Alumno: Brian Alex Fuentes Acuña

Padron:101785

Clase 4: Linealización

Introducción

La primera clase, nos encontramos con el problema de Ogata y el puente grúa. Al plantear las ecuaciones diferenciales, notamos que aparecía un término debido a la fuerza de gravedad:

$$f + Mg - Ky - B\frac{dy}{dt} = M\frac{d^2y}{dt^2}$$

En la clase 1, habrás demostrado que este sistema no es lineal. Se denomina lineal afín. Requiere el tipo más simple de linealización que es un desplazamiento del origen de coordenadas, como debés haberlo resuelto (quizás intuitivamente):

$$w = y - y_{equilibrio}$$

Donde $y_{equilibrio}$ es tal que el sistema está estático, es decir: $\frac{dy}{dt} = 0$ y $\frac{d^2y}{dt^2} = 0$ y no hay excitación aplicada $f = 0$:

$$Mg - Ky_{equilibrio} = 0$$
$$y_{equilibrio} = \frac{Mg}{K}$$

Y con $w = y - \frac{Mg}{K}$ podés demostrar fácilmente que la ecuación diferencial se transforma en:

$$f + Kw - B\frac{dw}{dt} = M\frac{d^2w}{dt^2}$$

Si llevamos a una descripción del sistema original en variables de estado con $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{bmatrix} y & \frac{dy}{dt} \end{bmatrix}^T$, llegamos a:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -K/M & -B/M \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1/M \end{bmatrix} u + \begin{bmatrix} 0 \\ g \end{bmatrix}$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}$$

Podemos hallar el punto de equilibrio con $\dot{\mathbf{x}} = 0$, para $u = 0$. De aquí sale:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -K/M & -B/M \end{bmatrix} \mathbf{x}_{equilibrio} + \begin{bmatrix} 0 \\ g \end{bmatrix}$$

que da:

$$\mathbf{x}_{equilibrio} = \begin{bmatrix} \frac{Mg}{K} \\ 0 \end{bmatrix}$$

y

$$y_{equilibrio} = x_{1_{equilibrio}} = \frac{Mg}{K}$$

Que es el mismo resultado, y por lo tanto es equivalente reemplazar por:

$$\mathbf{z} = \mathbf{x} - \mathbf{x}_{equilibrio}$$

Para que el sistema quede lineal:

$$\dot{\mathbf{z}} = \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -K/M & -B/M \end{bmatrix} \mathbf{z} + \begin{bmatrix} 0 \\ 1/M \end{bmatrix} u$$

Sin embargo, para obtener la solución original, debemos corregir:

$$\mathbf{x} = \mathbf{z} + \mathbf{x}_{equilibrio} = \mathbf{z} + \begin{bmatrix} \frac{Mg}{K} \\ 0 \end{bmatrix}$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{z} + \mathbf{y}_{equilibrio}$$

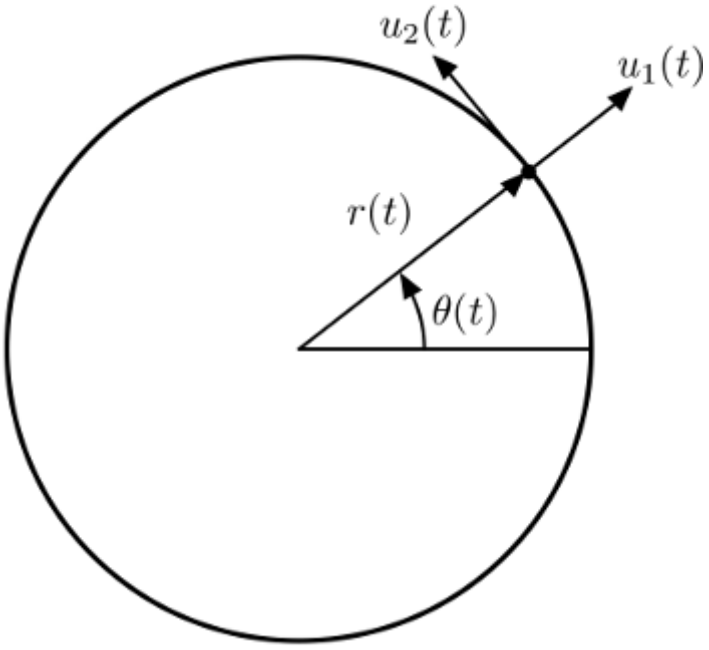
Con esto mostramos que podemos hacer la linealización tanto en la ecuación diferencial original, como en la descripción en variables de estado.

