Uygulama Notları: 8.5

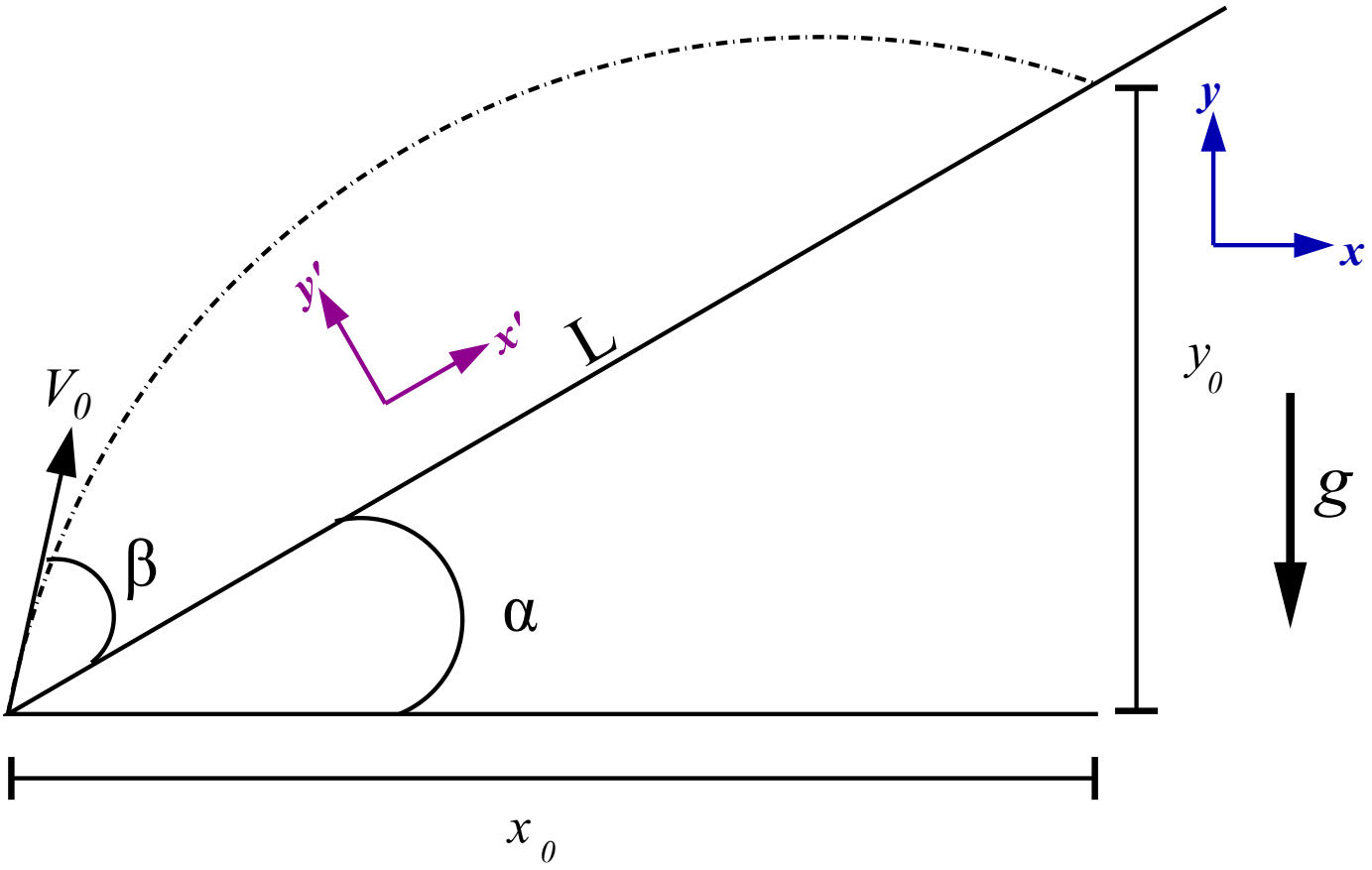
FİZ220 - Bilgisayar Programlama II | 29/05/2020

Lineer Cebir Uygulamaları & Grafik Çizimi (I)

Dr. Emre S. Taşcı, emre.tasci@hacettepe.edu.tr (mailto:emre.tasci@hacettepe.edu.tr)
Fizik Mühendisliği Bölümü
Hacettepe Üniversitesi

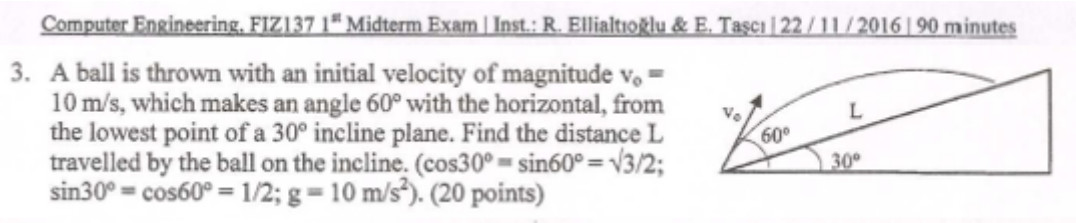
1. Eğik Düzleme Atılan Eğik Atış Problemi

(Neden?)



Yatayla α açısı yapan bir eğik düzlemin dibinden, eğik düzleme göre β açısı ve V_0 ilk hızıyla atılan bir cismin uçuş zamanını (t_0) ve eğik düzlem üzerine düştüğü yerin atış noktasına olan uzaklığını (L) bulunuz.

