# Cinemática de un Disco Rodando

Ángel Piñeiro. Física I

September 30, 2022

## 1 Cinemática Disco que rueda sin deslizar

**Objetivo:** Describir el movimiento de un disco que rueda sin deslizar sobre una plataforma horizontal.

**Procedimiento:** El movimiento del centro de masas del disco viene dado por la condición de rodadura sin deslizamiento:

$$\vec{r}_C(t) = \theta\, R\, \hat{i} = \omega\, t\, R\, \hat{i}$$

con el sistema de referencia fijo situado en el centro del disco en el instante inicial y con el movimiento en la dirección del eje $x$. La velocidad se calcula derivando la anterior ecuación:

$$\vec{V}_C(t) = \dot{\theta}\, R\, \hat{i} = \omega\, R\, \hat{i}$$

La posición de un punto fijo del perímetro del disco viene dado por la siguiente ecuación:

$$\vec{r}_P(t) = \vec{r}_C(t) + R\, cos(\theta)\, \hat{i} + R\, sen(\theta)\, \hat{j}$$

Si imponemos la condición de movimiento plano general (traslación + rotación) en un sólido rígido:

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i \Rightarrow \vec{r}_j = \vec{r}_i + \vec{r}_{ij} \Rightarrow \vec{v}_j = \vec{v}_i + \vec{v}_{ij} \Rightarrow \boxed{\vec{v}_j = \vec{v}_i + \vec{\omega} \times \vec{r}_{ij}}$$

obtenemos tanto la posición como la velocidad de cualquier punto del perímetro del disco en función del tiempo.

**Cálculo de la Base:** Lugar geométrico de los puntos del espacio en los que $\vec{v} = 0$

Utilizando la ecuación anterior, tomando el eje del disco como punto de referencia, e igualándola a cero:

$$0 = \vec{v}_C + \vec{\omega} \times \vec{r}_{Ci} \Rightarrow$$

Los puntos de la base vienen dados por la ecuación:

$$\vec{r}_B = \theta\, R\, \hat{i} - R\, \hat{j}$$

que son los puntos de la recta sobre la que se apoya el disco.

**Ruleta:** Puntos de la base expresados en un sistema de referencia localizado en el disco. Escogemos un sistema de referencia en el centro del disco y lo expresamos en función del sistema de referencia externo.

$$\hat{i} = cos(\theta)\, \hat{i}' + sen(\theta)\, \hat{j}'$$

$$\hat{j} = -sen(\theta)\, \hat{i}' + cos(\theta)\, \hat{j}'$$

Restando la posición del centro del disco y sustituyendo $\hat{i}$, $\hat{j}$, los puntos de la ruleta vendrían dados por:

$$\vec{r}'_B = -R\,(-sen(\theta)\,\hat{i}' + cos(\theta)\,\hat{j}')$$

que representan una circunferencia expresada en el sistema de coordenadas interno:

$$x'^2 + y'^2 = R^2$$

**NOTA:** Si el disco deslizase (o derrapase) la ecuación que describiría el movimiento de su centro sería:

$$\vec{r}_C(t) = a\,\theta\,R\,\hat{i}$$

donde la constante a sería mayor que cero en caso de deslizamiento y menor que cero en caso de derrapamiento. Además, la variación del ángulo podría ser una función no lineal del tiempo. ¿ Cómo crees que afectarían estos factores al movimiento y al cálculo de la base y de la ruleta?

[ ]:
```
from vpython import *
import numpy as np
import sympy as sp
sp.init_printing() # output formateado en latex
import sympy.physics.vector as spv
import IPython.display as disp

run = False
def runbutton(b): # Se llama a esta función cuando se hace click en el botón de␣
 ↪ejecutar
    global run
    if run: b.text = 'Ejecutar' # b es el botón
    else: b.text = 'Pausa'
    run = not run
```

[ ]:
```
omega=-3 # velocidad angular en rad/s. <0 => giro a la derecha
R=6 # radio del disco
tmax=6 # tiempo máximo de la simulación

t = sp.symbols('t') # tiempo
S= spv.ReferenceFrame('S') # Sistema de coordenadas para representar los vectores

#Definimos las distancias y sus derivadas
rC=-45*S.x-omega*t*R*S.x
radial=R*(sp.cos(omega*t)*S.x+sp.sin(omega*t)*S.y)
rP=rC+radial
vC=rC.diff(t,S)
vP=rP.diff(t,S)
aP=vP.diff(t,S)

#imprimimos las funciones
disp.display(disp.Math(r'\vec{r_C}(t)= '), rC)
disp.display(disp.Math(r'\vec{r_P}(t)= '), rP)
disp.display(disp.Math(r'\vec{v_P}(t)= '), vP)
disp.display(disp.Math(r'\vec{a_P}(t)= '), aP)
```