Ahora que vimos el caso más simple, podemos pasar al caso general más complejo.

Linealización alrededor de una trayectoria

Vayamos directo a la aplicación con un ejemplo: el modelo muy simplificado de un satélite en órbita circular.

Partimos de un modelo de una masa puntual atraída por la tierra en el origen de coordenadas polares de acuerdo a la siguiente figura:



Modelo de satélite en órbita

Suponemos que el satélite tiene dos formas de producir acción de control de su órbita, a través de propulsiones radial $u_1(t)$ y tangencial $u_2(t)$.

Las ecuaciones que dominan este movimiento son:

$$\ddot{r}(t) = r(t)\dot{\theta}^2(t) - \frac{\beta}{r^2(t)} + u_1(t)$$
$$\ddot{\theta}(t) = -2\frac{\dot{r}(t)\dot{\theta}(t)}{r(t)} + \frac{u_2(t)}{r(t)}$$

Donde β es una constante y $y(t) = \begin{bmatrix} r(t) \\ \theta(t) \end{bmatrix}$ define la salida con las variables de interés.

Para llegar a un modelo lineal, nos interesa plantear que el satélite va a llevar una órbita circular nominal y alrededor de ella van a existir perturbaciones, que vamos a tratar de compensar mediante un controlador. Vamos a suponer entonces, que esos desplazamientos van a ser suficientemente pequeños para buscar un modelo lineal equivalente.

Primero, tenemos que buscar las soluciones o trayectorias nominales del sistema. Serán de la forma de órbita circular ($r(t) =$ cte. y $\theta(t) =$ cte.) . Y se dará para:

$$\tilde{u}_1 = \tilde{u}_2 = 0 \quad \forall t \geq 0$$

Con condiciones iniciales:

- $r(0) = r_0$
- $\dot{r}(0) = 0$
- $\theta(0) = \theta_0$
- $\dot{\theta}(0) = \omega_0$

Ejercicio Demostrar que la solución nominal es de la forma:

- $\tilde{r}(t) = r_0$
- $\tilde{\theta} = \omega_0 t + \theta_0$

Y que llegás a la relación:

$$\omega_0 = \sqrt{\frac{\beta}{r_0^3}}$$

Definimos entonces un modelo en variables de estado a partir de las ecuaciones de movimiento antes expuestas. Proponemos:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} r(t) \\ \dot{r}(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix}$$

Ejercicio Agarrá lápiz y papel y llegá a la descripción en variables de estado. Debería darte:

$$\dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ x_1 x_4^2 - \beta / x_1^2 + u_1 \\ x_4 \\ -2 x_2 x_4 / x_1 + u_2 / x_1 \end{bmatrix}$$

y

$$\mathbf{y} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{C}} \mathbf{x}$$

Es decir, tenemos un sistema de la forma:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

Donde $\mathbf{f}$ es una función vectorial no lineal. Notar sin embargo, que la ecuación de salida si tiene la forma de sistema lineal $y = C\mathbf{x}$.

