Examen

October 23, 2020

1 Examen

Instrucciones

- Crea en tu carpeta, un archivo llamado `examen` y pega el texto de las problemas en él (respeta el formato).
- Contesta inmediatamente abajo del problema.
- Gráficas en calidad profesional (pon ejes, unidades, colores, leyenda, etc.)
- La ortografía, redacción y habilidades de comunicación se tomarán en cuenta.

1.0.1 Problema 1

(a) Usando `Simpy`, declara las funciones:

$$y(x) = \sin(x)$$

$$z(x) = \cos(x)$$

$$w(x) = \frac{1}{\cos(x) + \sin(2x)}$$

(b) Obtén la derivada de $g(x)$

$$g(x) = y(x) * z(x)$$

(c) Grafica $w(x)$ en el rango $[0, 1]$

(d) Integra de manera indefinida $w(x)$ y luego evalúala desde 0 a 1.

(e) ¿Cuál es el límite de $y(x), z(x), g(x)$ y $w(x)$ cuando $x \to 0$.?

(f) Expanda $y(x)$ y $z(x)$ hasta 3 orden en serie de Taylor.

[ ]:
```

```

(a) Usando `Simpy`, declara las funciones:

$$y(x) = \sin(x)$$

$$z(x) = \cos(x)$$

$$w(x) = \frac{1}{\cos(x) + \sin(2x)}$$

[1]:
```
from sympy import *
```

[2]:
```
y = Function("y")
z = Function("z")
w = Function("w")
g = Function("g")
x, x0 = symbols ("x, x_0")
```

[3]:
```
y_F = Eq(y(x), sin(x))
y_F
```

[3]: $y(x) = \sin(x)$

[4]:
```
z_F = Eq(z(x), cos(x))
z_F
```

[4]: $z(x) = \cos(x)$

[5]:
```
w_F = Eq(w(x), 1/(cos(x)+sin(2*x)))
w_F
```

[5]: $w(x) = \frac{1}{\sin(2x) + \cos(x)}$

(b) Obtén la derivada de $g(x)$

$$g(x) = y(x) * z(x)$$

[6]:
```
g_F = Eq(g(x), y_F.rhs*z_F.rhs)
g_F
```

[6]: $g(x) = \sin(x)\cos(x)$

[7]:
```
Derivada = Eq(Derivative(g(x),x), Derivative(y_F.rhs*z_F.rhs,x))
simplify(Derivada)
```

