Taller de LaTeX

Autor 1 Autor 2

5 de noviembre de 2022

Índice

Resumen

La mecánica cuántica a la rama de la física contemporánea dedicada al estudio de los objetos y fuerzas de muy pequeña escala espacial, es decir, de la materia a nivel del átomo y de las partículas que lo componen, así como los movimientos que las caracterizan.

Abstract

Quantum mechanics is the branch of contemporary physics dedicated to the study of objects and forces of very small spatial scale, that is, matter at the level of the atom and the particles that compose it, as well as the movements that characterize them.

Parte I

Introducción al LaTeX

SECCIÓN 1.

Escritura

La teoría de la relatividad especial, también llamada teoria de la relatividad restringida, es una teoría de la física publicada en 1905 por Albert Einstein. Surge de la observación de que la velocidad de la luz en el vacío es igual en todos los sistemas de referencia inerciales y de obtener todas las consecuencias del principio de *relatividad de Galileo*.

SECCIÓN 2.

Que es LaTeX

Su código abierto permitió que muchos usuarios realizasen nuevas utilidades que extendiesen sus capacidades con objetivos muy variados, a veces ajenos a la intención con la que fue creado: aparecieron diferentes dialectos de LaTeX que, a veces, eran incompatibles entre sí. Para atajar este problema, en 1989 Lamport y otros desarrolladores iniciaron el llamado «Proyecto LaTeX3». En el otoño boreal de 1993 se anunció una reestandarización completa de LaTeX, mediante una nueva versión que incluía la mayor parte de estas extensiones adicionales (como la opción para escribir transparencias o la simbología de la American Mathematical Society) con el objetivo de dar uniformidad al conjunto y evitar la fragmentación entre versiones incompatibles de LaTeX 2.09.

SECCIÓN 3.

Relatividad

Según el, cualquier experimento realizado en un sistema de REFERENCIA INERCIAL se desarrollara de manera idéntica en cualquier otro sistema inercial. Información que aparecerá en la página.

1. Primera sesión de LaTeX.
2. Semana libre
3. Segunda sesión de LaTeX.
    A. Primera sesión de LaTeX.
    B. Segunda sesión de LaTeX.
        a) Primera sesión de LaTeX.
        b) Segunda sesión de LaTeX.
            i. Primera sesión de LaTeX.
            ii. Segunda sesión de LaTeX.
            iii. Tercer sesión de LaTeX.
        c) Tercer sesión de LaTeX.
    C. Tercer sesión de LaTeX.
4. Tercer sesión de LaTeX.

- Primera sesión de LaTeX.
- Señale verdadero o falso.

  - Primer enunciado
  - ⋆ Segundo enunciado
  - Tercer enunciado

- Tercer sesión de LaTeX.

Tengo 35 $, ø

$$\boxed{P = 50\,\text{ATM}}$$

$$P = 50\ \text{atm} \tag{1}$$

3.1. Tipos de escritura

Párrafo

- Si yo uso `textbf`{texto}, lo que este dentro se pondrá en negrita:
  "**texto**"
- Si yo uso `textit`{texto}, lo que este dentro se pondrá en cursiva:
  "*texto*"
- Si yo uso `textsc`{texto}, lo que este dentro se pondrá todo en mayuscula:
  "TEXTO"
- Si yo uso `textsf`{texto}, lo que este dentro se pondrá todo en mayuscula:
  "texto"

- Si yo uso `textsl{texto}`, lo que este dentro se pondrá todo en mayuscula ss: "*texto*"

Escritura de formulas

Existen tres formas de escribir formulas:

- **Escritura lineal**: Para ello solamente se deben utilizan un símbolo de dolar en cada extremo ($).

  Ella no te ama $m = \frac{y_2 - y_1}{x_2 - x_1}$, me fue infiel

- **Escritura centrada**: Para ello se deben utilizan doble símbolo de dolar en cada extremo ($$).