En forma vectorial, la solución nominal tiene la siguiente forma:

$$\tilde{\mathbf{u}}(t) = \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\tilde{\mathbf{x}}(t) = \begin{bmatrix} r_0 \\ 0 \\ \omega_0 t + \theta_0 \\ \omega_0 \end{bmatrix}$$

$$\tilde{\mathbf{x}}(0) = \begin{bmatrix} r_0 \\ 0 \\ \theta_0 \\ \omega_0 \end{bmatrix}$$

Ahora empieza la diversión. Como vimos en la teórica, para un pequeño incremento en las señales de entrada, respecto de la solución o trayectoria nominal:

$$\mathbf{u}_\delta(t) = \mathbf{u} - \tilde{\mathbf{u}}$$

suponemos que se produce un pequeño desplazamiento de la trayectoria:

$$\mathbf{x}_\delta(t) = \mathbf{x} - \tilde{\mathbf{x}}$$

de manera tal que es posible aplicar una aproximación de Taylor de primer orden para relacionarlas. Que sale de:

$$\dot{\mathbf{x}} = \mathbf{f}\left(\tilde{\mathbf{x}} + \mathbf{x}_\delta(t), \tilde{\mathbf{u}} + \mathbf{u}_\delta(t)\right) \simeq \mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{x}_\delta(t) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{u}_\delta(t)$$

En donde:

$$\mathbf{A}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$$

y

$$\mathbf{B}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$$

Son las matrices Jacobianas con elementos $m_{i,j} = \frac{\partial f_i}{\partial v_j}$. Notar que como la trayectoria nominal es solución del sistema:

$$\dot{\tilde{\mathbf{x}}} = \mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$$

queda:

$$\dot{\mathbf{x}}_\delta = \dot{\mathbf{x}} - \dot{\tilde{\mathbf{x}}} \simeq \mathbf{A}(t)\mathbf{x}_\delta(t) + \mathbf{B}(t)\mathbf{u}_\delta(t)$$

Ejercicio: Es tu turno de tomar estas ecuaciones generales y aplicarlas al satélite en órbita.

RTA

Se hará solamente la matriz A, el resto sale por inspeccion. en todos los casos se aplica el jacobiano, salvo para la matriz C en donde al ser lineal con la salida no tiene sentido aplicarlo. Entonces aplicando el jacobiano a $\dot{x}$ se llega

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ x_4^2 + 2\beta x_1^3 & 0 & 0 & 2x_1 x_4 \\ 0 & 0 & 0 & 1 \\ 0 & -2x_1/x_4 & 0 & 0 \end{bmatrix}}_{\mathbf{A}}$$

Evaluada en $\tilde{\mathbf{x}}(0), \tilde{\mathbf{u}}(0)$

Deberías llegar a:

$$\dot{\mathbf{x}}_\delta(t) = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 3\omega_0^2 & 0 & 0 & 2r_0\omega_0 \\ 0 & 0 & 0 & 1 \\ 0 & -2\omega_0/r_0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \mathbf{x}_\delta(t) + \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1/r_0 \end{bmatrix}}_{\mathbf{B}} \mathbf{u}_\delta(t)$$

$$\mathbf{y}_\delta(t) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{C}} \mathbf{x}_\delta$$

Al ser lineal la ecuación de salida, el modelo incremental lleva la misma matriz de salida C. En un caso general, recordemos que hay que linealizar también dicha ecuación. O sea, dada:

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}, \mathbf{u})$$

Y una solución nominal con salida:

$$\tilde{\mathbf{y}}(t) = \mathbf{g}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$$

Con la aproximación de Taylor de primer orden, bajo la hipótesis de pequeños desplazamientos:

$$\mathbf{y}(t) = \mathbf{g}\left(\tilde{\mathbf{x}} + \mathbf{x}_\delta(t), \tilde{\mathbf{u}} + \mathbf{u}_\delta(t)\right) \simeq \mathbf{g}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) + \frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{x}_\delta(t) + \frac{\partial \mathbf{g}}{\partial \mathbf{u}}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{u}_\delta(t)$$

O sea:

$$\mathbf{y}_\delta(t) = \mathbf{g}\left(\tilde{\mathbf{x}} + \mathbf{x}_\delta(t), \tilde{\mathbf{u}} + \mathbf{u}_\delta(t)\right) - \mathbf{g}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = \underbrace{\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})}_{\mathbf{C}(t)}\mathbf{x}_\delta(t) + \underbrace{\frac{\partial \mathbf{g}}{\partial \mathbf{u}}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})}_{\mathbf{D}(t)}\mathbf{u}_\delta(t)$$

Tarea:

1. Simulá y compará respuestas del modelo no lineal y el lineal alrededor de la trayectoria nominal, órbita geoestacionaria, ante una pequeña perturbación en la acción $u_1(t)$ en forma de pulso de pequeña amplitud. Superponé las dos simulaciones en gráficos polares para verificar si la aproximación lineal es buena.

