Teoría de Estabilidad de las Soluciones de las Ecuaciones Diferenciales Ordinarias

Martínez Cerda Mario Antonio

A 19 de Marzo de 2021

1 Introducción

La teoría de estabilidad se refiere a la estabilidad de las soluciones de ecuaciones diferenciales ó trayectorias de u sistema dinámico bajo pequeñas perturbaciones de las condiciones iniciales.

La Teoría Cualitativa de Ecuaciones Diferenciales y Sistemas Dinámicos se enfoca en las propiedades asintóticas de las soluciones y sus trayectorias cuando el tiempo tiende a infinito. El ejemplo más sencillo de este tipo de comportamiento son los puntos de equilibrio o puntos fijos y las órbitas periódicas.

Se define un punto de equilibrio "x" para la ecuación diferencial

$$\frac{dx}{dt} = f(t,x)$$

si $f(t,x) = 0$ para todo tiempo t.

Cuando la función f no depende explícitamente del tiempo t, $f(x,(t))$, se dice que el sistema de ecuaciones es un sistema autónomo.

Los puntos de equilibrio o puntos críticos $(f(x(t)) = 0)$, se pueden clasificar de acuerdo a los signos de los eigenvalores de la linearización de la ecuación respecto a los puntos de equilibrio. Esto es, se evalúa la matriz Jacobiana de la ecuación en cada punto de equilibrio y se buscan los eigenvalores. El comportamiento de la solución del sistema en la vecindad de cada punto de equilibrio puede ser determinado cualitativamente.

Un punto de equilibrio es hiperbóloico si ninguno de sus eigenvalores tienen parte real cero. Si todos los eigenvalores tienen parte real negativa, el punto de equilibrio es estable. Si al menos un eigenvalor tiene parte real negativa y al menos uno tiene parte real positiva, el punto de equilibrio se le conoce como punto silla.

Estudiaremos los problemas de valor inicial en el caso de los sistemas autónomos.

$\frac{dx}{dt} = Ax$ con $x(0) = x_0$. Donde A es una matriz cuadrada de dimensiones nxn.

Concretamente, en este documento veremos la resolución de la actividad 9, aplicando lo ya mencionado sobre los métodos para solucionar este tipo de ecuaciones diferenciales, apreciando la estabilidad de cada una de ellas.

2 Ejercicios y evidencias

Por favor grafique en el espacio fase una familia de soluciones y determine el tipo de punto crítico de cada uno de los siguientes sistemas de ecuaciones.

2.1 Ejercicio 1.

$$\frac{dx}{dt} = y$$
$$\frac{dy}{dt} = -x$$

```
[ ]  #Realizamos la matriz:
     A = np.array([[0,1], [-1,0]])
     print("A = ", A)
```

```
A =  [[ 0  1]
 [-1  0]]
```

```
[ ]  #Traza de la matriz
     p = np.trace(A)
     print("Tr(A) = p = ", p)
```

```
Tr(A) = p =  0
```

```
[ ]  #Determinante de la matriz
     q = la.det(A)
     print("det(A) = q = ", q)
```

```
det(A) = q =  1.0
```

```
#Raices
dis = p**2 - 4*q
if (dis < 0):
  #Aplicamos valor absoluto para los cálculos
  absdis = abs(dis)
  λ1 = (p + np.sqrt(absdis))*(1/2)
  λ2 = (p - np.sqrt(absdis))*(1/2)
#Raices imaginarias
  print("λ1 = ", λ1, "i")
  print("λ2 = ", λ2,"i")
else:
  λ1 = (p + np.sqrt(dis))*(1/2)
  λ2 = (p - np.sqrt(dis))*(1/2)
#Raices reales
print("λ1 =", λ1)
print("λ2 =", λ2)
```

```
λ1 =  1.0 i
λ2 =  -1.0 i
λ1 = 1.0
λ2 = -1.0
```

Vamos que la matriz pertenece al siguiente caso:

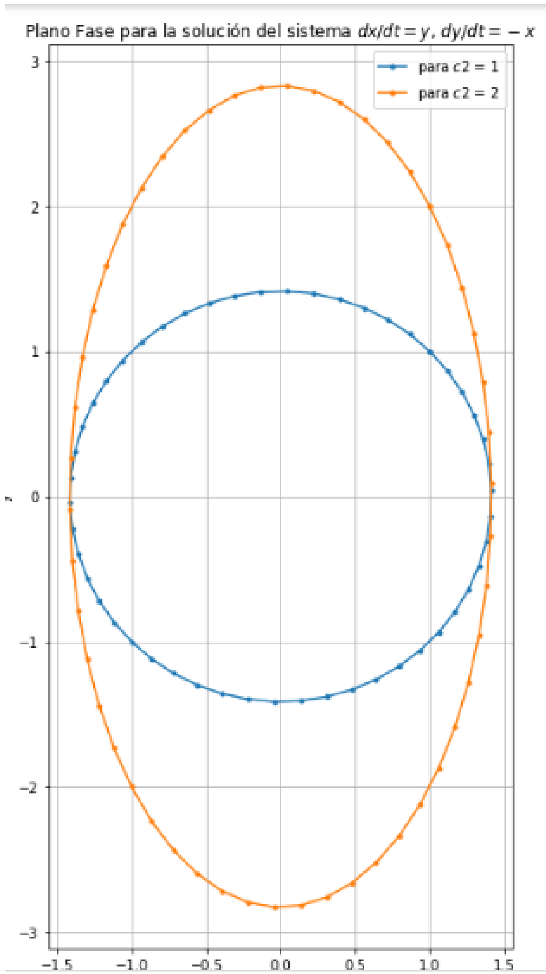
$$\begin{bmatrix} a & -b \\ b & a \end{bmatrix}, b \neq 0$$

La solución:

$$x(t) = [c_1(\cos(bt) + i\sin(bt))\,v_1 + c_2(\cos(bt) - i\sin(bt))\,v_2]\exp(at) \quad (1)\ (2) \qquad x(t) = c_1(\cos(bt) + i\sin(bt))\,v_1 + c_2(\cos(bt) - i\sin(bt))\,v_2 \quad (1)\ (2)$$

```
#Solucionando la ecuación diferencial:
#Condiciones:
π = np.pi
t = np.linspace(0, 2*π, 51)

c1 = 1
c2 = 1
c3 = 2

a = 0
b= 1
#Base canónico
v1 = np.array( [ [ 1 ], [ 0 ] ] )
v2 = np.array( [ [ 0 ], [ 1 ] ] )

#Definir calcular la solución:
def x( t, a, b, c1, c2, v1, v2 ):
#Primer término de la solución
    w1 = c1*( np.cos( b*t ) + np.sin( b*t ) )*v1
#Segundo término de la solución
    w2 = c2*( np.cos( b*t ) - np.sin( b*t ) )*v2
#Factor de la solución
    c = np.exp( a*t )
    return (w1 + w2)*c
#Solución para c1 = 1 y c2 = 1
Sol1 = x( t, a, b, c1, c2, v1, v2 )
xx1 = Sol1[ 0, : ]
xy1 = Sol1[ 1, : ]
Sol2 = x( t, a, b, c1, c3, v1, v2 )
xx2 = Sol2[ 0, : ]
xy2 = Sol2[ 1, : ]
```

```
#Graficando
plt.figure( figsize = ( 6, 12 ) )
plt.plot( xx1, xy1, ".-", label = "para $c2$ = 1" )
plt.plot( xx2, xy2, ".-", label = "para $c2$ = 2" )
plt.legend( loc = "best" )
plt.title( "Plano Fase para la solución del sistema $dx/dt = y$, $dy/dt = -x$" )
plt.xlabel( "x" )
plt.ylabel( "y" )

plt.grid()
plt.show()
```