  No olvidar que la función de onda se escribe:

  $$\Psi(r, \theta, \phi) = R(r)\Theta(\theta, \phi)$$

  La parte angular tiene solución con **armónicos esféricos**

- **Escritura enumerada**: Para ello debemos usar el comando `begin{equation}`

  Según $\cdots$, la ecuación fundamental de la termodinámica es:

  $$T\mathrm{d}S = \mathrm{d}U + p\mathrm{d}V - \mu\mathrm{d}N \tag{2}$$

  $$\mathrm{d}S = \frac{\mathrm{d}U}{T} + \frac{p\mathrm{d}V}{T} - \frac{\mu\mathrm{d}N}{T}$$

SECCIÓN 4.

Matrices

Segun De La Peña [1] y Muñoz [2]

Vectores

$$\vec{v} = \mathbf{v} = \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ \vdots \\ a_{1n} \end{pmatrix}$$

$$\vec{v} = \mathbf{v} = \begin{vmatrix} a_{11} \\ a_{12} \\ a_{13} \\ \vdots \\ a_{1n} \end{vmatrix}$$

Matrices

$$A = \begin{pmatrix} a_{11} & a_{21} & a_{31} & \dots & a_{m1} \\ a_{12} & a_{22} & a_{32} & \dots & a_{m2} \\ a_{13} & & & & \\ \vdots & & & & \\ a_{1n} & & & & \end{pmatrix}$$

Como se pudo ver en la pagina 7 indicamos las formas es escribir las ecuaciones. Y la solución final la pudimos ver en la pagina 6. La lista de asistencia aparece haciendo click aqui

Parte II

Teoría del Caos

SECCIÓN 5.

Atractor de Lorentz

Como una cuestión previa conviene aclarar conceptos porque tiende a confundirse Caos y Fractales (ver figura 2). En artículos de divulgación y en muchas publicaciones vienen juntos y mezclados, por lo que hay que precisar que Caos y Fractales no son sinónimos y tienen comportamientos distintos a pesar de compartir una formulación sencilla y que ciertos fenómenos caóticos tengan una estructura fractal como es el caso del atractor de Lorenz que podemos observar en la figura 1 visto en la pagina 10.

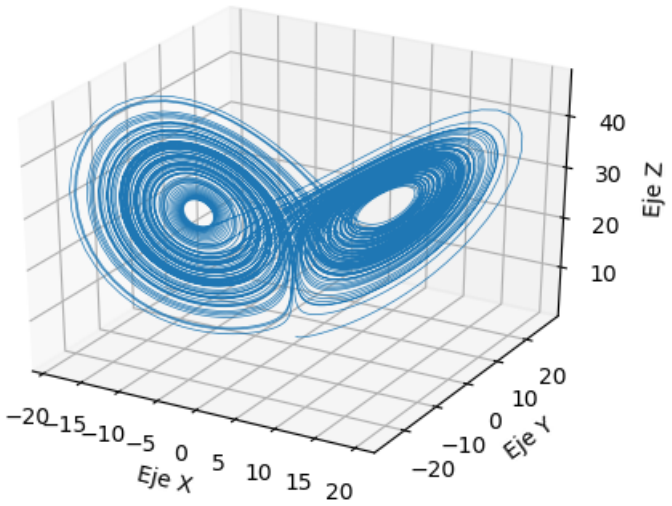
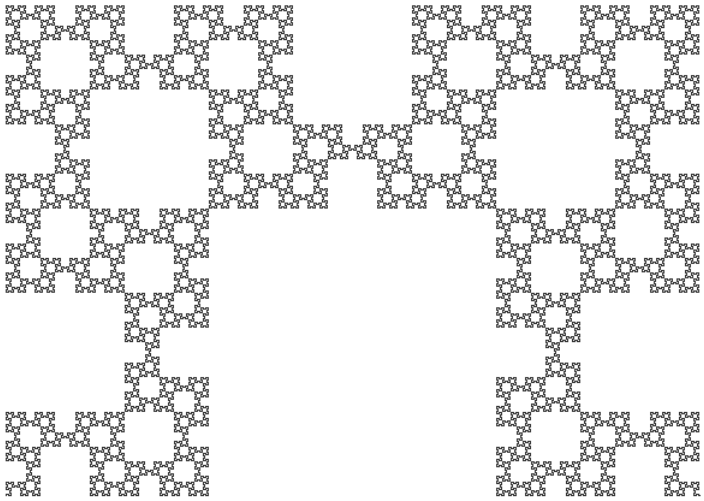