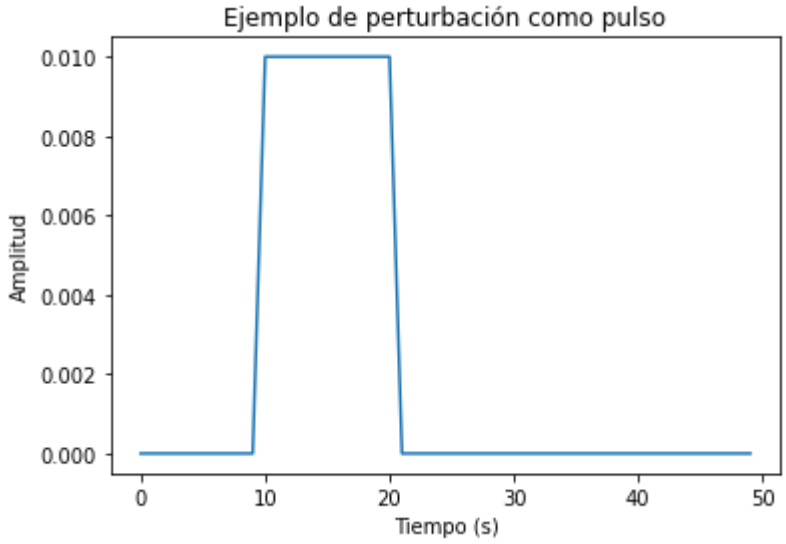
In [84]:

```
!pip install control
import numpy as np
import scipy as sp
import control as ctrl
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: control in /usr/local/lib/python3.7/dist-pa
ckages (0.9.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-pack
ages (from control) (1.19.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-pack
ages (from control) (1.4.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist
-packages (from control) (3.2.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->control) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python
3.7/dist-packages (from matplotlib->control) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/di
st-packages (from matplotlib->control) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/pyth
on3.7/dist-packages (from matplotlib->control) (2.8.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packag
es (from cycler>=0.10->matplotlib->control) (1.15.0)
```

In [85]:

```
# Para mostrar el tipo de perturbación les dejo un ejemplo:
tiempo_inicial = 0
tiempo_final =3600 # Ajustar este tiempo
paso_tiempo = 1 # Ajustar este tiempo
tiempo = np.arange(tiempo_inicial, tiempo_final, paso_tiempo)
amplitud_perturbacion = 0.01 # Probar valores
duracion_perturbacion = 10
inicio_perturbacion = 10

#Grafico la perturbacion en el rango de 0 a 50 seg.
u_1 = amplitud_perturbacion*((tiempo >= inicio_perturbacion) & (tiempo <= inicio_pertur
bacion + duracion_perturbacion))
plt.plot(tiempo[0:50], u_1[0:50])
plt.title('Ejemplo de perturbación como pulso')
plt.ylabel('Amplitud')
plt.xlabel('Tiempo (s)')
plt.show()
#-------------------------------------------------------------------------------
```



A continuación se definiran los variables a usar, para corroborar que las simulaciones esten bien hechas primero se debe simular sin perturbaciones en la entrada $u_1(t) = u_2(t) = 0$. Luego se con perturbacion pero solo en **u_1**.

In [86]:

```
# 1. Linealización
r0 = 4.22e7 # m para órbita geoestacionaria
w0 = 7.291e-5 # rad/s para órbita geoestacionaria
theta0 = 0

beta= r0**3 * w0**2
x0 = [r0, 0, theta0, w0]
x0d = [0, 0, 0, 0]
#se debe simular la orbita en un dia, pasados a segundos
dia_seg= 24*60*60 # segundos
```

In [87]:

```
A= np.array([ [0, 1, 0, 0], [2*w0**2, 0, 0, 2*r0*w0], [0, 0, 0, 1], [0, -2*w0/r0, 0, 0]
])
autval, autvect = np.linalg.eig(A)
print("autovalores:", autval)
#Ante la minima perturbacion oscila.
```

```
autovalores: [0.+0.j         0.+0.00010311j 0.-0.00010311j 0.+0.j        ]
```

Se procede a definir el sistema lineal mostrado anteriormente, para dos casos de interes.