Bu soruyu, 2016 yılında, bilgisayar mühendisliği bölümüne, FIZ137 - Fizik I dersinin 1. ara sınavında sormuşuz (Recai Hoca'yla):


Computer Engineering, FIZ137 1st Midterm Exam | Inst.: R. Ellialtıoğlu & E. Taşcı | 22 / 11 / 2016 | 90 minutes

3. A ball is thrown with an initial velocity of magnitude v_o = 10 m/s, which makes an angle 60° with the horizontal, from the lowest point of a 30° incline plane. Find the distance L travelled by the ball on the incline. ($\cos 30° = \sin 60° = \sqrt{3}/2$; $\sin 30° = \cos 60° = 1/2$; g = 10 m/s^2). (20 points)

Çözüm

En genel haliyle (sorumuzdan bağımsız olarak), V_0 ilk hızıyla, yatayla θ açısı yapacak şekilde atılan bir cismin t_0 anındaki konumunu veren hareket denklemlerini yazarak başlayalım:

$$x_0 = V_0 \cos\theta\, t_0 + \frac{1}{2}\ddot{x}\, t_0^2 \quad (1)$$

$$y_0 = V_0 \sin\theta\, t_0 + \frac{1}{2}\ddot{y}\, t_0^2 \quad (2)$$

Rutin Koordinatlar

Bizim durumumuzda, soruya özel olarak, şu bilgiler elimizde:

$$\theta = \alpha + \beta$$

$$\ddot{x} = 0$$

$$\ddot{y} = -g$$

$$\tan\alpha = \frac{y_0}{x_0} \rightarrow y_0 = x_0 \tan\alpha \quad (*)$$

$$x_0 = L\cos\alpha \rightarrow L = \frac{x_0}{\cos\alpha} \quad (**)$$

Bunları kullanarak:

$$(1) \Rightarrow x_0 = V_0 \cos(\alpha+\beta)\, t_0 + \frac{1}{2}(0)\, t_0^2 = V_0 \cos(\alpha+\beta)\, t_0$$

$$(2) \Rightarrow y_0 = V_0 \sin(\alpha+\beta)\, t_0 + \frac{1}{2}(-g)\, t_0^2 = V_0 \sin(\alpha+\beta)\, t_0 - \frac{1}{2} g t_0^2$$

$$(*) \Rightarrow V_0 \sin(\alpha+\beta)\, t_0 - \frac{1}{2} g t_0^2 = \tan\alpha\, V_0 \cos(\alpha+\beta)\, t_0$$

$$\rightarrow \frac{1}{2} g t_0 = V_0 \sin(\alpha+\beta) - V_0 \tan\alpha \cos(\alpha+\beta)$$

t_0'ı yalnız bırakıp trigonometrik eşitlikleri çalıştıralım:

$$
\begin{aligned}
t_0 &= \frac{2V_0}{g}[\sin(\alpha+\beta) - \cos(\alpha+\beta)\tan\alpha] \\
&= \frac{2V_0}{g}\left[\sin\alpha\cos\beta + \sin\beta\cos\alpha - \frac{(\cos\alpha\cos\beta - \sin\alpha\sin\beta)\sin\alpha}{\cos\alpha}\right] \\
&= \frac{2V_0}{g\cos\alpha}\left[\sin\alpha\cos\alpha\cos\beta + \cos^2\alpha\sin\beta - \sin\alpha\cos\alpha\cos\beta + \sin^2\alpha\sin\beta\right] \\
&= \frac{2V_0}{g\cos\alpha}\left[\sin\beta(\cos^2\alpha + \sin^2\alpha)\right] \\
&= \frac{2V_0}{g}\frac{\sin\beta}{\cos\alpha}
\end{aligned}
$$

$$
\boxed{t_0 = \frac{2V_0}{g}\frac{\sin\beta}{\cos\alpha}}
$$

Artık elimizde t_0 olduğuna göre, önce x_0'ı:

$$
x_0 = V_0\cos(\alpha+\beta)\,t_0 = V_0\cos(\alpha+\beta)\frac{2V_0}{g}\left(\frac{\sin\beta}{\cos\alpha}\right)
$$

$$
x_0 = \frac{2V_0^2}{g}\frac{\cos(\alpha+\beta)\sin\beta}{\cos\alpha}
$$

x_0'ı kullanarak da $(**)$ eşitliğinden -nihayet- L'yi bulabiliriz:

$$
(**) \Rightarrow \boxed{L = \frac{x_0}{\cos\alpha} = \frac{2V_0^2}{g}\frac{\cos(\alpha+\beta)\sin\beta}{\cos^2\alpha}}
$$

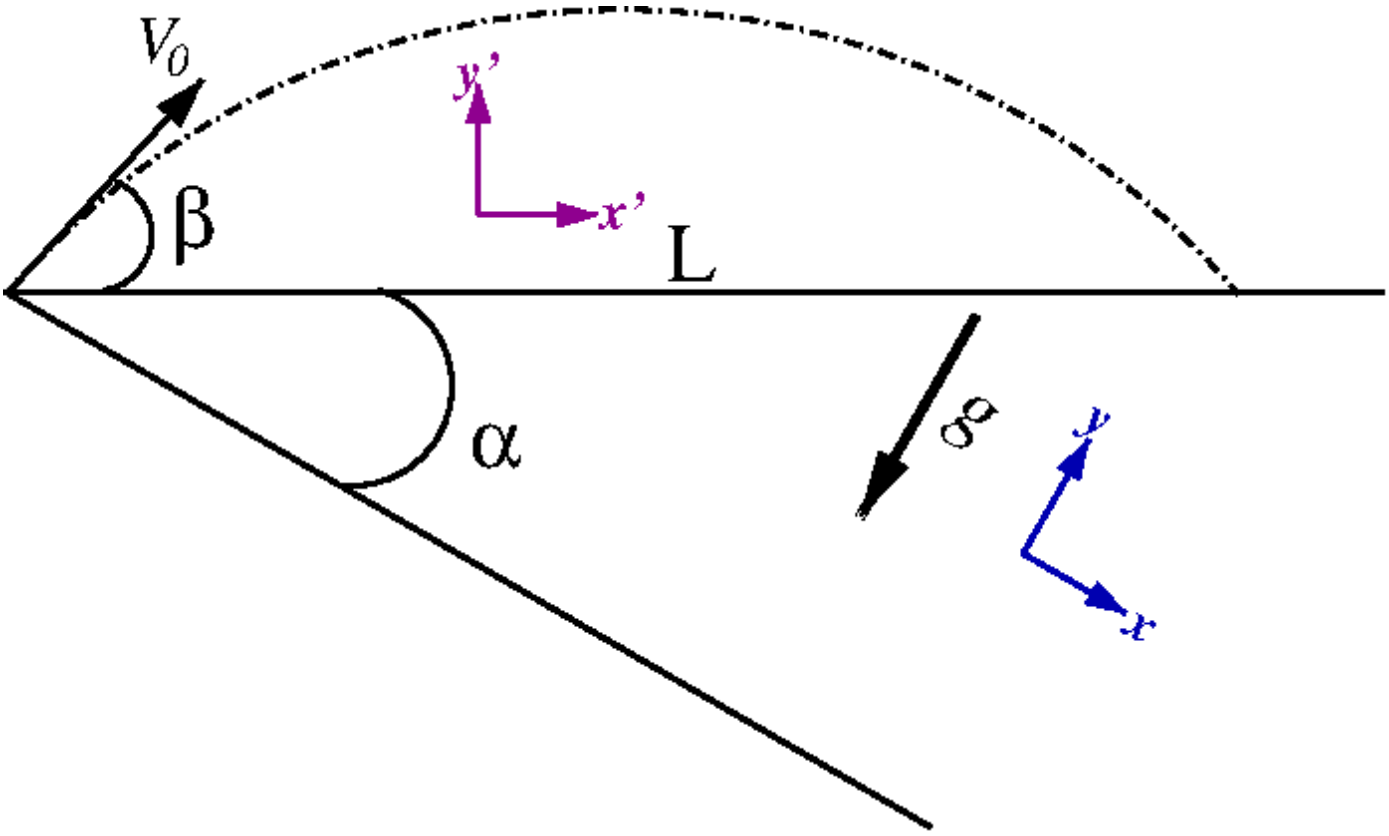
In [39]:

```
# Sorudaki değerleri yerine koyarsak:
import numpy as np
alpha = np.deg2rad(30) # derece -> radyan
beta = np.deg2rad(30) # derece -> radyan
V0 = 10 # m/s
g = 10 # m/s^2

t0 = (2*V0/g)*(np.sin(beta)/np.cos(alpha))
print("t0 = ",t0,"s")

#x0 = (2*V0**2/g)*(np.cos(alpha+beta)*np.sin(beta)/np.cos(alpha))
#print("x0 = ",x0," m")

L = (2*V0**2/g)*(np.cos(alpha+beta)*np.sin(beta)/np.cos(alpha)**2)
print(" L = ",L," m")
```