[7]: $\frac{d}{dx} g(x) = \cos(2x)$

(c) Grafica $w(x)$ en el rango $[0, 1]$

[8]:
```
import matplotlib.pyplot as plt
import numpy as np
```

[9]:
```
w_F
```

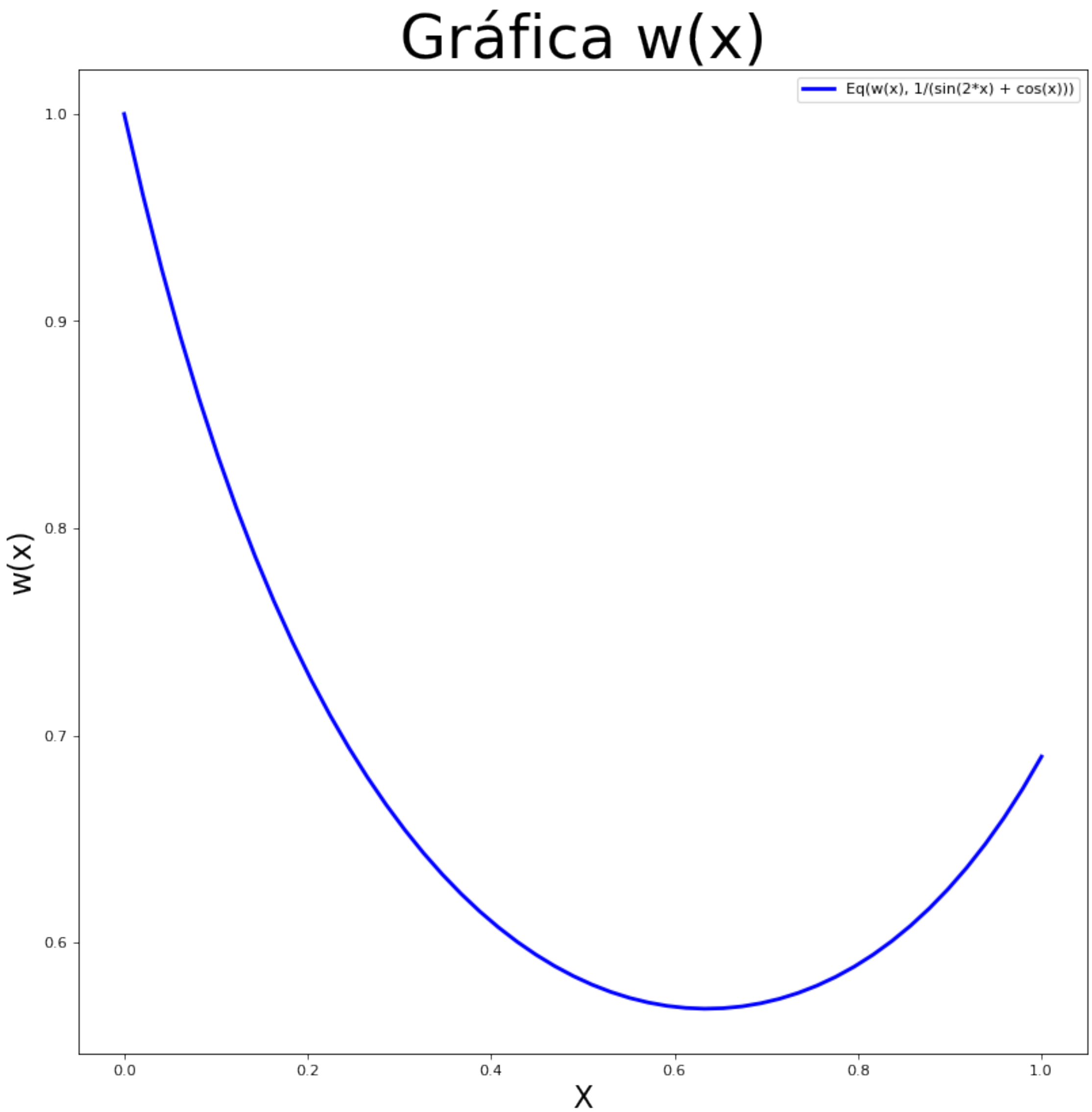
[9]:

$$w(x) = \frac{1}{\sin(2x) + \cos(x)}$$

[10]:
```
x_values = np.linspace(0,1)
w_funct = lambdify(x, w_F.rhs, modules=['numpy'])
y_values = w_funct(x_values)
```

[11]:
```
plt.figure(figsize=(12, 12), dpi=80)
plt.plot(x_values, y_values, linewidth=2.5, linestyle="-",color="blue", label =␣
↪w_F)
plt.xlabel('X', fontsize=20)
plt.ylabel('w(x)', fontsize=20)
plt.title("Gráfica w(x)", fontsize=40)
plt.legend(loc='best')
```

[11]: <matplotlib.legend.Legend at 0x7f03c5c66190>



(d) Integra de manera indefinida $g(x)$ y luego evalúala desde 0 a 1.

[12]:
```
g_int = Eq (integrate(g_F.lhs, x), integrate(g_F.rhs,x))
simplify(g_int)
```

[12]: $$\int g(x)\, dx = \frac{\sin^2(x)}{2}$$

[13]:
```
g_int_eval = Eq (integrate(g_F.lhs,(x,0,1)), g_int.rhs.subs(x,1) - g_int.rhs.
↪subs(x,0))
simplify(g_int_eval)
```

[13]:

$$\int\limits_0^1 g(x)\,dx = \frac{\sin^2(1)}{2}$$

(e) ¿Cuál es el límite de $y(x), z(x), g(x)$ y $w(x)$ cuando $x \to 0$.?

[14]:
```python
Eq(Limit(y_F.lhs, x, 0) ,limit(y_F.rhs, x, 0))
```

[14]:

$$\lim_{x\to 0^+} y(x) = 0$$

[15]:
```python
Eq(Limit(z_F.lhs, x, 0),limit(z_F.rhs, x, 0))
```

[15]:

$$\lim_{x\to 0^+} z(x) = 1$$

[16]:
```python
Eq(Limit(g_F.lhs, x, 0),limit(g_F.rhs, x, 0))
```

[16]:

$$\lim_{x\to 0^+} g(x) = 0$$

[17]:
```python
Eq(Limit(w_F.lhs, x, 0),limit(w_F.rhs, x, 0))
```

[17]:

$$\lim_{x\to 0^+} w(x) = 1$$

(f) Expanda $y(x)$ y $z(x)$ hasta 3 orden en serie de Taylor.