2.2 Ejercicio 2.

$$\frac{dx}{dt} = y$$
$$\frac{dy}{dt} = x$$

```
#Como conocemos la matriz y hacemos el mismo procedimiento anteriormente:
A = np.array( [ [ 1, 0 ], [ 0, 1 ] ] )
print("A = ", A)
#Calculamos la traza de A
p = np.trace( A )
print( "Tr(A) = p = ", p)
#Calculamos el determinante
q = la.det( A )
print( "det(A) = q = ", q)
#Raices:
dis = p**2 - 4*q
if ( dis < 0 ):
#Valor absoluto
  absdis = abs ( dis )
  λ1 = ( p + np.sqrt( absdis ) )*( 1/2 )
  λ2 = ( p - np.sqrt( absdis ) )*( 1/2 )
  #Raices imaginarias
  print( " λ1 = ", λ1, "i" )
  print( " λ2 = ", λ2, "i" )
else:
  λ1 = ( p + np.sqrt( dis ) )*( 1/2 )
  λ2 = ( p - np.sqrt( dis ) )*( 1/2 )
#Raices Reales:
  print( " λ1 = ", λ1 )
  print( " λ2 = ", λ2 )
```

```
A =  [[1 0]
 [0 1]]
Tr(A) = p =  2
det(A) = q =  1.0
 λ1 =  1.0
 λ2 =  1.0
```

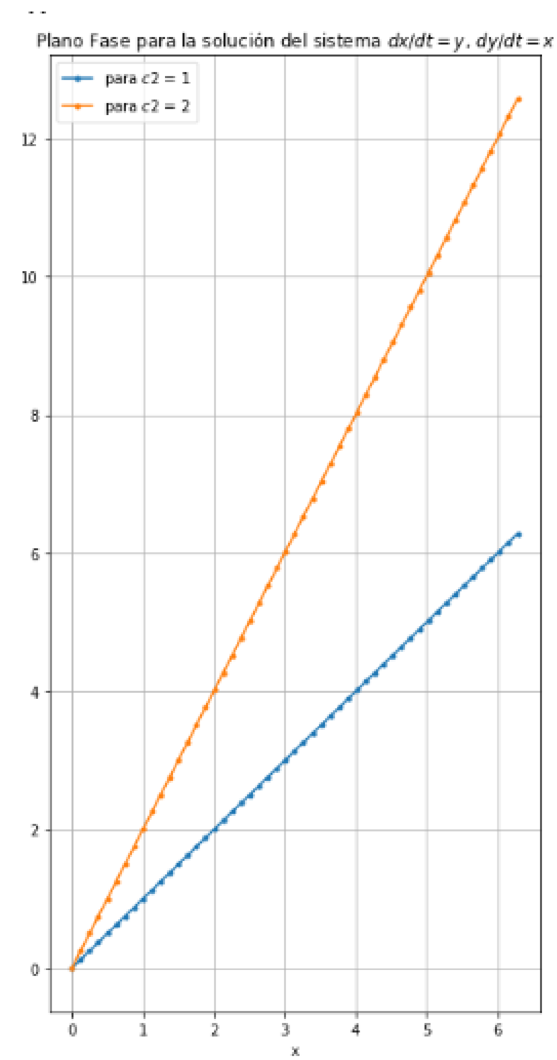
Pertenece al caso A. La solución, entonces:

$$x(t) = c_1 \exp(\lambda t) v_1 + c_2 \exp(\mu t) v_2$$

```
[ ] #Condiciones
π = np.pi
t = np.linspace( 0, 2*π, 51 )
λ = 1
μ = 1
c1 = 1
c2 = 1
c3 = 2

#Eigenvalores y Eigenvectores
eigvals, eigvecs = la.eig( A )
print( "Eigenvalores de A : ", eigvals )
print( eigvecs, " : Eigenvectores de A" )
#Eigenvector 1
eigv1 = ( eigvecs[:,0] ) # ( [ 1 ], [ 0 ] )
v1 = np.array( [ [ 1 ], [ 0 ] ] )
print( v1, " = v1 " )
#Eigenvector 2
eigv2 = ( eigvecs[:,1] ) # ( [ 0 ], [ 1 ] )
v2 = np.array( [ [ 0 ], [ 1 ] ] )
print( v2, " = v2 " )

#Definir la solución:
def x( t, λ, μ, c1, c2, v1, v2 ):
  w1 = c1*( t**λ )*v1
  w2 = c2*( t**μ )*v2
  return w1 + w2
#Solución para C1 = 1 y C2 = 1
Sol1 = x( t, λ, μ, c1, c2, v1, v2 )
```

```
xx1 = Sol1[ 0, : ]
xy1 = Sol1[ 1, : ]
Sol2 = x( t, λ, μ, c1, c3, v1, v2 )
xx2 = Sol2[ 0, : ]
xy2 = Sol2[ 1, : ]
#Graficamos
plt.figure( figsize = ( 6, 12 ) )
plt.plot( xx1, xy1, ".-", label = "para $c2$ = 1" )
plt.plot( xx2, xy2, ".-", label = "para $c2$ = 2" )
plt.legend( loc = "best" )
plt.title( "Plano Fase para la solución del sistema $dx/dt = y$, $dy/dt = x$" )
plt.xlabel( "x" )
plt.ylabel( "y" )

plt.grid()
plt.show()
```

```
Eigenvalores de A :  [1.+0.j 1.+0.j]
[[1. 0.]
 [0. 1.]]  : Eigenvectores de A
[[1]
 [0]]  = v1
[[0]
 [1]]  = v2
```



2.3 Ejercicio 3.

$$\frac{d^2x}{dt^2} + \omega_0^2 x = 0,\ \omega 0 > 0$$

```python
#Ya que conocemos la matriz, la definimos
ω0 = 1

A = np.array( [ [ - ω0**2 , 0 ], [ 0, 1 ] ] )
print( A, " = A" )
#Traza de la matriz
p = np.trace( A )
print( "Tr(A) = p = ", p)
#Determinante de la matriz
q = la.det( A )
print( "det(A) = q = ", q)
#Las raices son:
dis = p**2 - 4*q
if ( dis < 0 ):
  absdis = abs ( dis )
  λ1 = ( p + np.sqrt( absdis ) )*( 1/2 )
  λ2 = ( p - np.sqrt( absdis ) )*( 1/2 )
#Raices imaginarias
  print( " λ1 = ", λ1, "i" )
  print( " λ2 = ", λ2, "i" )
else:
  λ1 = ( p + np.sqrt( dis ) )*( 1/2 )
  λ2 = ( p - np.sqrt( dis ) )*( 1/2 )

  print( " λ1 = ", λ1 )
  print( " λ2 = ", λ2 )
```

```
[[-1  0]
 [ 0  1]]  = A
Tr(A) = p =  0
det(A) = q =  -1.0
 λ1 =  1.0
 λ2 =  -1.0
```