Figura 1: Atractor de Lorentz

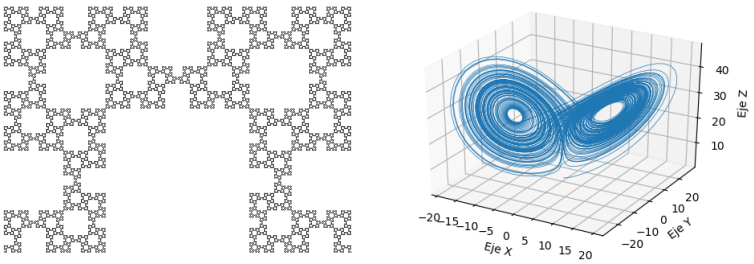
Los estudios de Edward Lorenz sobre "Dependencia sensitiva de las condiciones

inicialesz el famoso efecto mariposa son el origen de la Teoría del Caos.

Figura 2: Fractal de Fibonacci. Fuente: `https://rosettacode.org/wiki/Fibonacci_word/fractal`



El término atractor extraño se debe a David Ruelle y Floris Takens, físico-matemático el primero y matemático el segundo. Lo definieron como: una zona bien delimitada del espacio de fases en la que las líneas de la trayectoria del sistema nunca se cortan. Líneas de longitud infinita confinadas en área finita, describiendo órbitas no periódicas. Ni Ruelle ni Takens lo habían visto nunca, pero presagiaron su existencia. Ese monstruo matemático, según ellos, debería de ser fractal.



(a) Primera figura (b) Segunda figura

Figura 3: Gráficas importantes en la teoria del caos

Sea la tabla 1 podemos observar los implementos necesarios para poder restaurar nuestro laboratorio de Química General.

Tabla 1:

Instrumentos			
Items		Inventario	
Materiales	**Precio**	**Cantidad**	**Salón**
Probeta	70	5	KFC-1
Matraz	26	13	KFC-2
Pipeta	30	3	KFC-3

Implementos para un laboratorio de Química I

SECCIÓN 6.

Exponente de Lyapunov

El Exponente Lyapunov o Exponente característico Lyapunov de un sistema dinámico es una cantidad que caracteriza el grado de separación de dos trayectorias infinitesimalmente cercanas. Cuantitativamente, dos trayectorias en el espacio-fase con separación inicial δZ_0 diverge.

El radio de separación puede ser distinto para diferentes orientaciones del vector de separación inicial.

Aunque, hay un completo espectro del exponente Lyapunov; el número de ellos es igual al número de dimensiones del espacio-fase. Es común referirse sólo a la más grande, porque determina la predictibilidad de un sistema. Los exponentes característicos de Lyapunov (LCE del inglés Lyapunov Characteristic Exponents) son una herramienta que permite cuantificar la velocidad a la que se separan dos órbitas con condiciones iniciales infinitamente cer-

canas. Por ello, con frecuencia se emplean como indicadores de la presencia de caos.

Parte III

Teoremas en LaTeX

SECCIÓN 7.

Cajas

Para poder crear cajas en LaTeX se debe incluir dentro de una minipagina, por ejemplo en este caso:

El radio de separación puede ser distinto para diferentes orientaciones del vector de separación inicial.

Teorema 7.1. *Si una función g*

SECCIÓN 8.

Teorema

Teorema 8.1. *Si una función f*

Corolario 8.1.1. *dsad*

Lema 8.2. *fjgjf*

Definición 8.1. dads

Comentario. xcvxb

Demostración. Sea $\|x^2\|$, en un $\langle\varphi|\psi\rangle$ □

Corolario 8.2.1. *There's no right rectangle whose sides measure 3cm, 4cm, and 6cm.*

Parte IV

Insertar código en LaTeX

SECCIÓN 9.