Respecto a un punto x0

[41]:
```python
y_Tay = y_F.rhs.subs(x,x0) + diff(y_F.rhs).subs(x,x0)*(x-x0) + (diff(diff(y_F.
↪rhs)).subs(x,x0))/2*(x-x0)**2 + diff(diff(diff(y_F.rhs))).subs(x,x0)/
↪6*(x-x0)**3
Eq(y(x),y_Tay)
```

[41]:

$$y(x) = -\frac{(x-x_0)^3\cos(x_0)}{6} - \frac{(x-x_0)^2\sin(x_0)}{2} + (x-x_0)\cos(x_0) + \sin(x_0)$$

[42]:
```python
z_Tay = z_F.rhs.subs(x,x0) + diff(z_F.rhs).subs(x,x0)*(x-x0) + (diff(diff(z_F.
↪rhs)).subs(x,x0))/2*(x-x0)**2 + diff(diff(diff(z_F.rhs))).subs(x,x0)/
↪6*(x-x0)**3
Eq(z(x),z_Tay)
```

[42]:

$$z(x) = \frac{(x-x_0)^3\sin(x_0)}{6} - \frac{(x-x_0)^2\cos(x_0)}{2} - (x-x_0)\sin(x_0) + \cos(x_0)$$

NOTA Muestra las expresiones en cada inciso.

1.0.2 Problema 2

El atractor de Rössler esta descrito por el siguiente conjunto de ecuaciones:

$$\frac{dx}{dt} = -y - z$$

$$\frac{dy}{dt} = x + ay$$

$$\frac{dz}{dt} = b + z(x - c)$$

(a) Resuelva las ecuaciones numéricamente para

$$a = 0.13 \quad b = 0.2 \quad c = 6.5$$

y condiciones iniciales

$$x(0) = 0 \quad y(0) = 0 \quad z(0) = 0$$

use el método de Runge-Kutta de 2do orden.

[ ]:

```

```

[20]:

```
def atractor(estado, tiempo):
    g0= -estado[1]-estado[2]
    g1= estado[0] + .13*estado[1]
    g2= .2 + estado[2]*(estado[0]-6.5)
    return np.array([g0,g1,g2])
```

[21]:

```
N = 1000
time = np.linspace(0,100,N)
dt = 100/float(N-1)
#Tomamos 100 como el tiempo de la simulacion
```

[22]:

```
w = np.zeros([N,3])
w[0,0] = 0 #x(0)=0
w[0,1] = 0 #y(0)=0
w[0,2] = 0 #z(0)=0
w
```

[22]:

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       …,
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

[23]:

```
#w es la condicion inicial
#time es el tiempo en el que va
#dt es el tamaño del paso
#derivadas es atractor
```

```
def RK2(w, time, dt, derivadas):
    k0 = dt*derivadas(w, time)
    k1 = dt*derivadas(w + k0, time + dt)
    w_next = w + 0.5*(k0 + k1)

    return w_next
```

[24]:
```
for i in range(N-1):
    w[i+1] = RK2(w[i], time[i], dt, atractor)
w
```