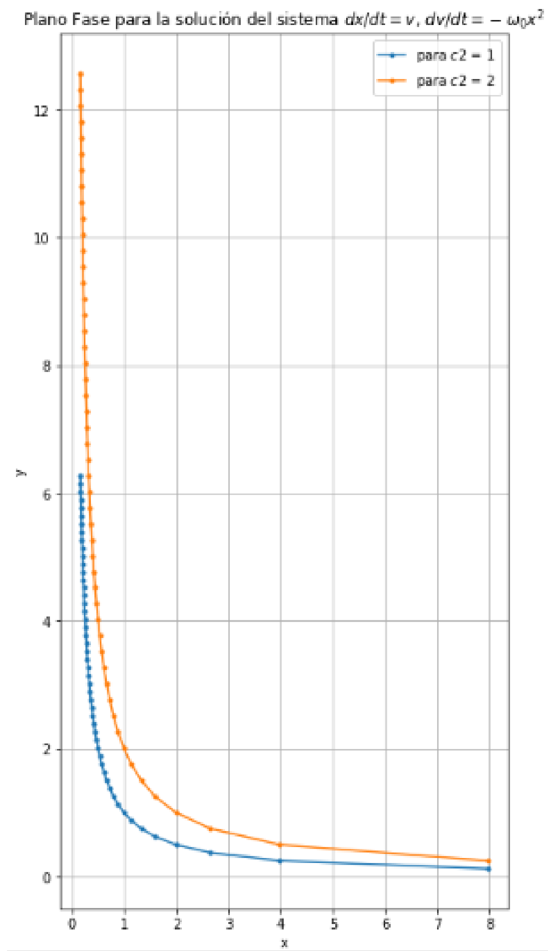
Pertenece al caso A. La solución es:

$$x(t) = c_1 \exp(\lambda t) v_1 + c_2 \exp(\mu t) v_2$$

```
#Condiciones del problema
π = np.pi
t = np.linspace( 0, 2*π, 51 )
λ = - ω0**2
μ = 1
c1 = 1
c2 = 1
c3 = 2
#Eigenvalores y Eigenvectores
eigvals, eigvecs = la.eig( A )
print( "Eigenvalores de A : ", eigvals )
print( eigvecs, " : Eigenvectores de A" )
#Eigenvector 1
eigv1 = ( eigvecs[:,0] )
v1 = np.array( [ [ 1 ], [ 0 ] ] )
print( v1, " = v1 " )
#Eigenvector 2
eigv2 = ( eigvecs[:,1] )
v2 = np.array( [ [ 0 ], [ 1 ] ] )
print( v2, " = v2 " )
#Función para la solución
def x( t, λ, μ, c1, c2, v1, v2 ):
  w1 = c1*( t**λ )*v1
  w2 = c2*( t**μ )*v2

  return w1 + w2
#Solcuion para C1 = 1 y C2 = 1
Sol1 = x( t, λ, μ, c1, c2, v1, v2 )
xx1 = Sol1[ 0, : ]
xy1 = Sol1[ 1, : ]
Sol2 = x( t, λ, μ, c1, c3, v1, v2 )
xx2 = Sol2[ 0, : ]
xy2 = Sol2[ 1, : ]
#Grafica
plt.figure( figsize = ( 6, 12 ) )
plt.plot( xx1, xy1, ".-", label = "para $c2$ = 1" )
```

```
plt.plot( xx2, xy2, ".-", label = "para $c2$ = 2" )
plt.legend( loc = "best" )
plt.title( "Plano Fase para la solución del sistema $dx/dt = v$, $dv/dt = - \omega_0x^{2}$" )
plt.xlabel( "x" )
plt.ylabel( "y" )

plt.grid()
plt.show()
```

```
Eigenvalores de A :  [-1.+0.j  1.+0.j]
[[1. 0.]
 [0. 1.]]  : Eigenvectores de A
[[1]
 [0]]  = v1
[[0]
 [1]]  = v2
```



2.4 Ejercicio 4.

$$\frac{dx}{dt} = -2x$$
$$\frac{dy}{dt} = 2z$$
$$\frac{dz}{dt} = -2y$$

```
[ ] A = np.array( [ [ -2, 0, 0 ], [ 0, 0, 2 ], [ 0, -2, 0 ] ] )
    print( A, " = A " )
```

```
[[-2  0  0]
 [ 0  0  2]
 [ 0 -2  0]]  = A
```

```
[ ] p = np.trace( A )
    print( "Tr(A) = ", p)
```

```
Tr(A) =  -2
```

```
[ ] q = la.det( A )
    print( "det(A) = ", q)
```

```
det(A) =  -8.0
```

```
[ ] eigvals, eigvecs = la.eig( A )
    print( eigvals, " : Eigenvalores " )
```

```
[ 0.+2.j  0.-2.j -2.+0.j]  : Eigenvalores
```

```
[ ] print( eigvecs, " : Eigenvectores " )
```

```
[[ 0.        +0.j          0.        -0.j          1.        +0.j        ]
 [-0.70710678+0.j         -0.70710678-0.j          0.        +0.j        ]
 [ 0.        -0.70710678j  0.        +0.70710678j  0.        +0.j        ]]  : Eigenvectores
```

```
[ ] Λ = np.array( [ [ complex( 0, 2 ), 0, 0 ],
                   [ 0, complex( 0, -2 ), 0 ],
                   [ 0, 0, -2 ] ] )
    print("Λ = ", Λ)
```

```
Λ =  [[ 0.+2.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.-2.j  0.+0.j]
 [ 0.+0.j  0.+0.j -2.+0.j]]
```

+ Código + Texto

```
[ ] eigvec = np.array( [ [ 0, 0, 1 ],
                        [ - np.sqrt( 2 )/2 , - np.sqrt( 2 )/2, 0 ],
                        [ complex( 0, - np.sqrt( 2 )/2 ), complex( 0, np.sqrt( 2 )/2 ), 0 ] ] )
    print(eigvec, "= P")
```

```
[[ 0.        +0.j          0.        +0.j          1.        +0.j        ]
 [-0.70710678+0.j         -0.70710678+0.j          0.        +0.j        ]
 [ 0.        -0.70710678j  0.        +0.70710678j  0.        +0.j        ]] = P
```

+ Código + Texto

```
[ ] print( la.inv( eigvec), " = P^-1 " )
```

```
[[-0.        -0.j         -0.70710678+0.j          0.        +0.70710678j]
 [-0.        -0.j         -0.70710678+0.j          0.        -0.70710678j]
 [ 1.        -0.j          0.        +0.j          0.        +0.j        ]]  = P^-1
```

```
[ ] prod = eigvec @ Λ @ la.inv ( eigvec )
    print(prod)
```

```
[[-2.+0.00000000e+00j  0.+0.00000000e+00j  0.+0.00000000e+00j]
 [ 0.+0.00000000e+00j  0.+2.02930727e-17j  2.+0.00000000e+00j]
 [ 0.+0.00000000e+00j -2.+0.00000000e+00j  0.+2.02930727e-17j]]
```

```
[ ] eigvals = np.array( [ -2, complex( 0, 2 ), complex( 0, -2 ) ] )
    print( " Eigenvalores : ", eigvals )
```

```
 Eigenvalores :  [-2.+0.j  0.+2.j  0.-2.j]
```

```
[ ] Λ = np.array( [ [ -2, 0, 0 ],
                   [ 0, complex( 0, 2 ), 0 ],
                   [ 0, 0, complex( 0, -2 ) ] ] )

    print( Λ, " = Λ " )
```

```
[[-2.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.+2.j  0.+0.j]
 [ 0.+0.j  0.+0.j  0.-2.j]]  = Λ
```

```
eig1 = np.array( [ [ 1 ], [ 0 ], [ 0 ] ] )
eig2 = np.array( [ [ 0 ], [ complex( 0, -1 ) ], [ 1 ] ] )
eig3 = np.array( [ [ 0 ], [ complex( 0, 1 ) ], [ 1 ] ] )
print( eig1, " = v1 " )
print( eig2, " = v2 " )
print( eig3, " = v3 " )
```

```
[[1]
 [0]
 [0]]  = v1
[[0.+0.j]
 [0.-1.j]
 [1.+0.j]]  = v2
[[0.+0.j]
 [0.+1.j]
 [1.+0.j]]  = v3
```

```
[ ] eigvec = np.array( [ [ 1, 0, 0 ],
                        [ 0, complex( 0, -1), complex( 0, 1 ) ],
                        [ 0, 1, 1 ] ] )

    print( eigvec, " = P " )
    print( la.inv( eigvec ), " = P^-1 " )
    prod = eigvec @ Λ @ la.inv ( eigvec )
    print( prod, " = A " )
```

```
[[1.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 0.-1.j 0.+1.j]
 [0.+0.j 1.+0.j 1.+0.j]]  = P
[[ 1. -0.j   0. -0.j  -0. -0.j ]
 [ 0. +0.j   0. +0.5j  0.5-0.j ]
 [ 0. +0.j   0. -0.5j  0.5-0.j ]]  = P^-1
[[-2.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.+0.j  2.+0.j]
 [ 0.+0.j -2.+0.j  0.+0.j]]  = A
```