El primero será sin perturbacion **f_lin()** y el otro será con la perturbación en la entrada **f_lin_pert()** en ambos casos con condiciones iniciales nulas ($x_{0d} = [0, 0, 0, 0]$)

In [88]:

```
#-------------------------------------------------------------------------
#se define la perturbacion como funcion:
"""
Por comodidad los parametros que definen a la perturbacion seran los mismos mostrados
mas arriba, esto para ver graficamente como es la perturbacion.
"""
def u_1(t):
    return amplitud_perturbacion*((t >= inicio_perturbacion) & (t <= inicio_perturbacio
n + duracion_perturbacion))

# Definir el sistema lineal incremental y simular.
"""
f_lin(t, x) Es mi funcion en donde se le carga el modelo en espacio de estados,
al mismo hay que cambiarle u_1d y u_2d quienes son las perturbaciones de entrada.
"""
def f_lin(t, x):
    u_1d= 0#u_1
    u_2d= 0
    f1= x[1]
    f2= 3*w0**2 * x[0] + 2*r0*w0 * x[3] + u_1d#(t)
    f3= x[3]
    f4 = (-2*w0/r0 ) * x[1] + u_2d/r0
    return [f1.item(), f2.item(), f3.item(), f4.item()]

def f_lin_pert(t, x):
    u_2d= 0
    f1= x[1]
    f2= 3*w0**2 * x[0] + 2*r0*w0 * x[3] + u_1(t)
    f3= x[3]
    f4 = (-2*w0/r0 ) * x[1] + u_2d/r0
    return [f1.item(), f2.item(), f3.item(), f4.item()]

#Integro de funcion en 24hs (pasados a segundos) y obtengo la solucion de cada variable
de estado.
sol_lin = sp.integrate.solve_ivp(f_lin, (0, dia_seg), x0d, t_eval= np.arange(0, dia_seg
, 1), dense_output=True, rtol=1e-5, atol=1e-5)
sol_lin_pert = sp.integrate.solve_ivp(f_lin_pert, (0, dia_seg), x0d, t_eval= np.arange(
0, dia_seg, 1), dense_output=True, rtol=1e-5, atol=1e-5)

#Se obtienen las variables de fase para graficar.
posicion_r_lin=       sol_lin.y[0]
posicion_theta_lin=   sol_lin.y[2]
velocidad_r_lin=      sol_lin.y[1]
velocidad_theta_lin=  sol_lin.y[3]
tiempo_lin =          sol_lin.t

posicion_r_lin_pert=       sol_lin_pert.y[0]
posicion_theta_lin_pert=   sol_lin_pert.y[2]
velocidad_r_lin_pert=      sol_lin_pert.y[1]
velocidad_theta_lin_pert=  sol_lin_pert.y[3]
tiempo_lin_pert =          sol_lin_pert.t
```

Se procede a definir de forma analoga al sistema no lineal. Para este se tiene que usar las condiciones iniciales de la solcion nominal ($x_0 = [r_0, 0, \theta_0, w_0]$)

