Modelação de Sistemas Físicos - Aula Prática nº5

Realização e resolução de problemas vetoriais e a sua representação gráfica

Lista de operações, funções e métodos úteis (numpy)

Descrição	Exemplo	Equiv. Matemático
Criação	`a = np.array([1, 2, 3])`	$\mathbf{a} = (1, 2, 3)$
Operações elemento-a-elemento	`b = np.ones(3) + 1`	$b_i = 1_i + 1 = (2, 2, 2)$
	`c = a * b`	$c_i = a_i b_i = (2, 4, 6)$
	`c = a ** b`	$c_i = a_i^{b_i} = (1, 4, 9)$
Produto interno (escalar)	`c = np.inner(a, b)`	$\mathbf{c} = \mathbf{a} \cdot \mathbf{b} = 2 + 4 + 6 = 12$
Produto externo (vetorial)	`c = np.cross(a, b)`	$\mathbf{c} = \mathbf{a} \times \mathbf{b} = (-2, 4, -2)$
Soma dos elementos	`c = np.sum(a)`	$c = \sum_i a_i = 6$
Módulo vetorial	`c = np.linalg.norm(a)`	$c = \lvert\mathbf{a}\rvert = \sqrt{\sum_i a_i^2}$

Problema 1

Um vetor a 2 dimensões tem as coordenadas $(3, 4)$.

a) Qual a sua intensidade ou comprimento?

$$\mathbf{a} = (3,\ 4)$$

$$a = \sqrt{3^2 + 4^2} = 5$$

In [1]:
```
import numpy as np

a = np.array([3,4])
a_norma = np.linalg.norm(a)
print('A intensidade do vetor "a" é', a_norma)
```

```
A intensidade do vetor "a" é 5.0
```

b) Qual o vetor unitário correspondente?

$$\hat{\mathbf{a}} = \frac{\mathbf{a}}{a} = \frac{(3,\ 4)}{5}$$

$$\hat{\mathbf{a}} = \left(\frac{3}{5}, \frac{4}{5}\right)$$

In [2]:
```python
print('O vetor unitário de "a" é', a / a_norma)
```

```
O vetor unitário de "a" é [0.6 0.8]
```

c) Determine o vetor $2\mathbf{a}$? Qual é o seu comprimento ou módulo?

$$2\mathbf{a} = 2 \times (3,\ 4) = (6,\ 8)$$

$$\|2\mathbf{a}\| = \sqrt{6^2 + 8^2} = 10 = 2a$$

In [3]:
```python
b = 2 * a
print('O vetor "2*(3,4)" é', b)
print('A sua norma é', np.linalg.norm(b))
```

```
O vetor "2*(3,4)" é [6 8]
A sua norma é 10.0
```

Problema 2

Considere os dois vetores $(1, 2)$ e $(-2, 3)$. Qual o seu produto escalar e qual o ângulo entre eles?

$$(1, 2) \cdot (-2, 3) = 1 \times (-2) + 2 \times 3 = 4$$

Módulo/norma é 4

$$4 = \|(1, 2)\| \times \|(-2, 3)\| \ \cos\theta$$

$$\cos\theta = \frac{4}{\sqrt{1^2 + 2^2}\sqrt{2^2 + 3^2}} = \frac{4}{\sqrt{65}}$$

$$\theta = \arccos\left(\frac{4}{\sqrt{65}}\right) = 60.255118\ldots^\circ$$

O produto escalar pode ser obtido pela função `np.inner`

In [4]:
```python
a = (1, 2)
b = (-2, 3)

a_dot_b = np.inner(a, b)
print('O produto escalar é', a_dot_b)
```

```
O produto escalar é 4
```

O produto escalar é definido por $\mathbf{a} \cdot \mathbf{b} = ab\cos\theta$, onde a e b são as normas dos vetores, e θ é o ângulo formado entre eles. Assim temos:

$$\cos\theta = \frac{\mathbf{a}\cdot\mathbf{b}}{ab},$$

e finalmente

$$\theta = \arccos\left(\frac{\mathbf{a}\cdot\mathbf{b}}{ab}\right)$$

In [5]:
```
a_norm = np.linalg.norm(a)
b_norm = np.linalg.norm(b)

# Note-se que as unidades da função arccos são radianos.
#      π rad = 180°, ou seja rad = 180/π
theta = np.arccos(a_dot_b / (a_norm * b_norm)) * 180 / np.pi
print('O ângulo formado entre "a" e "b" é', theta, "°")
```

```
O ângulo formado entre "a" e "b" é 60.25511870305777 °
```

Problema 3

Encontre um vetor perpendicular ao vetor $(3, 4)$, no espaço a 2D.