Los casos cumplen:

$A = P^{-1}\Lambda P$

$$\vec{x}(t) = \vec{x_0}\exp(At)$$

$$\vec{x}(t) = \vec{x_0}\exp(P\Lambda P^{-1}t)$$

$$\vec{x}(t) = \vec{x_0}\exp(P\Lambda P^{-1}t)$$

$$\vec{x}(t) = \vec{x_0}P\exp(\Lambda t)P^{-1}$$

$$\vec{x}(t) = \vec{x_0}\begin{bmatrix} 1 & 0 & 0 \\ 0 & -i & i \\ 0 & 1 & 1 \end{bmatrix}\exp\left(\begin{bmatrix} -2 & 0 & 0 \\ 0 & 2i & 0 \\ 0 & 0 & -2i \end{bmatrix}t\right)\begin{bmatrix} 1 & 0 & 0 \\ 0 & i/2 & 1/2 \\ 0 & -i/2 & 1/2 \end{bmatrix}$$

$$\vec{x}(t) = \vec{x_0}\begin{bmatrix} 1 & 0 & 0 \\ 0 & -i & i \\ 0 & 1 & 1 \end{bmatrix}\begin{bmatrix} e^{-2t} & 0 & 0 \\ 0 & e^{2it} & 0 \\ 0 & 0 & e^{-2it} \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & i/2 & 1/2 \\ 0 & -i/2 & 1/2 \end{bmatrix}$$

$$\vec{x}(t) = \vec{x_0}\begin{bmatrix} e^{-2t} & 0 & 0 \\ 0 & -ie^{2it} & 0 \\ 0 & 0 & e^{-2it} \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & i/2 & 1/2 \\ 0 & -i/2 & 1/2 \end{bmatrix}$$

$$\vec{x}(t) = \vec{x_0}\begin{bmatrix} e^{-2t} & 0 & 0 \\ 0 & -\frac{1}{2}e^{2it} & 0 \\ 0 & 0 & \frac{1}{2}e^{-2it} \end{bmatrix}$$

$$\vec{x}(t) = \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}\begin{bmatrix} e^{-2t} & 0 & 0 \\ 0 & -\frac{1}{2}e^{2it} & 0 \\ 0 & 0 & \frac{1}{2}e^{-2it} \end{bmatrix}$$

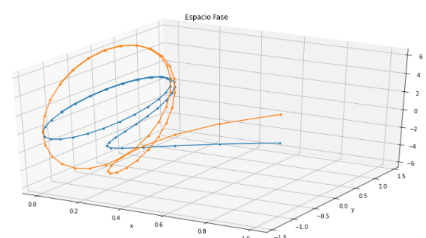
$$\vec{x}(t) = \left(x_0e^{-2t},\ -\tfrac{y_0}{2}e^{i2t},\ \tfrac{z_0}{2}e^{i(-2t)}\right)$$

$$\vec{x}(t) = \left(x_0e^{-2t},\ -\tfrac{y_0}{2}(\cos(2t) + i\sin(2t)),\ \tfrac{z_0}{2}(\cos(2t) - i\sin(2t))\right)$$

$$\vec{x}(t) = \left(c_1x_0e^{-2t},\ -\tfrac{c_2y_0}{2}(\cos(2t) + i\sin(2t)),\ \tfrac{c_3z_0}{2}(\cos(2t) - i\sin(2t))\right)$$

```python
t = np.linspace( 0, 2*np.pi, 51 )
cond0 = np.array( [ 1, 2, 3 ] )
c1 = 1
c2 = 1
c3 = 1
c4 = 3
def x( x0, t, c1, c2, c3 ):
  compx = x0[ 0 ]
  compy = x0[ 1 ]
  compz = x0[ 2 ]
    # print( x0 )
    # print( y0 )
    # print( z0 )
  x = c1 * compx * np.exp( -2*t )
  y = c2 * ( np.cos( 2*t ) + np.sin( 2*t ) )*( -compy/2 )
  z = c3 * ( np.cos( 2*t ) - np.sin( 2*t ) )*( compz/2 )
  sol = np.array( [ x, y, z ] )
  return sol
#Primera solución
Solve1 = x( cond0, t, c1, c2, c3 )
xx1 = Solve1[ 0, : ]
xy1 = Solve1[ 1, : ]
xz1 = Solve1[ 2, : ]
#Segunda solución
Solve2 = x( cond0, t, c1, c2, c4 )
xx2 = Solve2[ 0, : ]
xy2 = Solve2[ 1, : ]
xz2 = Solve2[ 2, : ]
#Creamos la figura
fig = plt.figure( figsize = ( 16, 8 ))
```

```python
#Plano 3d.
ax1 = fig.add_subplot( 111, projection = "3d" )
plt.plot( xx1, xy1, xz1, ".-", label = " $c3$ = 1 " )
plt.plot( xx2, xy2, xz2, ".-", label = " $c3$ = 3 " )
plt.title( "Espacio Fase" )
plt.xlabel( "x" )
plt.ylabel( "y" )
#plt.zlabel( "z" )
plt.show()
```



2.5 Ejercicio 5.

$$\frac{dx}{dt} = -x + z$$
$$\frac{dy}{dt} = 3y$$
$$\frac{dz}{dt} = -x - z$$

```
#Matriz A
A = np.array( [ [ -1, 0, 1 ], [ 0, 3, 0 ], [ -1, 0, -1 ] ] )
print( A, " = A " )
```

```
[[-1  0  1]
 [ 0  3  0]
 [-1  0 -1]]  = A
```

```
#Traza de la matriz A
p = np.trace( A )
print( "Tr(A) = ", p, " : Traza de la Matriz" )
```

```
Tr(A) =  1  : Traza de la Matriz
```

```
#Determinante
q = la.det( A )
print( "det(A) = ", q)
```

```
det(A) =  6.0
```

```
#Eigenvalores y eigenvectores
eigvals, eigvecs = la.eig( A )
print( eigvals, " : Eigenvalores " )
print( eigvecs, " : Eigenvectores " )
```

```
[-1.+1.j -1.-1.j  3.+0.j]  : Eigenvalores
[[0.70710678+0.j         0.70710678-0.j         0.        +0.j        ]
 [0.        +0.j         0.        -0.j         1.        +0.j        ]
 [0.        +0.70710678j 0.        -0.70710678j 0.        +0.j        ]]  : Eigenvectores
```

```
Λ = np.array( [ [ complex( -1, 1 ), 0, 0 ],
                [ 0, complex( -1, -1 ), 0 ],
                [ 0, 0, 3 ] ] )
```

```
print( Λ, " = Λ " )
```

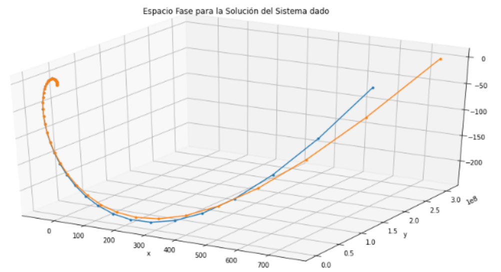
```
[[-1.+1.j  0.+0.j  0.+0.j]
 [ 0.+0.j -1.-1.j  0.+0.j]
 [ 0.+0.j  0.+0.j  3.+0.j]]  = Λ
```

```
prod = la.inv( eigvecs ) @ Λ @ eigvecs

print( prod )
```

```
[[ 1.+0.5j -2.+0.5j  0.+0.j ]
 [-2.+0.5j  1.+0.5j  0.+0.j ]
 [ 0.+0.j   0.+0.j  -1.-1.j ]]
```