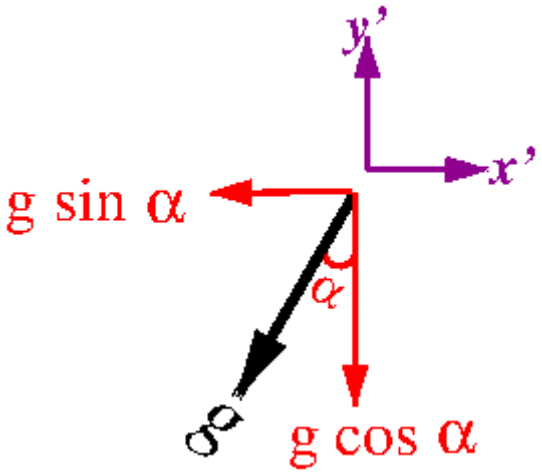
```
t0 =  1.1547005383792512 s
 L =  6.666666666666665  m
```

'Normal' Koordinatlar

Böyle çözünce size de biraz karışık gelmiyor mu? Bence çözmeden önce birazcık kafamızı çalıştırıp, daha "uygun" bir koordinat sistemi kullanalım: sorumuzun çiziminde morla işaretlenmiş olan, x'nün eğik düzlemle paralel olduğu (x', y') koordinat takımını. Bunun için de sistemimizi α kadar döndürüyoruz:



Bu yeni koordinat sisteminde, yerçekimi ivmesinin yeni eksenlere göre bileşenleri şu şekilde olacaktır:



(1) ve (2) genel hareket denklemlerini bu eksenler için yazmak için, bu koordinat sistemindeki özel durumları listeleyelim:

$$\theta = \beta$$

$$\ddot{x'} = -g \sin \alpha$$

$$\ddot{y'} = -g \cos \alpha$$

$$y'_0 = 0 \quad (*)$$

$$\tan \alpha' = \frac{y'_0}{x'_0} \rightarrow \tan \alpha' = 0 \leftrightarrow \alpha' = 0$$

$$x'_0 = L \cos \alpha' \rightarrow L = x'_0$$

Bunları kullanarak:

$$(1) \Rightarrow x_0' = V_0 \cos\beta\, t_0 + \frac{1}{2}(-g\sin\alpha)\, t_0^2 = V_0 \cos\beta\, t_0 - \frac{1}{2} g \sin\alpha\, t_0^2$$

$$(2) \Rightarrow y_0' = V_0 \sin\beta\, t_0 + \frac{1}{2}(-g\cos\alpha)\, t_0^2 = V_0 \sin\beta\, t_0 - \frac{1}{2} g \cos\alpha\, t_0^2$$

$$(*) \Rightarrow V_0 \sin\beta\, t_0 - \frac{1}{2} g \cos\alpha\, t_0^2 = 0$$

$$\rightarrow \frac{1}{2} g \cos\alpha\, t_0 = V_0 \sin\beta \Rightarrow \boxed{t_0 = \frac{2V_0}{g}\frac{\sin\beta}{\cos\alpha}}$$

[Bu noktada, yukarıdaki koordinat sisteminde bu aşamaya gelene kadar nelerle uğraştığımıza tekrardan bir bakmanızı rica ediyorum 8]

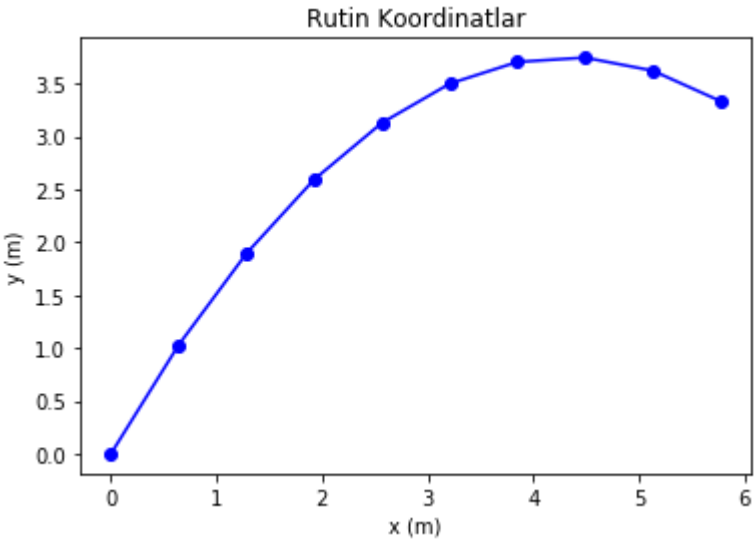
L'yi de hesaplayalım:

$$\begin{aligned}
L = x_0' &= V_0 \cos\beta\, t_0 - \frac{1}{2} g \sin\alpha\, t_0^2 \\
&= V_0 \cos\beta \left(\frac{2V_0}{g}\frac{\sin\beta}{\cos\alpha}\right) - \frac{1}{2} g \sin\alpha \left(\frac{2V_0}{g}\frac{\sin\beta}{\cos\alpha}\right)^2 \\
&= V_0 \cos\beta \left(\frac{2V_0}{g}\frac{\sin\beta}{\cos\alpha}\right) - \frac{1}{2} g \sin\alpha \left(\frac{2^2 V_0^2}{g^2}\frac{\sin^2\beta}{\cos^2\alpha}\right) \\
&= \frac{2V_0^2 \sin\beta}{g\cos\alpha}\left(\cos\beta - \frac{\sin\alpha\sin\beta}{\cos\alpha}\right) \\
&= \frac{2V_0^2 \sin\beta}{g\cos^2\alpha}(\cos\alpha\cos\beta - \sin\alpha\sin\beta) \\
&= \frac{2V_0^2}{g}\frac{\sin\beta\cos(\alpha+\beta)}{\cos^2\alpha}
\end{aligned}$$

Hazır elimiz değmişken, çizdirelim de bari...

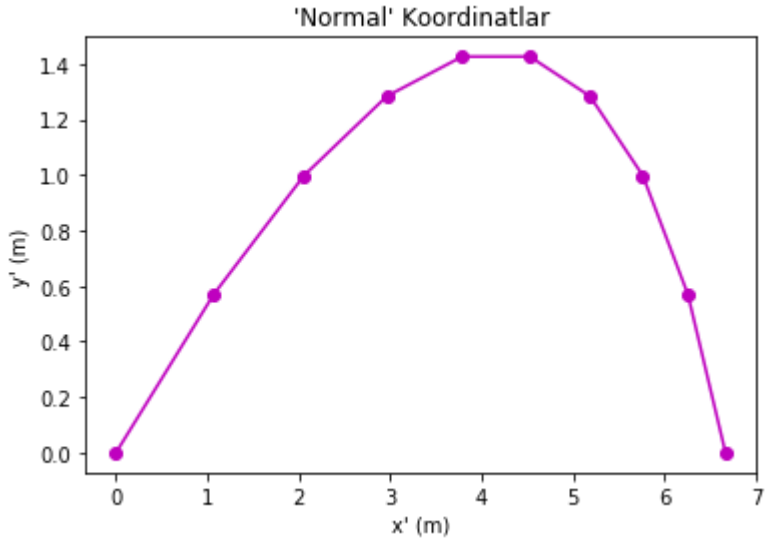
In [3]:

```
import numpy as np
import matplotlib.pyplot as plt

alpha = np.deg2rad(30) # derece -> radyan
beta = np.deg2rad(30) # derece -> radyan
V0 = 10 # m/s
g = 10 # m/s^2

t0 = (2*V0/g)*(np.sin(beta)/np.cos(alpha))

t = np.linspace(0,t0,10)
x = V0*np.cos(alpha+beta)*t
y = V0*np.sin(alpha+beta)*t - 0.5*g*t**2

plt.plot(x,y,"o-b")
plt.xlabel("x (m)")
plt.ylabel("y (m)")
plt.title("Rutin Koordinatlar")
#plt.xlim(0,8)
#plt.ylim(0,10)
plt.show()
```



In [34]:

```
import numpy as np
import matplotlib.pyplot as plt

alpha = np.deg2rad(30) # derece -> radyan
beta = np.deg2rad(30) # derece -> radyan
V0 = 10 # m/s
g = 10 # m/s^2

t0 = (2*V0/g)*(np.sin(beta)/np.cos(alpha))

t = np.linspace(0,t0,10)
xp = V0*np.cos(beta)*t - 0.5*g*np.sin(alpha)*t**2
yp = V0*np.sin(beta)*t - 0.5*g*np.cos(alpha)*t**2

plt.plot(xp,yp,"o-m")
plt.xlabel("x' (m)")
plt.ylabel("y' (m)")
plt.title("'Normal' Koordinatlar")
plt.show()
```



Ödev #1

Ölçeği ayarlayıp, iki grafikteki yayı aynı uzunluğa getirin.

Sonuç

Amerikalıların "ha öyle, ha böyle..." anlamına gelen bir deyimleri var (*tom-ay-to/to-mah-to* (https://en.wiktionary.org/wiki/tomayto,_tomahto) -- Gershwin'in şarkısının Fitzgerald & Armstrong tarafından yapılan müthiş yorumu içinse buyrunuz (https://www.youtube.com/watch?v=J2oEmPP5dTM)...). Sonuçta biz de sadece kafamızı çevirmekle, işlemlerin bazılarını kolaylaştırsak da, kafamızı döndürmek suretiyle evrensel yasalara müdahale edemediğimizden, atılan cisim, iki referans çerçevesinde de (görelilikle beraber, koordinat sistemi takımlarına "referans çerçevesi" gibi âfili isimler takmaktayız ;) aynı sürede havada kalıp, eğik düzlemin aynı yerine düştü, klasik fizikte bu zaten beklenen şeydi.

İki sistemdeki sonuçlarımızı özetlersek:

$$\begin{pmatrix} x \\ y \\ t_0 \\ L \end{pmatrix} = \begin{pmatrix} 5.7735 \\ 3.3333 \\ 1.1547 \\ 6.6667 \end{pmatrix}, \quad \begin{pmatrix} x' \\ y' \\ t_0 \\ L \end{pmatrix} = \begin{pmatrix} 6.6667 \\ 0 \\ 1.1547 \\ 6.6667 \end{pmatrix}$$

Acaba (x, y) setimize ne yaptık da (x', y')deki değerlere dönüştüler? Hatırladım, eksenleri döndürmüştük... O halde, dönüş matrisimizi ezberden bir tanımlayalım, bakalım ne olacak:

$$R_\alpha = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}; \ \vec{x} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

In [41]:

```
import numpy as np

alpha = np.deg2rad(30) # derece -> radyan

R_alpha = np.array([[np.cos(alpha),np.sin(alpha)],[-np.sin(alpha),np.cos(alpha)]])
x = np.array([[5.7735],[3.3333]])

xp = np.dot(R_alpha,x)
print(xp)
```

```
[[ 6.66664767e+00]
 [-2.75215653e-05]]
```

Yani:

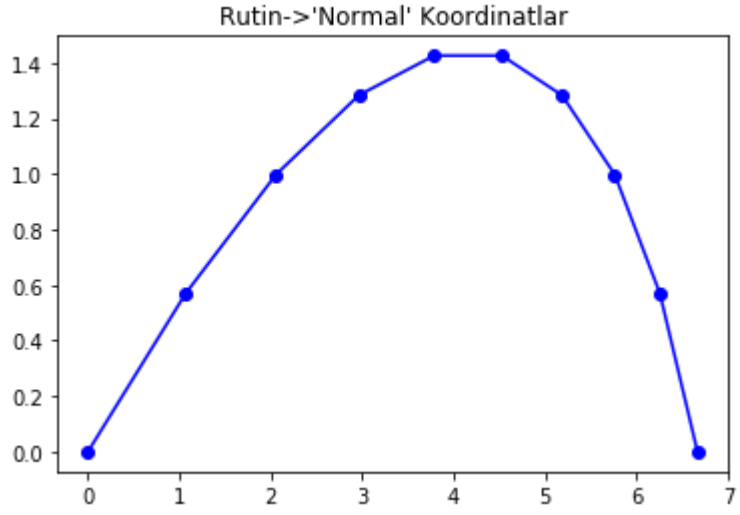
$$R_{\pi/6} . \vec{x} = \vec{x'}$$
$$\Updownarrow$$
$$\begin{pmatrix} \cos 30^o & \sin 30^o \\ -\sin 30^o & \cos 30^o \end{pmatrix} \begin{pmatrix} 5.7735 \\ 3.3333 \end{pmatrix} = \begin{pmatrix} 6.6667 \\ 0 \end{pmatrix}$$

Son söz

Demek ki, istesek tâ en başından bu çevrimi (ya da tersini) yapabilirmişiz:

In [4]:

```
import numpy as np
import matplotlib.pyplot as plt

alpha = np.deg2rad(30) # derece -> radyan
beta = np.deg2rad(30) # derece -> radyan
V0 = 10 # m/s
g = 10 # m/s^2

t0 = (2*V0/g)*(np.sin(beta)/np.cos(alpha))

t = np.linspace(0,t0,10)

# Rutin Koordinatlar
x = V0*np.cos(alpha+beta)*t
y = V0*np.sin(alpha+beta)*t - 0.5*g*t**2
xy = np.array([x,y])

# Dönüş matrisimiz
R_alpha = np.array([[np.cos(alpha),np.sin(alpha)],[-np.sin(alpha),np.cos(alpha)]])

# Rutin Koordinatlar -> 'Normal' Koordinatlar
xpyp=np.dot(R_alpha,xy)

plt.plot(xpyp[0,:],xpyp[1,:],"o-b")
#plt.xlabel("x'=R.xy0,: (m)")
#plt.ylabel("y'=R.xy1,: (m)")
plt.title("Rutin->'Normal' Koordinatlar")
#plt.xlim(0,8)
#plt.ylim(0,4)
plt.show()
```



Ödev #2

Bizim bildiğimiz -saatin tersi yönünde döndüren- dönüş matrisi:

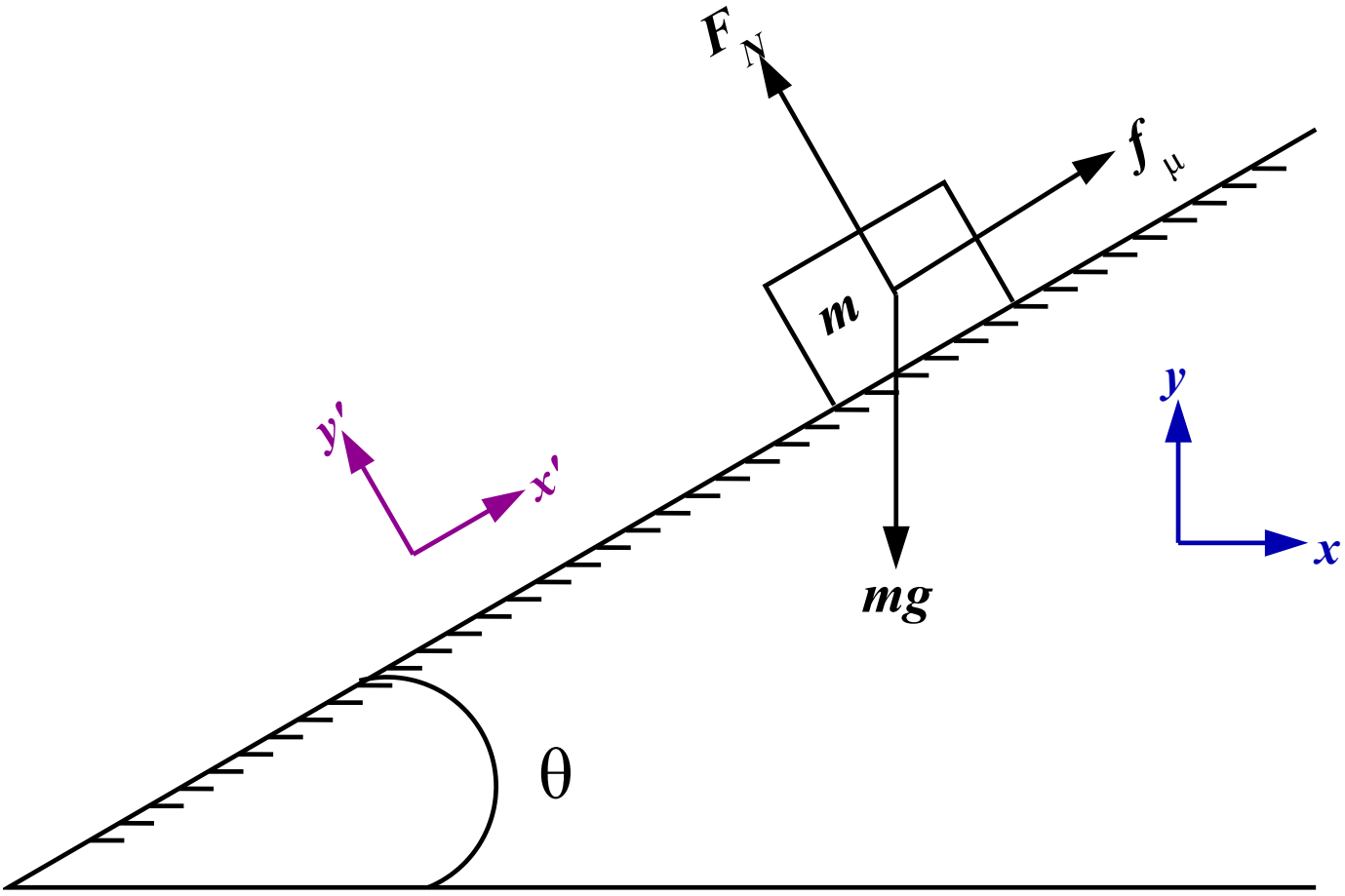
$$R_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

şeklinde değil miydi (yani "$-\sin\alpha$" sağ üstte; "$\sin\alpha$" sol altta olacak şekilde)?