Note que o produto escalar de dois vetores perpendiculares é nulo.

Use a função `arrow()` do `matplotlib.pyplot` para representar graficamente o vetor $(3, 4)$ e o vetor perpendicular encontrado. Verifica se aparecem perpendiculares.

Solução:

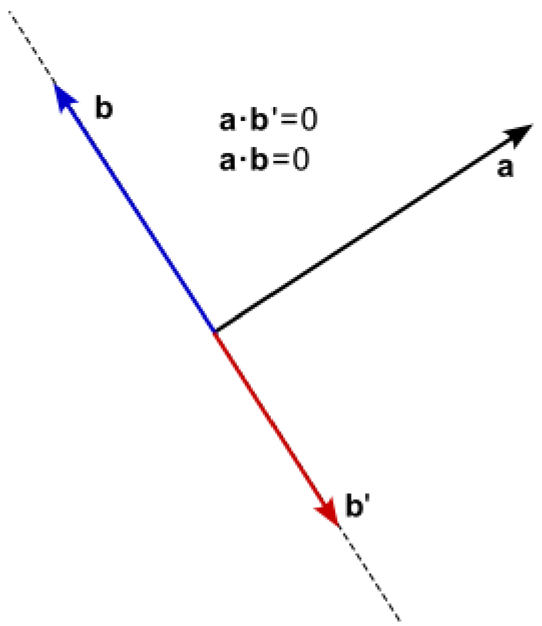
Considerando que $\mathbf{a}\cdot\mathbf{b} = ab\cos\theta$, então quando $\mathbf{a}\perp\mathbf{b}$ temos $\cos\theta = 0$ e

$$\sum_i a_i b_i = 0$$

Para o caso de $\mathbf{a} = (3, 4)$, of vetor $\mathbf{b} = (b_0, b_1)$ é obtido da seguinte forma,