La solución entonces:
$\vec{x}(t) = [c_1(\cos(t) + i\sin(t))v_1 + c_2(\cos(t) - i\sin(t))v_2][e^{-t}] + c_3 e^{3t} v_3$

```
t = np.linspace( 0, 2*np.pi, 51 )

c1 = 1
c2 = 1
c3 = 1
c4 = 2

v1 = np.array( [ [ np.sqrt( 2 )/2 ], [ 0 ], [   np.sqrt( 2 )/2 ] ] )
v2 = np.array( [ [ np.sqrt( 2 )/2 ], [ 0 ], [ - np.sqrt( 2 )/2 ] ] )
v3 = np.array( [ [ 0 ], [ 1 ], [ 0 ] ] )

def x( t, c1, c2, c3, v1, v2, v3 ):
  w1 = c1 * ( np.cos( t ) + np.sin( t ) ) * v1
  w2 = c2 * ( np.cos( t ) - np.sin( t ) ) * v2
  k = np.exp( t )
  w3 = c3 * np.exp( 3*t ) * v3
  sol = ( w1 + w2 ) * k + w3
  return sol

Solve1 = x( t, c1, c2, c3, v1, v2, v3 )
xx1 = Solve1[ 0, : ]
xy1 = Solve1[ 1, : ]
xz1 = Solve1[ 2, : ]

Solve2 = x( t, c1, c2, c4, v1, v2, v3 )
xx2 = Solve2[ 0, : ]
xy2 = Solve2[ 1, : ]
xz2 = Solve2[ 2, : ]


fig = plt.figure( figsize = ( 16, 8 ))
#Plano 3d
ax1 = fig.add_subplot( 111, projection = "3d" )
plt.plot( xx1, xy1, xz1, ".-", label = " $c3$ = 1 " )
plt.plot( xx2, xy2, xz2, ".-", label = " $c3$ = 3 " )
plt.title( "Espacio Fase para la Solución del Sistema dado" )
plt.xlabel( "x" )
plt.ylabel( "y" )
# plt.zlabel( "z" )
plt.show()
```



2.6 Ejercicio 6.

$$\frac{dx}{dt} = -x$$
$$\frac{dy}{dt} = x + 2y$$
$$x(0) = 0, y(0) = 3$$

```
[ ] #Agregamos la matriz A.
    A = np.array( [ [ -1, 0 ], [ 1, 2 ] ] )
    print("A = ", A)

    A =  [[-1  0]
     [ 1  2]]
```

```
[ ] #Traza de la matriz
    p = np.trace( A )
    print("Tr(A) = p = ", p)

    Tr(A) = p =  1
```

```
[ ] #Determinante de la matriz
    q = la.det(A)
    print("det(A) = q = ", q)

    det(A) = q =  -2.0
```

```
#Raices
dis = p**2 - 4*q
if ( dis < 0 ):

  absdis = abs ( dis )
  λ1 = ( p + np.sqrt( absdis ) )*( 1/2 )
  λ2 = ( p - np.sqrt( absdis ) )*( 1/2 )
#Raices imaginarias
  print( " λ1 = ", λ1, "i" )
  print( " λ2 = ", λ2, "i" )
else:
  λ1 = ( p + np.sqrt( dis ) )*( 1/2 )
  λ2 = ( p - np.sqrt( dis ) )*( 1/2 )
#Raices reales
  print( " λ1 = ", λ1 )
  print( " λ2 = ", λ2 )
eigvals, eigvecs = la.eig( A )
print("Eigenvectores:", eigvecs)
```

```
 λ1 =  2.0
 λ2 =  -1.0
Eigenvectores: [[ 0.          0.9486833 ]
 [ 1.         -0.31622777]]
```

$$\vec{x}(t) = \vec{x}_0 \exp(At)$$

$$A = P \Lambda P^{-1}$$

$$\vec{x}(t) = \vec{x}_0 \exp(P \Lambda P^{-1} t)$$

$$\vec{x}(t) = \vec{x}_0 P \exp(\Lambda t) P^{-1}$$

$$\vec{x}(t) = \vec{x}_0 \begin{bmatrix} 0 & -3 \\ 1 & 1 \end{bmatrix} \exp\left(\begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix} t \right) \begin{bmatrix} \frac{1}{3} & 1 \\ -\frac{1}{3} & 0 \end{bmatrix}$$

$$\vec{x}(t) = \vec{x}_0 \begin{bmatrix} 0 & -3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} e^{et} & 0 \\ 0 & e^{-t} \end{bmatrix} \begin{bmatrix} \frac{1}{3} & 1 \\ -\frac{1}{3} & 0 \end{bmatrix}$$

$$\vec{x}(t) = \vec{x}_0 \begin{bmatrix} 0 & -3e^{-t} \\ e^{2t} & e^{-t} \end{bmatrix} \begin{bmatrix} \frac{1}{3} & 1 \\ -\frac{1}{3} & 0 \end{bmatrix}$$

$$\vec{x}(t) = \vec{x}_0 \begin{bmatrix} e^{-t} & 0 \\ \frac{1}{3}e^{2t} - \frac{1}{3}e^{-t} & e^{2t} \end{bmatrix}$$

$$\vec{x}(t) = \begin{bmatrix} x_0 & y_0 \end{bmatrix} \begin{bmatrix} e^{-t} & 0 \\ \frac{1}{3}e^{2t} - \frac{1}{3}e^{-t} & e^{2t} \end{bmatrix}$$

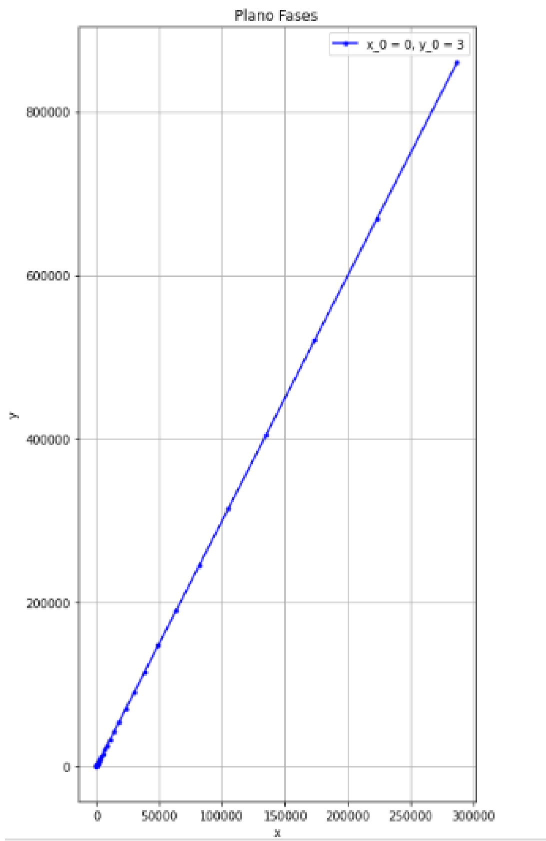
$$\vec{x}(t) = \left(x_0 e^{-t} + \frac{y_0}{3}(e^{2t} - e^{-t}),\ y_0 e^{2t} \right)$$

$$\begin{bmatrix} x(0) = 0 & y(0) = 3 \end{bmatrix}$$

$$\vec{x}(t) = \left(\frac{3}{3}(e^{2t} - e^{-t}),\ 3e^{2t} \right)$$

$$\vec{x}(t) = (e^{2t} - e^{-t},\ 3e^{2t})$$

```python
t = np.linspace( 0, 2*np.pi, 51 )

def x( t ):
  w1 = np.exp( 2*t ) - np.exp( -t )
  w2 = 3 * np.exp( 2*t )

  sol = np. array( [ w1, w2 ] )

  return sol

Solve = x( t )
xx = Solve[ 0, : ]
xy = Solve[ 1, : ]


plt.figure( figsize = ( 6, 12 ) )
plt.plot( xx, xy, "b.-", label = "x_0 = 0, y_0 = 3"  )
plt.legend( loc = "best" )
plt.title( "Plano Fases" )
plt.xlabel( "x" )
plt.ylabel( "y" )
plt.grid()
plt.show()
```