Paquete *listings*

Insertar tu propio código utilizando el paquete *listings*

Este paquete ordena los códigos con el siguiente entorno, el cual permite tambien introducir el código en el mismo documento:

```
%Personalizacion
\newtheorem{theorem}{Teorema}[section]
\newtheorem{corollary}{Corolario}[theorem]
\newtheorem{lemma}[theorem]{Lema}
\theoremstyle{definition}
\newtheorem{definition}{Definicion}[section]
\theoremstyle{remark}
\newtheorem*{remark}{Comentario}
```

Pero también podemos incluir el propio archivo, previamente colocado en la misma carpeta de origen, sin necesidad de tener que escribir nada mas que el siguiente comando:

```
from numpy import sin, cos
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
# Constantes utilizadas
G=9.81
```

```
7  R=0.045
8  M=0.19
9  H=0.15
10 I1=0.0026
11 I3=0.0001
12 def derivs(state, t):
13  dydx = np.zeros_like(state)
14  dydx[0] = state[1]
15  dydx[1]=(I3*state[3]*state[0]*np.cos(state[2])+I3*state[3]*state
       [5]-2*I1*state[3]*state[0]*np.cos(state[2]))/(I1*np.sin(state
       [2]))
16  dydx[2] = state[3]
17  dydx[3]=(state[1]**2)*state[3]*np.sin(state[2])*np.cos(state[2])-(
       I3/I1)*((state[1]**2)*state[3]*np.sin(state[2])*np.cos(state[2])
       +state[1]*state[3]*state[5]*np.sin(state[2]))+state[3]*((M*G*H*
       np.sin(state[2]))/(I1))
18  dydx[4] = state[5]
19  dydx[5]=state[3]*state[0]*np.sin(state[2])-((I3*state[3]*state[0]*
       np.cos(state[2])+I3*state[3]*state[5]-2*I1*state[3]*state[0]*np.
       cos(state[2]))/(I1*np.sin(state[2])))*np.cos(state[2])
20  return dydx
21 dt = 0.0001
22 t = np.arange(0.0, 5, dt)
23
24 # th1, th2, th3 son los angulos de Euler iniciales (grados)
25 # w1, w2, w3 son las velocidades angulares iniciales (grados por
       segundo)
26 th1=0.001
27 w1=0.001
28 th2=0.001
```

```
29 w2=0.001
30 th3=0.001
31 w3=0.001
32 # Condiciones iniciales
33 state = np.radians([th1,w1,th2,w2,th3,w3])
34 y = integrate.odeint(derivs, state, t)
35
36 # Energia total del sistema
37 E=(I1*(y[:,1]*np.sin(y[:,2])**2+y[:,3]**2))/2+(I3*((y[:,1]**2)*((np
       .cos([y[:,2]]))**2)+(y[:,5]**2)+2*y[:,1]*y[:,5]*np.cos(y[:,2])))
       /(2)+M*G*H*np.cos(y[:,2])
38
39 plt.subplot(2,2,1)
40 plt.plot(t,y[:, 4],'-c')
41 plt.grid()
42 plt.xlabel('t')
43 plt.ylabel('Phi')
44 plt.title('Phi(t) vs tiempo')
45 plt.subplot(2,2,2)
46 plt.plot(t,y[:, 5],'-r')
47 plt.grid()
48 plt.xlabel('t')
49 plt.ylabel('ww3')
50 plt.title('Velocidad angular vs tiempo')
51 plt.subplot(2,2,3)
52 plt.plot(y[:, 4],y[:, 5],'-y')
53 plt.grid()
54 plt.xlabel('Phi')
55 plt.ylabel('ww3')
56 plt.title('Velocidad angular vs Angulo')
```

```
plt.subplot(2,2,4)
plt.plot(t,E,'-c')
plt.grid()
plt.xlabel('t')
plt.ylabel('E')
plt.title('Energia vs tiempo')


plt.tight_layout()
plt.show()
```