$$3b_0 + 4b_1 = 0 \Leftrightarrow b_1 = -3b_0/4.$$

Notemos que existem várias soluções para o problema,



Basta então escolher um valor para b_0, por exemplo $b_0 = 4$ para obtermos

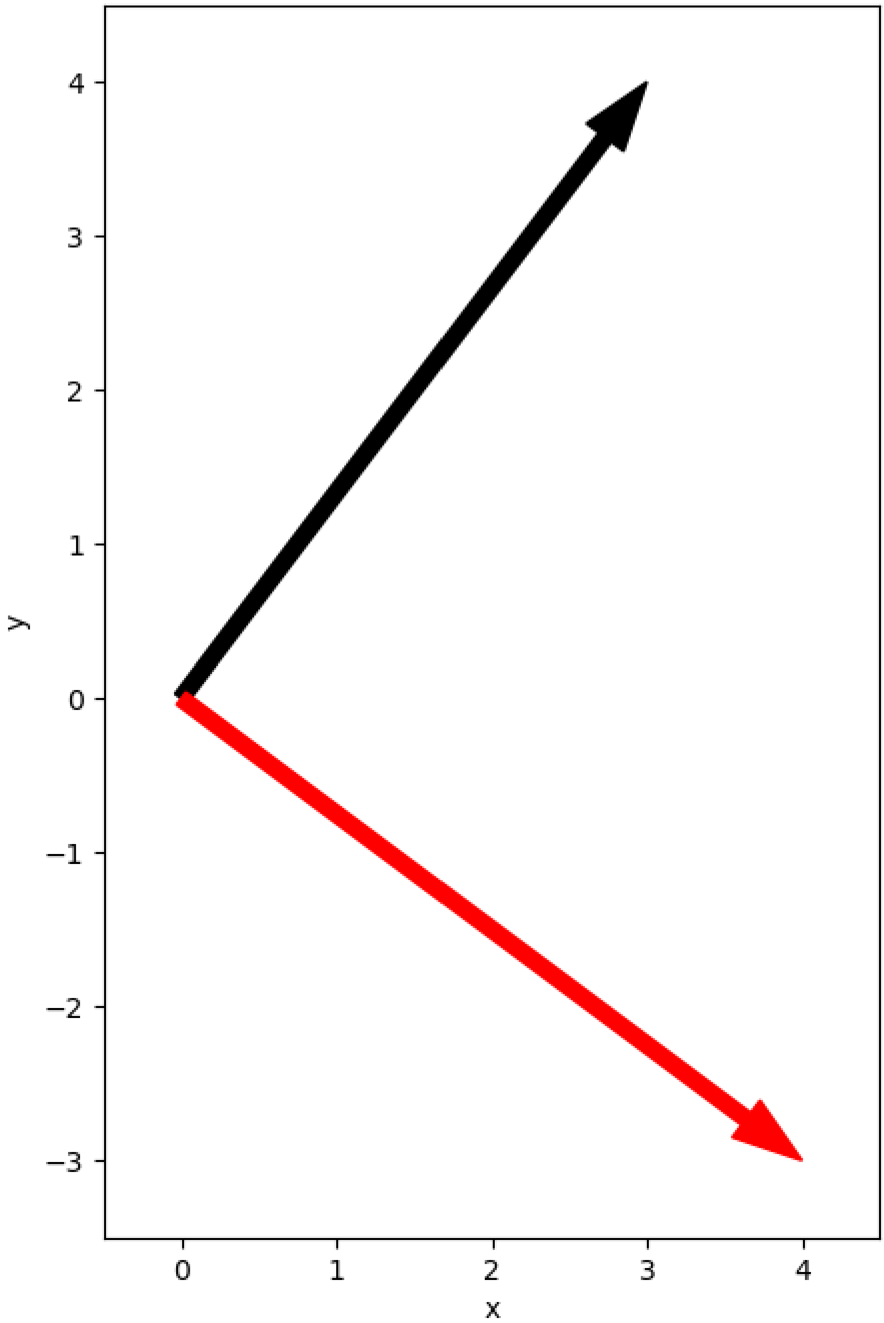
$$\mathbf{b} = (4, -3)$$

Podemos então verificar $\mathbf{a} \cdot \mathbf{b} = 3 \times 4 + 4 \times (-3) = 0$

Problema 4

Use a função `arrow()` de `matplotlib.pyplot` para representar graficamente o vetor $\mathbf{a} = (3,\ 4)$ e o vetor $\mathbf{b}$ perpendicular encontrado no problema anterior. Verifique que são efetivamente perpendiculares.

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt

a = np.array([3,4])
b = np.array([4,-3])

plt.figure(figsize=(5,8))
plt.axis([-0.5, 4.5, -3.5, 4.5])
plt.arrow(0,0,a[0],a[1],color='k',width=0.1, length_includes_head=True)   # represent
plt.arrow(0,0,b[0],b[1],color='r',width=0.1, length_includes_head=True)   # represent
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```



Podemos verificar que de facto são perpendiculares

Problema 5

Um simples robô pode deslocar-se no chão executando dois tipos de instruções. Pode rodar no sentido horário por um determinado ângulo, e pode avançar em linha reta uma determinada distância.