2.7 Ejercicio 7.

$$\frac{dx}{dt} = 2x + y$$
$$\frac{dy}{dt} = x + y$$
$$x(1) = 1,\ y(1) = 1$$

```python
A = np.array( [ [ 2, 1 ], [ 1, 1 ] ] )
print( "A =", A )
p = np.trace(A)
print("Tr(A) = p = ", p)
q = la.det(A)
print("det(A) = q = ",q)

#Raices
dis = p**2 - 4*q

if (dis< 0):
  absdis = abs (dis)
  λ1 = ( p + np.sqrt( absdis ) )*( 1/2 )
  λ2 = ( p - np.sqrt( absdis ) )*( 1/2 )
  print( " λ1 = ", λ1, "i" )
  print( " λ2 = ", λ2, "i" )
else:
  λ1 = ( p + np.sqrt( dis ) )*( 1/2 )
  λ2 = ( p - np.sqrt( dis ) )*( 1/2 )
  print( " λ1 = ", λ1 )
  print( " λ2 = ", λ2 )
  print( " " )
eigvals, eigvecs = la.eig( A )
print( eigvecs, " : Eigenvectores " )

eigvals = np.array( [ ( 3 + np.sqrt(5) )/2, ( 3 - np.sqrt(5) )/2 ] )
print( eigvals, " : Eigenvalores " )
print( " " )

v1 = np.array( [ [ 1 + np.sqrt( 5 ) ],
                [ 2 ] ] )
print( v1, " = v1 " )
print( " " )
```

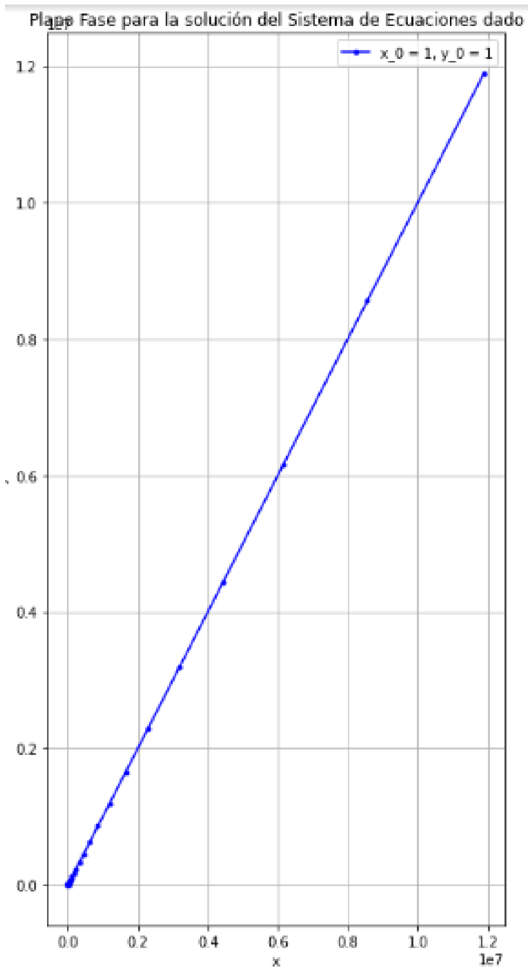
```python
v2 = np.array( [ [ 1 - np.sqrt( 5 ) ],
                 [ 2 ] ] )
print( v2, " = v2 " )
print( " " )

eigvecs = np.array( [ [ 1 + np.sqrt( 5 ), 1 - np.sqrt( 5 ) ],
                      [ 2, 2 ] ] )
print( eigvecs, " : Eigenvectores " )
print( " " )

Λ = np.array( [ [ ( 3 + np.sqrt(5) )/2, 0 ],
                [ 0, ( 3 - np.sqrt(5) )/2 ] ] )

print( Λ, " = Λ " )
print( " " )

print( eigvecs, " = P " )
print( " " )

print( la.inv( eigvecs ), " = P^-1 " )
print( " " )

prod = eigvecs @ Λ @ la.inv( eigvecs )

print( prod )
```

```
A = [[2 1]
 [1 1]]
Tr(A) = p =  3
det(A) = q =  1.0
 λ1 =  2.618033988749895
 λ2 =  0.3819660112501051

[[ 0.85065081 -0.52573111]
 [ 0.52573111  0.85065081]]  : Eigenvectores
[2.61803399 0.38196601]  : Eigenvalores

[[3.23606798]
 [2.        ]]  = v1

[[-1.23606798]
 [ 2.        ]]  = v2

[[ 3.23606798 -1.23606798]
 [ 2.          2.        ]]  : Eigenvectores

[[2.61803399 0.        ]
 [0.         0.38196601]]  = Λ

[[ 3.23606798 -1.23606798]
 [ 2.          2.        ]]  = P

[[ 0.2236068   0.1381966]
 [-0.2236068   0.3618034]]  = P^-1

[[2. 1.]
 [1. 1.]]
```

Saltándonos los cálculos obtenemos que la solución del sistema de ecuaciones diferenciales es:

$$\vec{x_0}(t) = \left(\frac{1+\sqrt{5}}{2\sqrt{5}} e^{\frac{3+\sqrt{5}}{2}t}, \frac{1+\sqrt{5}}{2\sqrt{5}} e^{\frac{3-\sqrt{5}}{2}t} \right)$$

```
t = np.linspace( 0, 2*np.pi, 51 )
def x(t):
  fac = ( 1 + np.sqrt( 2 ) )/( 2*np.sqrt( 2 ) )
  f_e1 = ( 3 + np.sqrt( 5 ) )/( 2 )
  f_e2 = ( 3 - np.sqrt( 5 ) )/( 2 )
  w1 = fac * np.exp( f_e1 * t )
  w2 = fac * np.exp( f_e2 * t )
  sol = np.array( [ w1, w1 ] )
  return sol
Solve = x(t)
xx = Solve[ 0, : ]
xy = Solve[ 1, : ]
plt.figure( figsize = ( 6, 12 ) )
plt.plot( xx, xy, "b.-", label = "x_0 = 1, y_0 = 1"  )
plt.legend( loc = "best" )
plt.title( "Plano Fase para la solución del Sistema de Ecuaciones dado" )
plt.xlabel( "x" )
plt.ylabel( "y" )

plt.grid()
plt.show()
```



2.8 Ejercicio 8.

$$\frac{dx}{dt} = Ax$$
$$x(0) = (0,3)$$
$$\text{A}= \begin{bmatrix} 0 & 3 \\ 1 & -2 \end{bmatrix}$$

```
[ ] A = np.array( [ [ 0, 3 ], [ 1, -2 ] ] )
    print("A = ", A)
    p = np.trace (A)
    print( "Tr(A) = p = ", p)
    q = la.det(A)
    print( "det(A) = q = ", q)

    #Raices
    dis = p**2 - 4*q

    if ( dis < 0 ):
      absdis = abs ( dis )
      λ1 = ( p + np.sqrt( absdis ) )*( 1/2 )
      λ2 = ( p - np.sqrt( absdis ) )*( 1/2 )
      print( " λ1 = ", λ1, "i" )
      print( " λ2 = ", λ2, "i" )
    else:
      λ1 = ( p + np.sqrt( dis ) )*( 1/2 )
      λ2 = ( p - np.sqrt( dis ) )*( 1/2 )
      print( " λ1 = ", λ1 )
      print( " λ2 = ", λ2 )
      print( " " )
    eigvals, eigvecs = la.eig( A )

    print( "Eigenvectores =", eigvecs )
```

```
A =  [[ 0  3]
 [ 1 -2]]
Tr(A) = p =  -2
det(A) = q =  -3.0
 λ1 =  1.0
 λ2 =  -3.0

Eigenvectores = [[ 0.9486833  -0.70710678]
 [ 0.31622777  0.70710678]]
```

```python
eigvals = np.array( [ 1, -3 ] )
print( "Eigenvalores =", eigvals)
v1 = np.array( [ [ 3 ],
                 [ 1 ] ] )
print( v1, " = v1 " )
v2 = np.array( [ [ -1 ],
                 [ 1 ] ] )
print( "v2 = ", v2 )
eigvecs = np.array( [ [ 3, -1 ],
                      [ 1, 1 ] ] )
print( "Eigenvectores =", eigvecs)
Λ = np.array( [ [ 1, 0 ],
                [ 0, -3 ] ] )
print( Λ, " = Λ " )
print( eigvecs, " = P " )
print( la.inv( eigvecs ), " = P^-1 " )
prod = eigvecs @ Λ @ la.inv( eigvecs )
print( prod )
```

```
Eigenvalores = [ 1 -3]
[[3]
 [1]]  = v1
v2 =  [[-1]
 [ 1]]
Eigenvectores = [[ 3 -1]
 [ 1  1]]
[[ 1  0]
 [ 0 -3]]  = Λ
[[ 3 -1]
 [ 1  1]]  = P
[[ 0.25  0.25]
 [-0.25  0.75]]  = P^-1
[[ 0.  3.]
 [ 1. -2.]]
```

