr_3D.py', wdir='C:/Users/Pablo
Teixeira/Documents/Civil Pro/SOLID_MECHANICS/Homework/TALLER_6/TALLER6_Pack')
Esfuerzos principales:
 [ 4.  1. -1.]  [Pa]
Esfuerzo cortante máximo:
 2.5000000000000013  [Pa]
Direcciones principales:
 [0.         0.89442719 0.4472136 ]  (Principal mayor)
 [1. 0. 0.] (Principal media)
 [ 0.          0.4472136  -0.89442719] (Principal menor)

Dirección 1 correspondiente a tao_max:
 [ 0.          0.9486833  -0.31622777]
Dirección 2 correspondiente a tao_max:
 [0.         0.31622777 0.9486833 ]
```