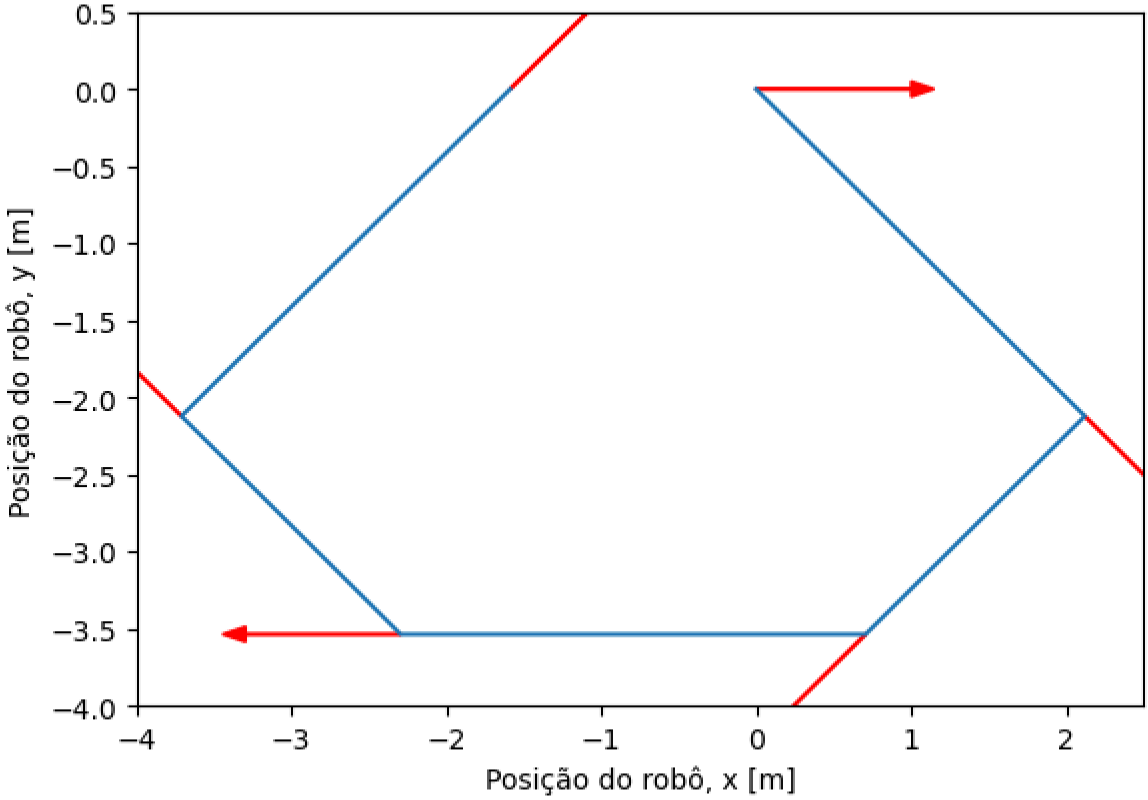
As instruções são dados ao robô na forma de *tuples* `(ang, dist)` , que significa que o robô deve rodar por um ângulo `ang` (em graus) e depois avançar uma distância `dist` (metros).

O robô começa na origem, orientado ao longo do eixo x. É-lhe dada a seguinte sequência de instruções:

(45,3), (90,2), (45,3), (45,2), (90,3)

a) Calcule a posição do robô após cada passo. Faça um gráfico da trajetória do robô.

In [7]:
```python
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(6.5,4.5))       # Preparar a representação gráfica
plt.axis([-4, 2.5, -4, 0.5])

x = 0.0; y = 0.0; theta = 0.0
pos = np.array([[x, y, theta]])     # posicionamento. Robô inicia na orígem com ang=0
plt.arrow(x, y, np.cos(-theta * np.pi / 180), np.sin(-theta * np.pi / 180), color='r'
          width=0.01, head_width=0.1)  # representar direção do robô

ang = 45; dist = 3                  # instrução 1
theta += ang
x += dist * np.cos(-theta * np.pi / 180)
y += dist * np.sin(-theta * np.pi / 180)
pos = np.append(pos, [[x, y, theta]], 0)
plt.arrow(x, y, np.cos(-theta * np.pi / 180), np.sin(-theta * np.pi / 180), color='r'
          width=0.01, head_width=0.1)  # representar direção do robô

ang = 90; dist = 2                  # instrução 2
theta += ang
x += dist * np.cos(-theta * np.pi / 180)
y += dist * np.sin(-theta * np.pi / 180)
pos = np.append(pos, [[x, y, theta]], 0)
plt.arrow(x, y, np.cos(-theta * np.pi / 180), np.sin(-theta * np.pi / 180), color='r'
          width=0.01, head_width=0.1)  # representar direção do robô

ang = 45; dist = 3                  # instrução 3
theta += ang
x += dist * np.cos(-theta * np.pi / 180)
y += dist * np.sin(-theta * np.pi / 180)
pos = np.append(pos, [[x, y, theta]], 0)
plt.arrow(x, y, np.cos(-theta * np.pi / 180), np.sin(-theta * np.pi / 180), color='r'
          width=0.01, head_width=0.1)  # representar direção do robô

ang = 45; dist = 2                  # instrução 4
theta += ang
x += dist * np.cos(-theta * np.pi / 180)
y += dist * np.sin(-theta * np.pi / 180)
pos = np.append(pos, [[x, y, theta]], 0)
plt.arrow(x, y, np.cos(-theta * np.pi / 180), np.sin(-theta * np.pi / 180), color='r'
          width=0.01, head_width=0.1)  # representar direção do robô

ang = 90; dist = 3                  # instrução 5
theta += ang
x += dist * np.cos(-theta * np.pi / 180)
y += dist * np.sin(-theta * np.pi / 180)
pos = np.append(pos, [[x, y, theta]], 0)
plt.arrow(x, y, np.cos(-theta * np.pi / 180), np.sin(-theta * np.pi / 180), color='r'
          width=0.01, head_width=0.1)  # representar direção do robô

plt.plot(pos[:,0], pos[:,1])
plt.xlabel("Posição do robô, x [m]")
plt.ylabel("Posição do robô, y [m]")
plt.show()
```