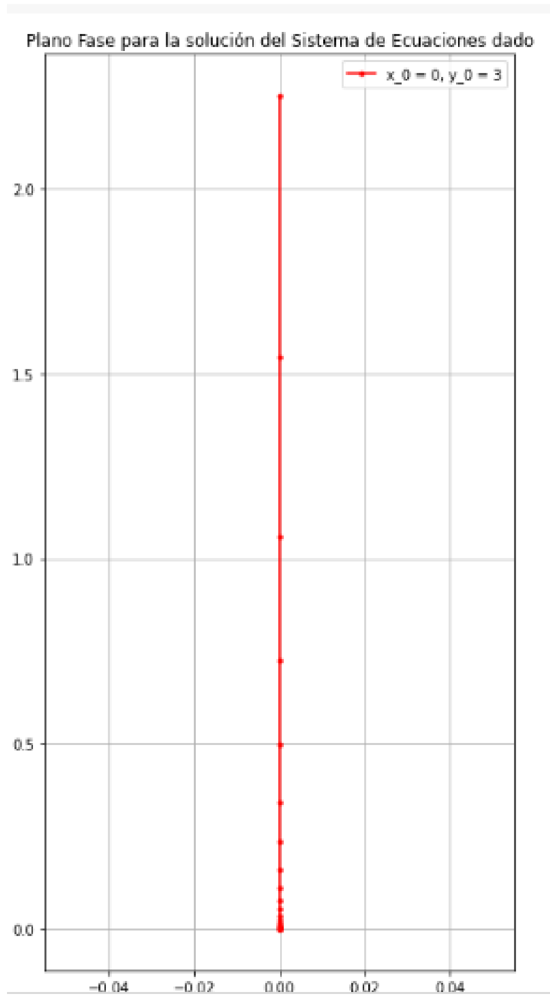
Solución al sistema de ecuaciones diferenciales:

$\vec{x_0}(t) = \left(\frac{3}{4}x_0e^t,\ \frac{3}{4}y_0e^{-3t}\right)$

x0 = 0:

$\vec{x_0}(t) = \left(0,\ \frac{3}{4}y_0e^{-3t}\right)$

```python
t = np.linspace( 0, 2*np.pi, 51 )
def x(t):
  x_0 = np.array( [ 0, 3 ] )
  x = x_0[ 0 ]
  y = x_0[ 1 ]
  w1 = ( 3/4 )*x*np.exp( t )
  w2 = ( 3/4 )*y*np.exp( -3*t )
  sol = np.array( [ w1, w2 ] )
  return sol
Solve = x(t)
xx = Solve[ 0, : ]
xy = Solve[ 1, : ]
plt.figure( figsize = ( 6, 12 ) )
plt.plot( xx, xy, "r.-", label = "x_0 = 0, y_0 = 3"  )
plt.legend( loc = "best" )
plt.title( "Plano Fase para la solución del Sistema de Ecuaciones dado" )
plt.xlabel( "x" )
plt.ylabel( "y" )
plt.grid()
plt.show()
```



2.9 Ejercicio 9.

$$\frac{dx}{dt} = Ax$$
$$x(0) = (0, -b, b)$$
$$A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 2 & -3 \end{bmatrix}$$

```python
A = np.array( [ [ 2, 0, 0 ], [ 0, -1, 0 ], [ 0, 2, -3 ] ] )
print( "A =", A )
print( " " )

p = np.trace( A ) # Calculamos la traza de la matriz A
print( "Tr(A) = ", p )
print( " " )

q = la.det( A ) # Calculamos el determinante de la matriz A
print( "det(A) = ", q)
print( " " )

eigvals, eigvecs = la.eig( A )
print( "Eigenvalores = ",eigvals)
print( "Eigenvectores = ",eigvecs )

Λ = np.array( [ [ 2, 0, 0 ],
                [ 0, -3, 0 ],
                [ 0, 0, -1 ] ] )

print( "Λ = ", Λ )
print( " " )
prod = la.inv( eigvecs ) @ Λ @ eigvecs
print( prod )
```

```
A = [[ 2  0  0]
 [ 0 -1  0]
 [ 0  2 -3]]

Tr(A) =  -2

det(A) =  6.0

Eigenvalores =  [ 2.+0.j -3.+0.j -1.+0.j]
Eigenvectores =  [[1.         0.         0.        ]
 [0.         0.         0.70710678]
 [0.         1.         0.70710678]]
Λ =  [[ 2  0  0]
 [ 0 -3  0]
 [ 0  0 -1]]
```

```
 [0.         1.         0.70710678]]
Λ =  [[ 2  0  0]
 [ 0 -3  0]
 [ 0  0 -1]]

[[ 2.          0.          0.        ]
 [ 0.         -1.          1.41421356]
 [ 0.          0.         -3.        ]]
```

```python
eigvals = np.array( [ -1, 2, -3 ] )
print( " Eigenvalores = ", eigvals )
print( " " )

Λ = np.array( [ [ -1, 0, 0 ],
                [ 0, 2, 0 ],
                [ 0, 0, -3 ] ] )

print( "Λ =",Λ)
print( " " )

eig1 = np.array( [ [ 0 ], [ 1 ], [ 1 ] ] )
eig2 = np.array( [ [ 1 ], [ 0 ], [ 0 ] ] )
eig3 = np.array( [ [ 0 ], [ 0 ], [ 1 ] ] )
print( eig1, " = v1 " )
print( " " )
print( eig2, " = v2 " )
print( " " )
print( eig3, " = v3 " )
print( " " )

eigvec = np.array( [ [ 0, 1, 0 ],
                     [ 1, 0, 0 ],
                     [ 1, 0, 1 ] ] )

print( eigvec, " = P " )
print( " " )

print( la.inv( eigvec ), " = P^-1 " )
print( " " )
```

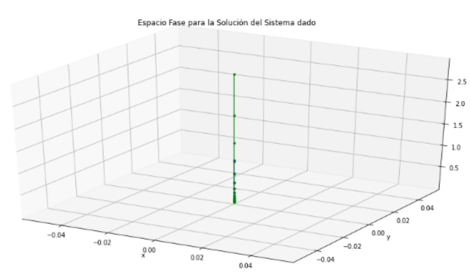
```
prod = eigvec @ Λ @ la.inv( eigvec )

print( prod, " = A " )
```

```
 Eigenvalores =  [-1  2 -3]

Λ = [[-1  0  0]
 [ 0  2  0]
 [ 0  0 -3]]

[[0]
 [1]
 [1]]  = v1

[[1]
 [0]
 [0]]  = v2

[[0]
 [0]
 [1]]  = v3

[[0 1 0]
 [1 0 0]
 [1 0 1]]  = P

[[-0.  1. -0.]
 [ 1.  0. -0.]
 [ 0. -1.  1.]]  = P^-1

[[ 2.  0.  0.]
 [ 0. -1.  0.]
 [ 0.  2. -3.]]  = A
```