Dependiendo del tipo de código en el que estemos trabajando será posible indicar este para darle una cierta personalización.

```
from numpy import sin, cos
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
# Constantes utilizadas
G=9.81
R=0.045
M=0.19
H=0.15
I1=0.0026
I3=0.0001
def derivs(state, t):
 dydx = np.zeros_like(state)
 dydx[0] = state[1]
 dydx[1]=(I3*state[3]*state[0]*np.cos(state[2])+I3*state[3]*state
    [5]-2*I1*state[3]*state[0]*np.cos(state[2]))/(I1*np.sin(state
    [2]))
 dydx[2] = state[3]
```

```
17  dydx[3]=(state[1]**2)*state[3]*np.sin(state[2])*np.cos(state[2])-(
      I3/I1)*((state[1]**2)*state[3]*np.sin(state[2])*np.cos(state[2])
      +state[1]*state[3]*state[5]*np.sin(state[2]))+state[3]*((M*G*H*
      np.sin(state[2]))/(I1))
18  dydx[4] = state[5]
19  dydx[5]=state[3]*state[0]*np.sin(state[2])-((I3*state[3]*state[0]*
      np.cos(state[2])+I3*state[3]*state[5]-2*I1*state[3]*state[0]*np.
      cos(state[2]))/(I1*np.sin(state[2])))*np.cos(state[2])
20  return dydx
21 dt = 0.0001
22 t = np.arange(0.0, 5, dt)
23
24 # th1, th2, th3 son los angulos de Euler iniciales (grados)
25 # w1, w2, w3 son las velocidades angulares iniciales (grados por
      segundo)
26 th1=0.001
27 w1=0.001
28 th2=0.001
29 w2=0.001
30 th3=0.001
31 w3=0.001
32 # Condiciones iniciales
33 state = np.radians([th1,w1,th2,w2,th3,w3])
34 y = integrate.odeint(derivs, state, t)
35
36 # Energia total del sistema
37 E=(I1*(y[:,1]*np.sin(y[:,2])**2+y[:,3]**2))/2+(I3*((y[:,1]**2)*((np
      .cos([y[:,2]]))**2)+(y[:,5]**2)+2*y[:,1]*y[:,5]*np.cos(y[:,2])))
      /(2)+M*G*H*np.cos(y[:,2])
38
```

```
plt.subplot(2,2,1)
plt.plot(t,y[:, 4],'-c')
plt.grid()
plt.xlabel('t')
plt.ylabel('Phi')
plt.title('Phi(t) vs tiempo')
plt.subplot(2,2,2)
plt.plot(t,y[:, 5],'-r')
plt.grid()
plt.xlabel('t')
plt.ylabel('ww3')
plt.title('Velocidad angular vs tiempo')
plt.subplot(2,2,3)
plt.plot(y[:, 4],y[:, 5],'-y')
plt.grid()
plt.xlabel('Phi')
plt.ylabel('ww3')
plt.title('Velocidad angular vs Angulo')
plt.subplot(2,2,4)
plt.plot(t,E,'-c')
plt.grid()
plt.xlabel('t')
plt.ylabel('E')
plt.title('Energia vs tiempo')


plt.tight_layout()
plt.show()
```

Y también poder indicar cierto sector del código:

```
# Condiciones iniciales
```

```
state = np.radians([th1,w1,th2,w2,th3,w3])
y = integrate.odeint(derivs, state, t)

# Energia total del sistema
E=(I1*(y[:,1]*np.sin(y[:,2])**2+y[:,3]**2))/2+(I3*((y[:,1]**2)*((np
    .cos([y[:,2]]))**2)+(y[:,5]**2)+2*y[:,1]*y[:,5]*np.cos(y[:,2])))
    /(2)+M*G*H*np.cos(y[:,2])
```

Se puede usar los siguientes tipos de lenguajes:

ABAP, ACSL, Ada, Algol, Ant, Assembler, Awk, bash, Basic, C#, C++, C, Caml, Clean, Cobol, Comal, csh, Delphi, Eiffel, Elan, erlang, Euphoria, Fortran, GCL, Go (golang), Gnuplot, Haskell, HTML, IDL, inform, Java, JVMIS, ksh, Lisp, Logo, Lua, make, Mathematica, Matlab, Mercury, MetaPost, Miranda, Mizar, ML, Modelica, Modula-2, MuPAD, NASTRAN, Oberon-2, Objective C, OCL, Octave, Oz, Pascal, Perl, PHP, PL/I, Plasm, POV, Prolog, Promela, Python, R, Reduce, Rexx, RSL, Ruby, S, SAS, Scilab, sh, SHELXL, Simula, SQL, tcl, TeX, VBScript, Verilog, VHDL, VRML, XML, XSLT.