b) Quais são as coordenadas finais do robô?

In [8]:
```
x_f = x; y_f = y; theta_f = theta
print("As coordenadas finais do robô são:")
print("    r = ({0:.2f}, ".format(x_f), "{0:.2f})".format(y_f))
print("    ang = {0:.2f}".format(theta_f))
```

```
As coordenadas finais do robô são:
    r = (-1.59,  -0.00)
    ang = 315.00
```

c) Qual é a instrução necessária para fazer o robô retornar ao ponto inicial?

Considerando que $\mathbf{p}_\mathrm{f} = (\mathbf{r}_\mathrm{f},\, \theta_\mathrm{f})$ são as coordenadas finais do robô, onde $\mathbf{r}_\mathrm{f} = (x_\mathrm{f},\, y_\mathrm{f})$, então a intrução para o retorno ao início (orígem das coordenadas), consite em avançar ao longo de um vetor $-\mathbf{r}_\mathrm{f}$, ou seja

1. Orientar o robô ao longo da direção $\theta = \arcsin(-y_\mathrm{f}/r_\mathrm{f})$
2. Percorrer uma distância $d = r_\mathrm{f}$.

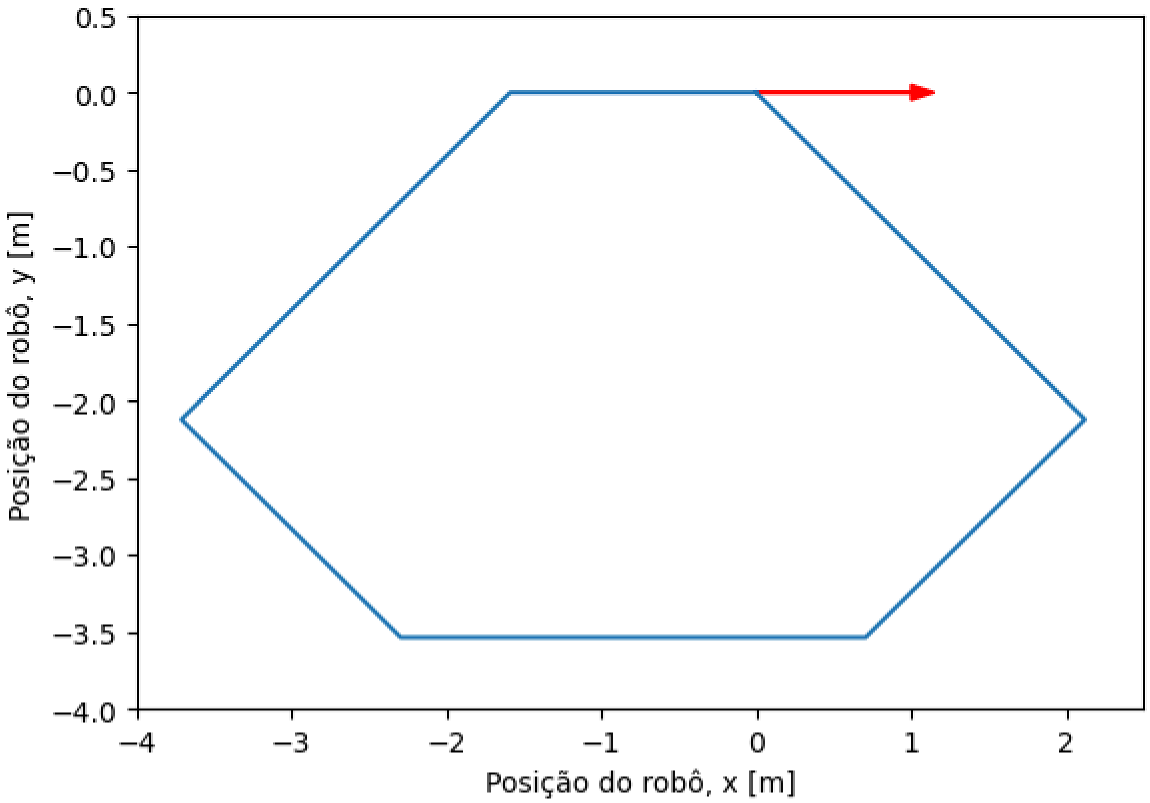
Considerando que o último ângulo (orientação do robô) era θ_f, o robô terá primeiro que rodar $\delta\theta = \arcsin(-y_\mathrm{f}/r_\mathrm{f}) - \theta_\mathrm{f}$ e depois avançar a distância d.