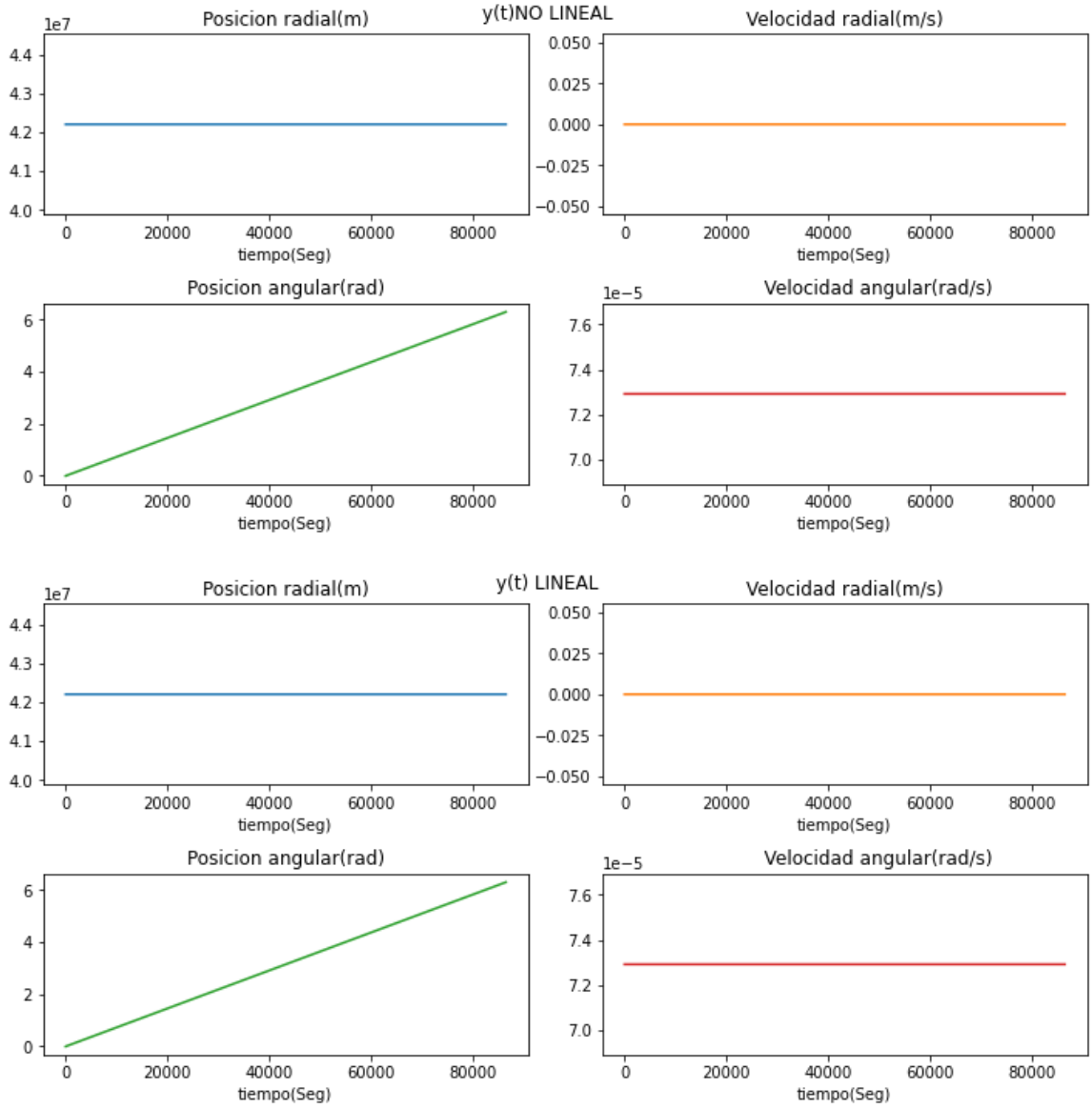
In [89]:

```
# Definir el sistema no lineal y simular. Tip: para simularlo vas a necesitar algo como
esto:
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html#s
cipy.integrate.solve_ivp
def u(t):
  return 0

# el sistema no lineal sin perturbacion de entrada
"""
  La simulación del sistema no lineal sin perturbar
  tendra que dar la respuesta nominal
"""
def f_nl(t, x):
  #f2= x1*x4^2 - beta/x1^2 + u_1
  #f4= -2*x2*x4/x1 + u_2/x1
    u_1=0 #comentar para poner perturbacion
    u_2= 0
    f1= x[1]
    f2= x[0]*x[3]**2 - beta/x[0]**2 + u_1#(t)
    f3= x[3]
    f4 = -2*x[1]*x[3]/x[0] + u_2/x[0]
    return [f1.item(), f2.item(), f3.item(), f4.item()]

#el sistema no lineal con la perturbacion como entrada
def f_nl_pert(t, x):
  #f2= x1*x4^2 - beta/x1^2 + u_1
  #f4= -2*x2*x4/x1 + u_2/x1
    u_2= 0
    f1= x[1]
    f2= x[0]*x[3]**2 - beta/x[0]**2 + u_1(t)
    f3= x[3]
    f4 = -2*x[1]*x[3]/x[0] + u_2/x[0]
    return [f1.item(), f2.item(), f3.item(), f4.item()]

#-obtengo las respuestas al sistema no lineal
sol      = sp.integrate.solve_ivp(f_nl, (0, dia_seg), x0, t_eval= np.arange(0, dia_seg,
1), dense_output=True, rtol=1e-5, atol=1e-5)
sol_pert = sp.integrate.solve_ivp(f_nl_pert, (0, dia_seg), x0, t_eval= np.arange(0, dia
_seg, 1), dense_output=True, rtol=1e-5, atol=1e-5)

posicion_r=       sol.y[0]
posicion_theta=   sol.y[2]
velocidad_r=      sol.y[1]
velocidad_theta=  sol.y[3]
tiempo =          sol.t

posicion_r_pert=      sol_pert.y[0]
posicion_theta_pert=  sol_pert.y[2]
velocidad_r_pert=     sol_pert.y[1]
velocidad_theta_pert= sol_pert.y[3]
tiempo_pert =         sol_pert.t
```