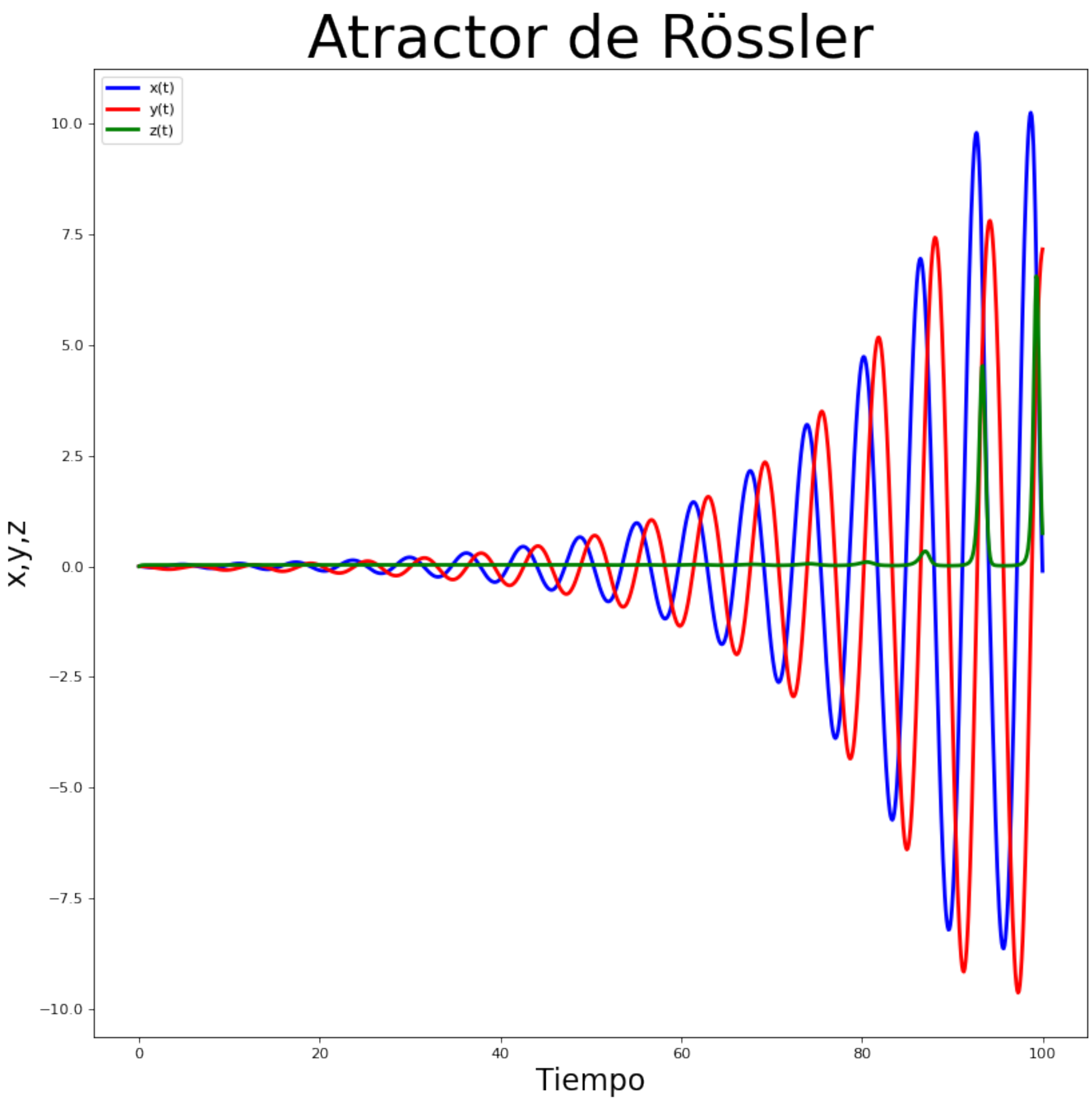
[24]:
```
array([[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [-1.00200300e-03,  0.00000000e+00,  1.35070005e-02],
       [-2.91111346e-03, -1.68623483e-04,  2.10815803e-02],
       …,
       [ 1.55564717e+00,  6.83853572e+00,  2.09547702e+00],
       [ 7.00009482e-01,  7.04007878e+00,  1.28180975e+00],
       [-1.04859792e-01,  7.16112165e+00,  7.44607627e-01]])
```

(b) Muestra en una gráfica el comportamiento de las soluciones en el tiempo (i.e. grafica $x(t)$, $y(t)$ y $z(t)$).

[25]:
```
xdata = [w[i,0] for i in range(N)]
ydata = [w[i,1] for i in range(N)]
zdata = [w[i,2] for i in range(N)]
```

[38]:
```
plt.figure(figsize=(12, 12), dpi=80)
plt.plot(time, xdata, linewidth=2.5, linestyle="-",color="blue", label = "x(t)")
plt.plot(time, ydata, linewidth=2.5, linestyle="-",color="red", label = "y(t)")
plt.plot(time, zdata, linewidth=2.5, linestyle="-",color="green", label =␣
 ↪"z(t)")
plt.title("Atractor de Rössler", fontsize=40)
plt.xlabel('Tiempo', fontsize=20)
plt.ylabel('x,y,z', fontsize=20)
plt.legend(loc='best')
```

[38]:
```
<matplotlib.legend.Legend at 0x7f03c41b8df0>
```



(c) Muestra como se ve el **atractor de Rössler** en 3D (i.e. en el espacio).

[27]:
```
from mpl_toolkits.mplot3d.axes3d import Axes3D
```

[40]:
```
fig = plt.figure(figsize=(15, 10), dpi=80)
ax = fig.add_subplot(1,1,1, projection='3d')
ax.plot(xdata, ydata, zdata,color="green", label="Soluciones")
plt.title("Atractor de Rössler", fontsize=40)
plt.legend(loc='best')
ax.set_xlabel("x(t)", fontsize=20)
ax.set_ylabel("y(t)", fontsize=20)
ax.set_zlabel("z(t)", fontsize=20)
```

[40]: Text(0.5, 0, 'z(t)')