Note-se que $\delta\theta \geq 0$ (robô só roda no sentido dos ponteiros do relógio), e portanto $\delta\theta$ é o menor positivo tal que $\delta\theta = \arcsin(-y_\mathrm{f}/r_\mathrm{f}) - \theta_\mathrm{f} + 360n$, em que n é um número inteiro. Esta última operação pode ser efetuada com a função `numpy.remainder()`

In [9]:
```
d = np.sqrt(x_f**2 + y_f**2)

plt.figure(figsize=(6.5,4.5))
plt.axis([-4, 2.5, -4, 0.5])

#ang = np.arcsin(-y_f / d) - theta_f                          # 1. Rotação <= 360 graus
ang = np.remainder(np.arcsin(-y_f / d) - theta_f, 360)    # 1. Rotação ângulo arbitrár
dist = d                                                  # 2. Avanço
```

```
theta += ang
x += dist * np.cos(-theta * np.pi / 180)
y += dist * np.sin(-theta * np.pi / 180)
pos = np.append(pos, [[x, y, theta]], 0)
print("Instrução de retorno:")
print("    dist = ({0:.2f}, ".format(dist))
print("    ang = {0:.2f}".format(ang))
plt.arrow(x, y, np.cos(-theta * np.pi / 180), np.sin(-theta * np.pi / 180), color='r'

plt.plot(pos[:,0], pos[:,1])
plt.xlabel("Posição do robô, x [m]")
plt.ylabel("Posição do robô, y [m]")
plt.show()
```

```
Instrução de retorno:
    dist = (1.59,
    ang = 45.00
```



Problema 6

Um semáforo que tem um peso $T_3 = 200\ \mathrm{N}$ (massa vezes aceleração gravítica) é suspenso por dois cabos que fazem ângulos θ_1 e θ_2 com o horizontal, como mostrado na ilustração.

Se $\theta_1 = 30^\circ$ e $\theta_2 = 60^\circ$, quais terão que ser as forças T_1 e T_2 para que o sumáforo não caia?

Note que para um objeto se manter parado, a soma das forças (força resultante) deve ser nula.

Começa por calcular os componentes horizontais e verticais de cada força.

Solução:

$$\sum_i \mathbf{T}_i = \mathbf{T}_1 + \mathbf{T}_2 + \mathbf{T}_3 = \mathbf{0}$$

$$T_1(-\cos\theta_1,\ \sin\theta_1) + T_2(\cos\theta_2,\ \sin\theta_2) + T_3(0,\ -1) = (0,\ 0)$$

$$T_1\left(-\frac{\sqrt{3}}{2},\,\frac{1}{2}\right)+T_2\left(\frac{1}{2},\,\frac{\sqrt{3}}{2}\right)+T_3(0,\,-1)=(0,\,0)$$