Ejemplos de codificaciones utilizando el paquete *listings*

Tambien podemos utilizar otro tipo de lenguajes:

Código en Scilab

```
clear()
r=2 //coordenda generalizada
h=4 //velocidad generalizada

```

```
m2=0.5 //Masa del carro
m1=0.2 //Masa del pendulo
g=9.81 //Gravedad
l=0.25 //Longitud del pendulo
u=0.5 //Fuerza externa

Tf=25 //Tiempo final
pi=3.14159265359

function dx=F(t,x)
    dx=[x(3);x(4);(u+m1*l*sin(x(2))*(x(4).^2)-m1*g*sin(x(2))*cos(x
    (2)))/(m2+m1-m1*(cos(x(2)))^2);(u*cos(x(2))-(m1+m2)*g*sin(x(2))+
    m1*l*sin(x(2))*cos(x(2))*x(4).^2)/(m1*l*(cos(x(2)))^2-(m1+m2)*l)
    ]
endfunction

t0=0;
t=0:0.001:Tf;
x1=ode([0.001;1.00001;0;0],t0,t,F);
x2=ode([0.002;1.00002;0;0],t0,t,F)

d0=sqrt((x1(r,1)-x2(r,1)).^2+(x1(h,1)-x2(h,1)).^2)
l(1)=0
for i=2:length(t)
    d=sqrt((x1(r,i)-x2(r,i)).^2+(x1(h,i)-x2(h,i)).^2)
    l(1,i)=d
    l(2,i)=log(d/d0)/t(i)
end

b=x2(r,:).*l(r,:)  //Angulo estable
```

```
32
33 figure(1)
34 xgrid
35 xlabel('tiempo (seg)')
36 ylabel('$\theta$')
37 plot(t,b,'r-') //Angulo estable
38
39 figure(2)
40 xgrid
41 xlabel('tiempo (seg)')
42 ylabel('$\lambda$')
43 plot(t,l(r,:),'b-') //Exponente de Liapunov
```

Código en Fortran

```
1     PROGRAM ADAMS_MOULTON
2     REAL(4) X(100),Y(100),N
3     WRITE(*,*)' SOLUCION DE EDO POR EL METODO ADAMS-MOULTON '
4     WRITE(*,*)''
5     WRITE(*,*)' INGRESE LAS CONDICIONES INICIALES'
6     WRITE(*,*)' INGRESA X(0),Y(0),XF,N'
7     READ(*,*) X(0),Y(0),TI,N
8
9     H=(TI-X(0))/N
10
11    WRITE(*,*)'VALOR DE H:', H
12
13    do i=0,N
14        X(i+1)=X(i)+H
15    end do
16
```

```
17      WRITE(*,*)'RESULTADOS'
18      WRITE(*,*)' X Y '
19      WRITE(*,10) X(0),Y(0)
20      !euler
21      do J=0,1
22          Y(J+1)=Y(J)+H*F(X(J),Y(J))
23          WRITE(*,10) X(J+1),Y(J+1)
24
25      end do
26      !bashforth
27      A=Y(2)
28      do K=2,N-1
29          Y(K+1)=Y(K)+(H/24)*(9*F(X(K+1),A)+19*F(X(K),Y(K))-5*F(X(K
    -1),Y(K-1))+F(X(K-2),Y(K-2)))
30          WRITE(*,10)X(K+1),Y(K+1)
31      END DO
32
33      DO I=1,M
34          ER=ABS(XP(K2,I)-XP(K2+1,I))
35          IF (ER.GT.TOL)THEN
36              K2=K2+1
37              GOTO 10
38          ELSE
39          END IF
40      END DO
41
42  10  FORMAT (2(F9.4),2X)
43  END
44
45  FUNCTION F(X,Y)
```

```
46 F=-12*X*Y+Y*Y
47 return
48 END
```