Ama sistemimizi saatin tersi yönünde döndürmemize rağmen, neden yukarıda işlemi yaparken "$-\sin\alpha$"yı sol altta, "$\sin\alpha$"yı sağ üstte aldık?... (Açıklayın)

2. Sürtünmeli Eğik Düzlemde Kayan Kütle

(Nasıl?)



İkinci örneğimizden herkese merhabalar! Soru (*normal kuvveti ve cismin eğik düzlem üzerindeki L mesafeyi ne kadar sürede kat edeceğini hesaplayın:* $\theta = 30^o;\ \mu = 0.2;\ m = 1.2\,kg;\ L = 100\,m;\ g = 9.81\,m/s^2$) çok klasik, o kadar klasik ki, bilinçaltınızda çözmeye başladınız bile -- hatta durun an itibarı ile aklınızdan ne yaptığınızı da söyleyeyim: mg ağırlığını F_N normal kuvvet ve f_μ sürtünme kuvveti yönlerinde bileşenlerine ayırıyorsunuz!..

Rutin Koordinatlar:

Ama ayırmayalım hemen; öyle, körlemesine dalalım probleme, "vardır bir hikmeti" deyip, alışılageldik, yere paralel x-ekseni ve ona 90^o yukarı uzanan y-ekseninden oluşan klasik koordinat takımımızı kullanıp, F_N ile f_μ'yü bu eksenlerdeki bileşenlerine ayırıp, Newton'ın 2. yasasını tatbik edelim:

x-ekseni boyunca kuvvetler:

$$f_\mu \cos\theta - F_N \sin\theta = m\ddot{x} \quad (1)$$

y-ekseni boyunca kuvvetler:

$$f_\mu \sin\theta + F_N \cos\theta - mg = m\ddot{y} \quad (2)$$

Seçtiğimiz koordinat sistemine dair bağıntı:

$$\tan\theta = \frac{y}{x} \rightarrow y = x\tan\theta \Rightarrow \ddot{y} = \ddot{x}\tan\theta \quad (3)$$

Genel olarak bildiğimiz:

$f_\mu = \mu F_N \quad (4)$

(4)'ü (1) ve (2)'de yerine yazalım:

$$(1): \ \mu F_N \cos\theta - F_N \sin\theta = m\ddot{x}$$
$$\rightarrow m\ddot{x} + (\sin\theta - \mu\cos\theta) F_N = 0 \quad (1')$$

$$(2): \ \mu F_N \sin\theta + F_N \cos\theta - mg = m\ddot{y}$$
$$\rightarrow -m\ddot{y} + (\cos\theta + \mu\sin\theta) F_N = mg \quad (2')$$

$$(3): \ \ddot{y} = \ddot{x}\tan\theta$$
$$\rightarrow \tan\theta\,\ddot{x} - \ddot{y} = 0 \quad (3')$$

Bu üç bilinmeyenli $(\ddot{x}, \ddot{y}, F_N)$ üç denklemimizi $(1'), (2'), (3')$, matris - vektör çarpımı olarak yazabiliriz:

$$\begin{pmatrix} m & 0 & \sin\theta - \mu\cos\theta \\ 0 & -m & \cos\theta + \mu\sin\theta \\ \tan\theta & -1 & 0 \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ F_N \end{pmatrix} = \begin{pmatrix} 0 \\ mg \\ 0 \end{pmatrix}$$

Bu lineer sistemi nasıl çözebileceğimizi biliyoruz (değil mi? ;)

In [25]:

```
import numpy as np

# Soruda verilenler:
theta = np.deg2rad(30) # derece -> radyan
mu = 0.41
m = 1.2 # kg
L = 100 # m
g = 9.81 # m/s^2

A = np.array([[m,0,np.sin(theta)-mu*np.cos(theta)],
              [0,-m,np.cos(theta)+mu*np.sin(theta)],
              [np.tan(theta),-1,0]])
b = np.array([0,m*g,0])

s = np.linalg.solve(A,b)
print(s)
```

```
[-1.23127961 -0.71087961 10.19485105]
```

Yani:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ F_N \end{pmatrix} = \begin{pmatrix} -1.231\ m/s^2 \\ -0.711\ m/s^2 \\ 10.195\ N \end{pmatrix}$$

(ivmelerin negatif olması, sola ve aşağı doğru ivmelendiğini, yani yönünü gösteriyor).

Eğik düzlemdeki L mesafesini ne kadar zamanda kat edeceğine gelince:

$$x_0 = L\cos\theta = \frac{1}{2}\ddot{x}t_0^2 \rightarrow t_0 = \sqrt{\frac{2L\cos\theta}{\ddot{x}}}$$

In [20]:

```
t0 = (np.sqrt(2*L*np.cos(theta)/np.abs(s[0])))
print("t0: ",t0,"s")
```

```
t0:  11.86047194749622 s
```

'Normal' Koordinatlar

Bir de, bize mantıklı gelen koordinat sistemi olan (x', y') üzerinden gidelim: Ne de olsa bir öncekinde hem normal kuvveti, hem de sürtünme kuvvetini bileşenlerine ayırmak zorunda kalmıştık, burada sadece ağırlığı bileşenlere ayıracağız; y ekseni yönünde herhangi bir yerdeğiştirme olmaması da cabası! $(\ddot{y}' = 0)$

x'-ekseni boyunca kuvvetler:

$$f_\mu - mg\sin\theta = m\ddot{x}' \quad (1)$$

y'-ekseni boyunca kuvvetler:

$$F_N - mg\cos\theta = m\ddot{y}' \quad (2)$$

Seçtiğimiz koordinat sistemine dair bağıntı:

$$\ddot{y}' = 0 \quad (3)$$

Genel olarak bildiğimiz:

$$f_\mu = \mu F_N \quad (4)$$

Bu denklemleri kullanıp, bu koordinat sisteminde hareket denklemlerimizi lineer denklem seti olarak temsil edersek:

$$\begin{pmatrix} m & 0 & -\mu \\ 0 & m & -1 \\ 0 & 220 & 0 \end{pmatrix} \begin{pmatrix} \ddot{x}' \\ \ddot{y}' \\ F_N \end{pmatrix} = \begin{pmatrix} -mg\sin\theta \\ -mg\cos\theta \\ 0 \end{pmatrix}$$

Ödev #3

En alt satırın orta sütunundaki "220" değeri nereden geldi? Dersimizin koduyla aynı oluşu tesadüf mü?