[ ]:
```
escena1 = canvas(background=color.white, autoscale=False, width=1000,height=200)

tiempo=0 # inicializamos el tiempo
```

```
#Calculamos las componentes de los vectores
rCx=spv.dot(rC, S.x).evalf(subs={t: tiempo})
rCy=spv.dot(rC, S.y).evalf(subs={t: tiempo})
vCx=spv.dot(vC, S.x).evalf(subs={t: tiempo})
vCy=spv.dot(vC, S.y).evalf(subs={t: tiempo})
rPx=spv.dot(rP, S.x).evalf(subs={t: tiempo})
rPy=spv.dot(rP, S.y).evalf(subs={t: tiempo})
vPx=spv.dot(vP, S.x).evalf(subs={t: tiempo})
vPy=spv.dot(vP, S.y).evalf(subs={t: tiempo})

#base = box(canvas=escena1, pos=vec(0,-0.2-R,0), size=vec(50,0.2,10), texture=textures.
 ↪wood) # Asociamos la plataforma a la variable "base"
disco=extrusion(path=[vec(0,0,0), vec(0,0,-0.5)],color=color.orange,
    shape=[shapes.circle(radius=R), shapes.line(start=(0,0), end=(0,R-0.01), np=20,␣
 ↪thickness=0.2)],
                pos=vec(rCx,rCy,0), align='left')

# Dibujamos el eje de rotación y colocamos el vector velocidad y un vector radial␣
 ↪hacia un punto fijo del perímetro
eje=cylinder(canvas=escena1,pos=disco.pos+vec(0,0,1),axis=vector(0,0,-2), radius=1,␣
 ↪color=color.red, make_trail=True, emissive=True)
eje.v=vector(vCx,vCy,0)
attach_arrow(eje, 'v', scale=0.2, color=color.blue, shaftwidth=0.5) # vector velocidad␣
 ↪en azul para representación

# Dibujamos un punto del perímetro del disco y le pegamos su vector velocidad
borde=cylinder(canvas=escena1,pos=vec(rPx,rPy,0.5),axis=vector(0,0,-1), radius=0.2,␣
 ↪color=color.red, make_trail=True, emissive=True)
borde.v=vec(vPx,vPy,0)
attach_arrow(borde, 'v', scale=0.2, color=color.blue, shaftwidth=0.5) # vector␣
 ↪velocidad en azul para representación

button(text='Ejecutar', bind=runbutton) # Dibujamos el botón de ejecución

base=graph(title='Curva Polar Fija (Base)', width=450, height=350,
                  xmin=0, #xmax=30, #ymin=-R-1, ymax=R+1,
                  align='left', xtitle='<i>x</i>', ytitle='<i>y</i>')
ruleta=graph(title='Curva Polar Móvil (Ruleta)', width=450, height=350,
                xmin=-R-1, xmax=R+1, ymin=-R-1, ymax=R+1,
                align='right', xtitle="<i>x\'</i>", ytitle='<i>y\'</i>')

b = gcurve(color=color.black, legend=True, label="", graph=base)
r = gcurve(color=color.black, legend=True, label="", graph=ruleta)

dt=0.05
for i in range(0,int(tmax/dt)):
    rate(10)
    if run:
        tiempo=tiempo+dt # Actualizamos el tiempo
        ang=omega*dt
        #
        rCx=spv.dot(rC, S.x).evalf(subs={t: tiempo})
```

```
        rCy=spv.dot(rC, S.y).evalf(subs={t: tiempo})
        vCx=spv.dot(vC, S.x).evalf(subs={t: tiempo})
        vCy=spv.dot(vC, S.y).evalf(subs={t: tiempo})
        rPx=spv.dot(rP, S.x).evalf(subs={t: tiempo})
        rPy=spv.dot(rP, S.y).evalf(subs={t: tiempo})
        vPx=spv.dot(vP, S.x).evalf(subs={t: tiempo})
        vPy=spv.dot(vP, S.y).evalf(subs={t: tiempo})
        #
        b.plot(-omega*tiempo*R, -R)
        r.plot(R*sp.sin(omega*tiempo), R*sp.cos(omega*tiempo))
        disco.pos=vec(rCx,rCy,0)
        disco.rotate(angle=ang, axis=vec(0,0,1), origin=disco.pos)
        eje.pos=disco.pos+vec(0,0,+1)
        borde.pos=vec(rPx,rPy,0.5)
        borde.v=vec(vPx,vPy,0)
```