Código en Java

```
1  #pragma strict
2
3  var origin : Transform; // What is considered the origin to the
       camera
4  var zoom : float;
5  var zoomMin : float = -5;
6  var zoomMax : float = 5;
7  var seekTime = 1.0;
8  var smoothZoomIn = false;
9  private var defaultLocalPosition : Vector3;
10 private var thisTransform : Transform;
11 private var currentZoom : float;
12 private var targetZoom : float;
13 private var zoomVelocity : float;
14
15 function Start()
16 {
17   // Cache component instead of looking it up every frame
18   thisTransform = transform;
19
20   // The default position is the position that is set in the editor
21   defaultLocalPosition = thisTransform.localPosition;
22
23   // Default the current zoom to what was set in the editor
24   currentZoom = zoom;
```

```
25 }
26
27 function Update()
28 {
29   // The zoom set externally must still be within the min-max range
30   zoom = Mathf.Clamp( zoom, zoomMin, zoomMax );
31
32   // Only collide with non-Player (8) layers
33   var layerMask = ~((1 << 8) | (1 << 2));
34
35   var hit : RaycastHit;
36   var start = origin.position;
37   var zoomedPosition = defaultLocalPosition + thisTransform.parent.
      InverseTransformDirection( thisTransform.forward * zoom );
38   var end = thisTransform.parent.TransformPoint( zoomedPosition );
39
40   // Cast a line from the origin transform to the camera and find
      out if we hit anything in-between
41   if ( Physics.Linecast( start, end, hit, layerMask ) )
42   {
43     // We hit something, so translate this to a zoom value
44     var position = hit.point + thisTransform.TransformDirection(
     Vector3.forward );
45     var difference = position - thisTransform.parent.TransformPoint
     ( defaultLocalPosition );
46     targetZoom = difference.magnitude;
47   }
48   else
49     // We didn't hit anything, so the camera should use the zoom
     set externally
```

```
50      targetZoom = zoom;
51
52    // Clamp target zoom to our min-max range
53    targetZoom = Mathf.Clamp( targetZoom, zoomMin, zoomMax );
54
55    if ( !smoothZoomIn && ( targetZoom - currentZoom ) > 0 )
56    {
57      // Snap the current zoom to our target if it is closer. This is
        useful if
58      // some object is between the camera and the origin
59      currentZoom = targetZoom;
60    }
61    else
62    {
63      // Smoothly seek towards our target zoom value
64      currentZoom = Mathf.SmoothDamp( currentZoom, targetZoom,
       zoomVelocity, seekTime );
65    }
66
67    // Set the position of the camera
68    zoomedPosition = defaultLocalPosition + thisTransform.parent.
       InverseTransformDirection( thisTransform.forward * currentZoom )
       ;
69    thisTransform.localPosition = zoomedPosition;
70  }
```

Código en LaTeX

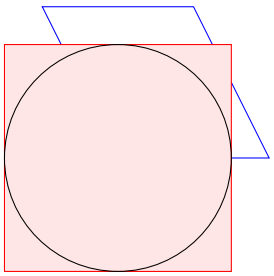
```
1  \section{Teorema}
2  \begin{theorem}
3  Si una funcion $f$
```

```
\end{theorem}

\begin{corollary}
dsad
\end{corollary}

\begin{lemma}
fjgjf
\end{lemma}

\begin{definition}
dads
\end{definition}

\begin{remark}
xcvxb
\end{remark}
```

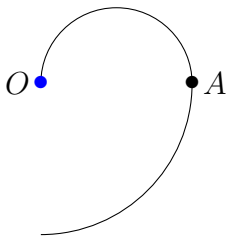
Listing 1: Código en LaTeX

Crear figuras con Tikz

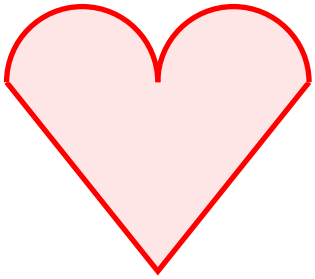
Podemos realizar ciertas gráficas en LaTeX con ayuda del paquete *tikz*, el cual podremos ver los siguientes resultados:



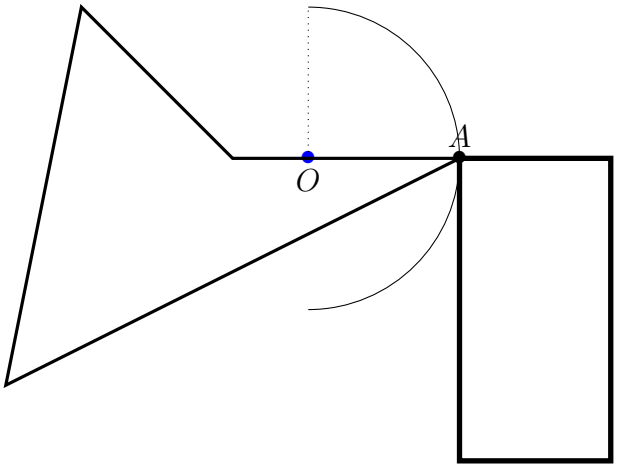
También podemos hacer nuestros propios arcos:



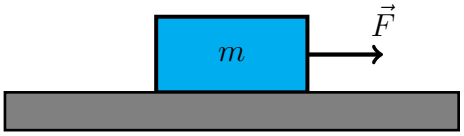
Otro ejemplo:



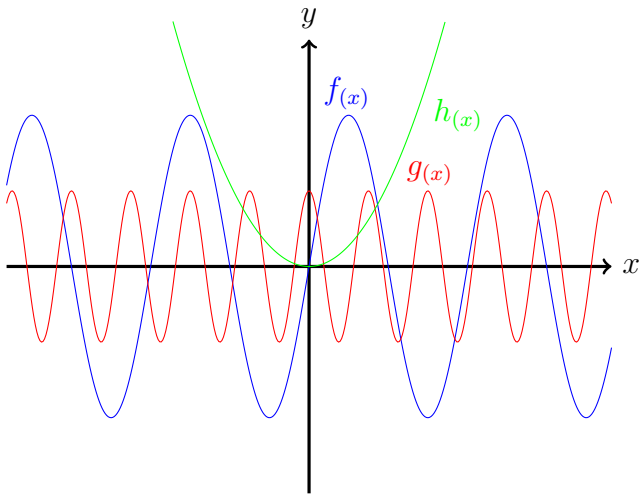
Si vamos a trabajar varias veces con el mismo punto, es recomendable asignarle ya una coordenada en especifico.



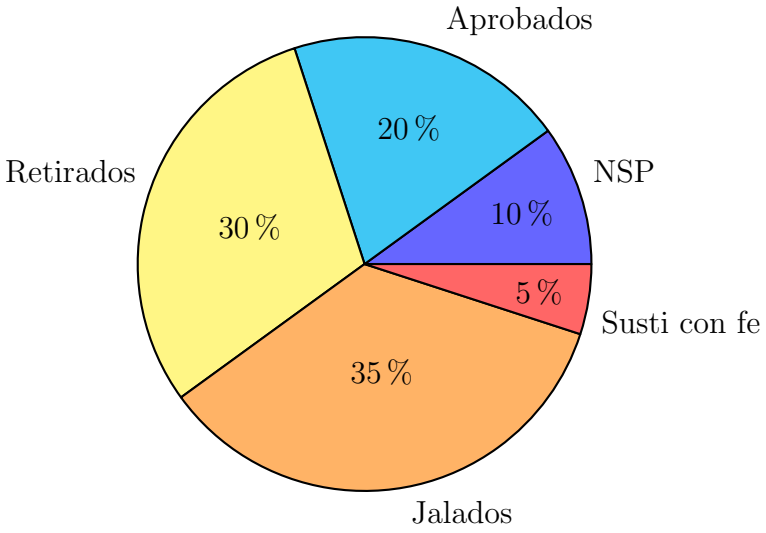
Tambien podemos trazar vectores:



Tikz también nos permite poder graficar ciertas funciones con una cantidad especifica de valores.



Tambien podemos establecer graficas estadisticas:



Bibliografía

[1] De La Peña, L. (2014). Introducción a la mecánica cuántica. Fondo de Cultura económica.

[2] Muñoz, J. M. (2015). Mecánica cuántica y libre albedrío: cinco cuestiones fundamentales. Principia: an international journal of epistemology, 19(1), 65-92.