Solución al sistema de ecuaciones:

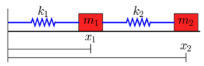
$\vec{x}(t) = (0, 0, be^{-3t})$

```
t = np.linspace( 0.01, 2*np.pi, 51 )
b1 = 1
b2 = 3
def x( t, b ):
  w1 = 0*np.exp( -t )
  w2 = 0*np.exp( 2*t )
  w3 = b*np.exp( -3*t )
  sol = np.array( [ w1, w2, w3 ] )
  return sol
Solve1 = x( t, b1 )
xx1 = Solve1[ 0, : ]
xy1 = Solve1[ 1, : ]
xz1 = Solve1[ 2, : ]
Solve2 = x( t, b2 )
xx2 = Solve2[ 0, : ]
xy2 = Solve2[ 1, : ]
xz2 = Solve2[ 2, : ]

fig = plt.figure( figsize = ( 16, 8 ))
ax1 = fig.add_subplot( 111, projection = "3d" )
plt.plot( xx1, xy1, xz1, "b.-", label = " $c3$ = 1 " )
plt.plot( xx2, xy2, xz2, "g.-", label = " $c3$ = 3 " )
plt.title( "Espacio Fase para la Solución del Sistema dado" )
plt.xlabel( "x" )
plt.ylabel( "y" )

plt.show()
```



2.10 Ejercicio 10.

Se tiene el siguiente sistema de resortes acoplados con dos masas.



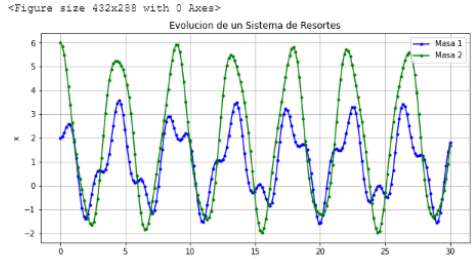
El sistema está sujeto del primer resorte de la izquierda. Cuando el sistema se deja en reposo, las longitudes de los resortes son L1 y L2.
Las ecuaciones de movimiento están dadas por las siguientes ecuaciones diferenciales acopladas:

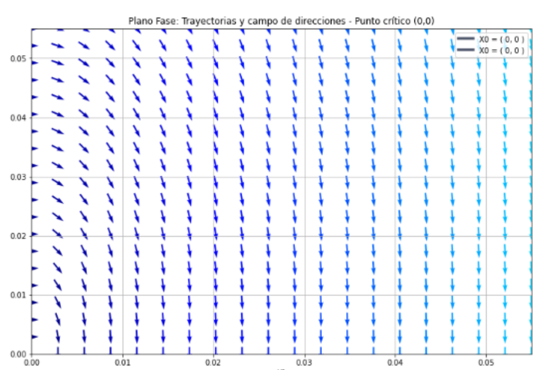
$$
\begin{aligned}
m_1 x_1'' + b_1 x_1' + k_1(x_1 - L_1) - k_2(x_2 - x_1 - L_2) &= 0 \quad (1)\\
m_2 x_2'' + b_2 x_2' + k_2(x_2 - x_1 - L_2) &= 0 \quad (2)
\end{aligned}
$$

Encuentre las soluciones como funciones de t y grafiquelas, así como las trayectorias en el espacio fase. Suponga que L1 = L2 = 1 y que las masas son iguales m1 = m2 = 1.
10.1) Encuentre los eigenvalores del sistema y diga como son las soluciones.
10.2) Caso sin fricción. b1 = b2 = 0, k1 = 6, k2 = 4, condiciones iniciales (x1(0), x'1(0),x2(0), x'2(0)) = (1,0,4,0)
10.3) Igual que el caso anterior pero, con fricción: b1=0.1, b2=0.2

```
[[  0   1   0   0]
 [-10   0   4   0]
 [  0   0   0   1]
 [  4   0  -4   0]]  = A

Puntos criticos del Sistema de Ecuaciones
Xf0 =  [0. 0. 0. 0.]


Jacobiano evaluado en punto critico
[[  0   1   0   0]
 [-10   0   4   0]
 [  0   0   0   1]
 [  4   0  -4   0]]

Eigenvalores del Jacobiano en X
λ1 =  (2.220446049250313e-16+3.464101615137756j)
λ2 =  (2.220446049250313e-16-3.464101615137756j)
λ3 =  (-1.3877787807814457e-16+1.414213562373095j)
λ4 =  (-1.3877787807814457e-16-1.414213562373095j)
```

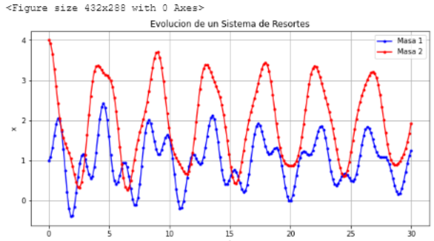
```
Periodo:  1.813799364234217
[[1.]
 [0.]
 [2.]
 [0.]]
<Figure size 432x288 with 0 Axes>
```



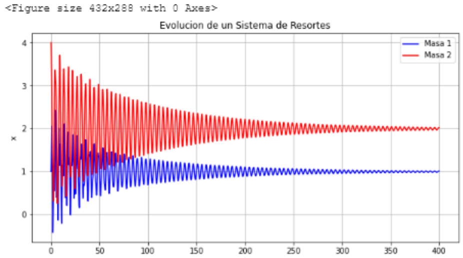


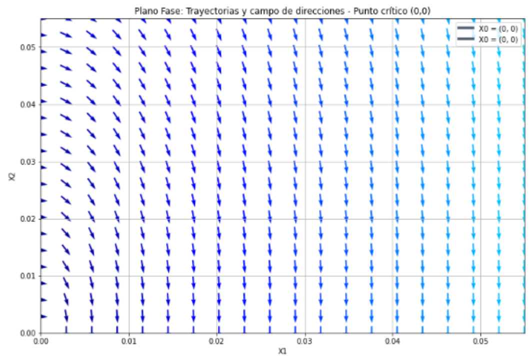
```
[[  0.    1.    0.    0. ]
 [-10.   -0.1   4.    0. ]
 [  0.    0.    0.    1. ]
 [  4.    0.   -4.   -0.2]]  = A

[[1.]
 [0.]
 [2.]
 [0.]]

Puntos criticos del Sistema de Ecuaciones
Xf0 =  [0. 0. 0. 0.]

Jacobiano evaluado en punto critico
[[  0.    1.    0.    0. ]
 [-10.   -0.1   4.    0. ]
 [  0.    0.    0.    1. ]
 [  4.    0.   -4.    0. ]]

Eigenvalores del Jacobiano en X
λ1 =  (-0.03999936001028768+3.463593494852212j)
λ2 =  (-0.03999936001028768-3.463593494852212j)
λ3 =  (-0.010000639989712207+1.414291364612996j)
λ4 =  (-0.010000639989712207-1.414291364612996j)
```

```
Periodo:  1.8139444972104801

[[1.]
 [0.]
 [2.]
 [0.]]
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```





3 Conclusión

Este tema fue de gran interés debido a que en la materia de Métodos matemáticos para la Física 2 vemos un poco sobre este tema. Sin embargo, cabe aclarar que la dificultad de esta actividad en concreto fue muy complicada al menos para mi, ya que, más de un error cometí en el código y corregirlo fue complicado, además de que, el ejercicio 10 no lo incluí por que no pude hacerlo, sin más. Por otra parte, las gráficas de las soluciones al los problemas nos hablan mucho sobre las soluciones que imparte el estado fase de casa solucion del sistema dado.