Solving for each component,

$$\begin{cases} -\sqrt{3}T_1+T_2 &=& 0 \\ T_1+\sqrt{3}T_2 &=& 2T_3 = 400\,\mathrm{N} \\ T_3 &=& 200\,\mathrm{N} \end{cases}$$

e finalmente,

$$\begin{cases} T_1 &=& 100\,\mathrm{N} \\ T_2 &=& 100\sqrt{3}\,\mathrm{N} \\ T_3 &=& 200\,\mathrm{N} \end{cases}$$

In [10]:
```
theta_1 = 30.0
theta_2 = 60.0
T_1 = 100.0 * np.array([-np.cos(theta_1 * np.pi / 180), np.sin(theta_1 * np.pi / 180)
T_2 = 100.0 * np.sqrt(3) * np.array([np.cos(theta_2 * np.pi / 180), np.sin(theta_2 *
T_3 = 200.0 * np.array([0.0, -1.0])

T_1 + T_2 + T_3
```

Out[10]:
```
array([ 1.42108547e-14, -5.68434189e-14])
```

Problema 7

Encontre o produto vetorial $(2.0,\,3.0,\,-2.0)\times(-1.5,\,-1.0,\,2.0)$. Calcule também o ângulo entre os dois vetores através do produto escalar.

In [11]:
```
a = np.array([2.0, 3.0, -2.0])
b = np.array([-1.5, -1.0, 2.0])

print("O produto vetorial (a x b) é:")
print(np.cross(a,b))

# cos(θ) = a.b / (ab)
a_norm = np.linalg.norm(a)
b_norm = np.linalg.norm(b)
theta = np.arccos(np.inner(a,b) / (a_norm * b_norm))
print('O ângulo entre os vetores "a" e "b" é θ = {0:.2f}°'.format(theta * 180 / np.pi
```

```
O produto vetorial (a x b) é:
[ 4.  -1.   2.5]
O ângulo entre os vetores "a" e "b" é θ = 154.26°
```

Problema 8

Uma bola de futebol é pontapeada de modo a rodar sobre si própria, o que adiciona a força de Magnus às outras forças existêntes. A força de Magnus resulta do escoamento do ar ser diferente nos lados diametralmente opostos da bola.

Se a rotação for descrita pelo vetor $\boldsymbol{\omega}=(0,\,0,\,10)\;\mathrm{rad/s}$ e a velocidade for $\mathbf{v}=(0,\,1,\,0)\;\mathrm{m/s}$, calcule a força de Magnus, se for definida por

$$\mathbf{F}_{\text{mag}} = \frac{1}{2} A\, \rho_{\text{ar}}\, r\, \boldsymbol{\omega} \times \mathbf{v},$$

ond $A = \pi r^2$ é a área de secção de corte da bola, $r = 11$ cm é o raio da bola, e $\rho_{\text{ar}} = 1.225\ \mathrm{km/m}^3$ a massa volúmica do ar.

Notando que $\mathbf{v}$ e $\boldsymbol{\omega}$ são ortogonais e assentam no plano yz, podemos simplificar o produto externo, ficando assim o vetor $\mathbf{F}_{\text{mag}}$ ao longo da direção Ox. Além disso, assumindo a convenção de um sistema de eixos *baseado na mao direita*, percebemos que o produto externo $\boldsymbol{\omega} \times \mathbf{v}$ é resulta num vetor ao longo de $-Ox$, i.e., $\boldsymbol{\omega} \times \mathbf{v}$ é negativo.

$$\mathbf{F}_{\text{mag}} = -\frac{\pi r^3}{2} \rho_{\text{ar}}\, \omega\, v\, \hat{\mathbf{x}} = (-0.0256,\ 0,\ 0)\ \mathrm{N},$$

In [12]:
```
r = 0.11                              # raio da bola, m
A = np.pi * r ** 2                    # área de secção de corte da bola, m^2
rho = 1.225                           # massa volúmica do ar, kg/m^3

omega = np.array([0, 0, 10])          # velocidade angular, rad/s
v = np.array([0, 1, 0])               # velocidade de translação, m/s

# O produto externo é obtido apartir da função np.cross(a, b)
F_mag = 0.5 * A * rho * r * np.cross(omega, v)
print("O vetor força F_mag = ({0:.4f}, ".format(F_mag[0]), "{0:.0f}, ".format(F_mag[1
```

```
O vetor força F_mag = (-0.0256,  0,  0)
```