Çözüme geçersek:

In [22]:

```
import numpy as np

# Soruda verilenler:
theta = np.deg2rad(30) # derece -> radyan
mu = 0.41
m = 1.2 # kg
L = 100 # m
g = 9.81 # m/s^2

A = np.array([[m,0,-mu],[0,m,-1],[0,220,0]])
b = np.array([-m*g*np.sin(theta),-m*g*np.cos(theta),0])

su = np.linalg.solve(A,b)
print(su)
```

```
[-1.42175922  0.         10.19485105]
```

Yani:

$$\begin{pmatrix} \ddot{x}' \\ \ddot{y}' \\ F_N \end{pmatrix} = \begin{pmatrix} -1.422\ m/s^2 \\ 0\ m/s^2 \\ 10.195\ N \end{pmatrix}$$

Eğik düzlemdeki L mesafesini ne kadar zamanda kat edeceğine gelince:

$$x'_0 = L = \frac{1}{2}\ddot{x}' t_0^2 \rightarrow t_0 = \sqrt{\frac{2L}{\ddot{x}'}}$$

In [24]:

```
t0 = (np.sqrt(2*L/np.abs(su[0])))
print("t0: ",t0,"s")
```

```
t0:  11.86047194749622 s
```

Sonuç

Yine şaşırtıcı olmayan bir biçimde, bizim seçtiğimiz koordinatlardan bağımsız olarak cismin üzerine binen normal kuvvet de, verilen mesafeyi kat ettiği zaman da iki referans çerçevesinde aynı çıktı.

İlk sorudan sonra, birinden diğerine $(\ddot{x}, \ddot{y}) \leftrightarrow (\ddot{x}', \ddot{y}')$ nasıl geçebileceğimizi artık tahmin edebilirsiniz:

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

Hemen bir teyit alalım:

In [28]:

```
import numpy as np

theta = np.deg2rad(30)

xydd = np.array([-1.23127961, -0.71087961])

R_theta = np.array([[np.cos(theta),np.sin(theta)],[-np.sin(theta),np.cos(theta)]])

xyddpp = np.dot(R_theta,xydd)
print(xyddpp)
```

```
[-1.42175923e+00  3.70762565e-09]
```

Welcome to Sadet... (popülasyon 2i+3 $\in \mathbb{C}$)

Sadede gelirsek, görünürde bu ikinci örnekte ilk örneği biraz(!) daha geliştirdik, bilgimizi pekiştirdik ama aslında iki örnek birbirinden çok farklı (en büyük farkları da 1. örneği matris-vektör çarpımı şeklinde yazamamızda çünkü orada aktif olarak rol oynayan zaman ve/veya onu aradan çıkarıp x ile y değişkenlerinin aralarındaki bağıntı doğrusal değil, 2. dereceden (*quadratic*): hal böyle olunca pirincin taşını ayıklayamıyoruz maalesef (yine aynı nedenden ötürü oturup soruyu ellerimizle çözdük (başka (bir dolu) sayısal yaklaşımla tabii ki çözebiliriz ama lineer cebrin dışına çıkmış olurduk o zaman)). Bu soruda ise her şey gayet net:

$$\begin{pmatrix} m & 0 & \sin\theta - \mu\cos\theta \\ 0 & -m & \cos\theta + \mu\sin\theta \\ \tan\theta & -1 & 0 \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ F_N \end{pmatrix} = \begin{pmatrix} 0 \\ mg \\ 0 \end{pmatrix} \quad (^*)$$

Biz dan-dan-dan çözerken, arka planda, belki de farkına bile varmadan neler yaptığımıza gelirsek: koordinat dönüşümünü $(\ddot{x}, \ddot{y})$'ye nasıl etki ettirip, $(\ddot{x}', \ddot{y}')$ gidebileceğimizi gördük:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \ddot{x}' \\ \ddot{y}' \end{pmatrix} \quad (^*)$$

Boyut uyumu açısından F_N'i de değişmeyecek şekilde ekleyelim:

$$\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ F_N \end{pmatrix} = \begin{pmatrix} \ddot{x}' \\ \ddot{y}' \\ F_N \end{pmatrix} \quad (^{**})$$

Bu yukarıdaki işlem aslında gayet ciddi bir matris eşitliği, sembolik olarak şöyle bir şey:

$$A \cdot \vec{x} = \vec{x}'$$

İki tarafı da soldan, matrisimizin tersi A^{-1} ile çarparsak:

$$\underbrace{A^{-1} \cdot A}_{\mathbb{1}} \cdot \vec{x} = A^{-1}\vec{x}'$$

$$\rightarrow \vec{x} = A^{-1}\vec{x}'$$

$(\ddot{x}, \ddot{y})$'den $(\ddot{x}', \ddot{y}')$'ye gidebiliyorduk, artık $(\ddot{x}', \ddot{y}')$'den $(\ddot{x}, \ddot{y})$'ye de gidebiliyoruz bu sayede! Bu iş için dönüş matrisimizin tersini bilmek gerekiyor. Dilerseniz bilgisayara hesaplatabilirsiniz (`R_inv = np.linalg.inv(R)`) ama kafamızı kullanmak daha iyi: eğer eksenlerimi saatin tersi yönünde θ açısı kadar döndürmek için:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

matrisini kullanıyorsam, geri gelmek için (yani işlemi "tersine" almak için), bu sefer saat yönünde θ kadar gitmeliyim: bir başka deyişle, $-\theta$ kadar dönüş yapmalıyım. Kosinüs çift fonksiyon olduğu için, θ yerine $-\theta$ yazdığımda bir şey değişmeyecek $(\cos(-\theta) = \cos(\theta))$ ama sinüs tek fonksiyon olduğundan, işaretler değişecek $(\sin(-\theta) = -\sin(\theta))$. Demek ki:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}^{-1} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Bunu da (**) eşitliğinin iki yanına soldan etki ettirdiğimizde:

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

(*) eşitliğinde $\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix}$ gördüğümüz yere bunu yazalım:

$$\begin{pmatrix} m & 0 & \sin\theta - \mu\cos\theta \\ 0 & -m & \cos\theta + \mu\sin\theta \\ \tan\theta & -1 & 0 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \ddot{x}' \\ \ddot{y}' \\ F_N \end{pmatrix} = \begin{pmatrix} 0 \\ mg \\ 0 \end{pmatrix}$$

Kısa bir ara

İşimiz neredeyse bitiyor, çok az kaldı. Özdeğer ve özvektörlerde hatırlıyorsanız, köşegenleştirme yapıyorduk, yani `[ozdegerler,ozvektorler] = np.linalg.eig(A)` dediğimizde, örneğin:

In [39]:

```
A = np.array([[2.,3.],[4.,5.]])
print(A)
print()
[ozdegerler,ozvektorler] = np.linalg.eig(A)
print("Özdeğerler: ",ozdegerler,"\n-----\nÖzvektörler:\n",ozvektorler)
```

```
[[2. 3.]
 [4. 5.]]

Özdeğerler:  [-0.27491722  7.27491722]
-----
Özvektörler:
 [[-0.79681209 -0.49436913]
 [ 0.60422718 -0.86925207]]
```

Özdeğerler köşegenleştirilmiş matris (D), özvektör matrisi de (u) olarak alındığında, $u \cdot D \cdot u^{-1} = A$ eşitliği gerçeklenir. Devam edersek:

In [44]:

```
D = np.diag(ozdegerler)
print("D:",D)

u = ozvektorler.copy()
print("u:",u)

u_inv = np.linalg.inv(u)

print("u.D.u^{-1} = ",np.linalg.multi_dot((u,D,u_inv)))
```

```
D: [[-0.27491722  0.        ]
 [ 0.          7.27491722]]
u: [[-0.79681209 -0.49436913]
 [ 0.60422718 -0.86925207]]
u.D.u^{-1} =  [[2. 3.]
 [4. 5.]]
```

Şimdi tekrar örneğimize geri dönelim, nerede kalmıştık?

$$\begin{pmatrix} m & 0 & \sin\theta - \mu\cos\theta \\ 0 & -m & \cos\theta + \mu\sin\theta \\ \tan\theta & -1 & 0 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \ddot{x}' \\ \ddot{y}' \\ F_N \end{pmatrix} = \begin{pmatrix} 0 \\ mg \\ 0 \end{pmatrix}$$

Bu, kesinlikle bir özdeğer denklemi değil -- soldaki vektör ile sağdaki vektör bambaşka. Ama yine de hazır iki matris, bir de vektör çarpımı bulmuşken,

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

matrisine u^{-1} muamelesi çekelim, eşitliğin iki yanını da soldan bunun tersi ile çarpalım:

$$\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} m & 0 & \sin\theta - \mu\cos\theta \\ 0 & -m & \cos\theta + \mu\sin\theta \\ \tan\theta & -1 & 0 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \ddot{x}' \\ \ddot{y}' \\ F_N \end{pmatrix}$$
$$= \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ mg \\ 0 \end{pmatrix}$$

Sağ taraftaki çarpımı kafadan yapabilirim ama sol taraftaki çarpımı kaldırmaya kalbim yetmez. Neyse, hele bir sağ tarafı halledelim, sola da döneriz elbet:

$$\begin{pmatrix} & \ddots & \end{pmatrix} \cdot \begin{pmatrix} \ddot{x}' \\ \ddot{y}' \\ F_N \end{pmatrix} = \begin{pmatrix} mg\sin\theta \\ mg\cos\theta \\ 0 \end{pmatrix}$$

Sayısal olarak da olsa, elimizdeki değerlerle sol tarafı yapalım:

In [62]:

```
print("m:",m,"\tmu:",mu,"\ttheta:",np.rad2deg(theta))

u = np.array([[np.cos(theta),np.sin(theta),0],[-np.sin(theta),np.cos(theta),0],[0,0
u_inv = np.linalg.inv(u)
A = np.array([[m,0,np.sin(theta)-mu*np.cos(theta)],[0,-m,np.cos(theta)+mu*np.sin(th
Q = np.linalg.multi_dot((u,A,u_inv))
print()
print(Q)
```

```
m: 1.2  mu: 0.41        theta: 29.999999999999996

[[ 6.00000000e-01 -1.03923048e+00  6.61025404e-01]
 [-1.03923048e+00 -6.00000000e-01  8.55070416e-01]
 [ 1.11022302e-16 -1.15470054e+00  0.00000000e+00]]
```

$(\ddot{x}', \ddot{y}', F_N)$ değerlerini yukarıda hesaplamıştık (su) olarak. Q matrisimizle $(\ddot{x}', \ddot{y}', F_N)$ vektörünü çarpıp, sonucu kontrol edelim:

In [67]:

```
print("  su:",su)
print("Q.su:",np.dot(Q,su))
print(m*g*np.sin(theta),m*g*np.cos(theta),0)
```

```
  su: [-1.42175922  0.         10.19485105]
Q.su: [ 5.88600000e+00  1.01948511e+01 -1.57846983e-16]
5.885999999999999 10.194851053350412 0
```

Bir de bizim o zaman bulmuş olduğumuz değerlerle kıyaslayalım:

$$\begin{pmatrix} m & 0 & -\mu \\ 0 & m & -1 \\ 0 & 220 & 0 \end{pmatrix} \begin{pmatrix} \ddot{x}' \\ \ddot{y}' \\ F_N \end{pmatrix} = \begin{pmatrix} -mg\sin\theta \\ -mg\cos\theta \\ 0 \end{pmatrix}$$

In [63]:

```
K=np.array([[m,0,-mu],[0,m,-1],[0, 220,0]])
print(K)
print(np.dot(K,su))
```

```
[-1.42175922  0.         10.19485105]
[ 5.88600000e+00  1.01948511e+01 -1.57846983e-16]
5.885999999999999 10.194851053350412
[[  1.2    0.    -0.41]
 [  0.     1.2   -1.  ]
 [  0.   220.     0.  ]]
[ -5.886      -10.19485105   0.        ]
```

Son söz

İşin özü, koordinat dönüşümüyle sadece "koordinatları" değil, bütün denklemleri dönüştürüyor olmamız.