Luego se procede a hacer los graficos de las variables de fase calculada para observar las respuestas en funcion del tiempo. Estas respuestas son de los sistemas sin perturbar, observar que son similares sino es que iguales. Ademas de que se obtienen las señales esperadas, es decir con radio constante(r_0) y el angulo (θ_0) varia de forma lineal, con velocidad constante w_0.

In [90]:

```
# Graficar respuestas superpuestas de y(t) para el sistema lineal y el sistema no linea
l

##Graficos de y(t) no lineal, sin perturbar
fig, axs = plt.subplots(2, 2, figsize=(10,5))
fig.suptitle('y(t)NO LINEAL')
axs[0, 0].plot(tiempo, posicion_r )
axs[0, 0].set_title('Posicion radial(m)')
axs[0, 1].plot(tiempo, velocidad_r , 'tab:orange')
axs[0, 1].set_title('Velocidad radial(m/s)')
axs[1, 0].plot(tiempo, posicion_theta , 'tab:green')
axs[1, 0].set_title('Posicion angular(rad)')
axs[1, 1].plot(tiempo, velocidad_theta , 'tab:red')
axs[1, 1].set_title('           Velocidad angular(rad/s)')

for ax in axs.flat:
    ax.set(xlabel='tiempo(Seg)')

# Hide x labels and tick labels for top plots and y ticks for right plots.
fig.tight_layout()
########################################
##Graficos de y(t) LINEAL, sin perturbar.

fig, axs = plt.subplots(2, 2, figsize=(10,5))
fig.suptitle('y(t) LINEAL')
axs[0, 0].plot(tiempo_lin, posicion_r_lin + r0)
axs[0, 0].set_title('Posicion radial(m)')
axs[0, 1].plot(tiempo_lin, velocidad_r_lin, 'tab:orange')
axs[0, 1].set_title('Velocidad radial(m/s)')
axs[1, 0].plot(tiempo_lin, posicion_theta_lin + (w0*tiempo_lin + theta0), 'tab:green')
axs[1, 0].set_title('Posicion angular(rad)')
axs[1, 1].plot(tiempo_lin, velocidad_theta_lin + w0, 'tab:red')
axs[1, 1].set_title('           Velocidad angular(rad/s)')

for ax in axs.flat:
    ax.set(xlabel='tiempo(Seg)')

# Hide x labels and tick labels for top plots and y ticks for right plots.
fig.tight_layout()
```



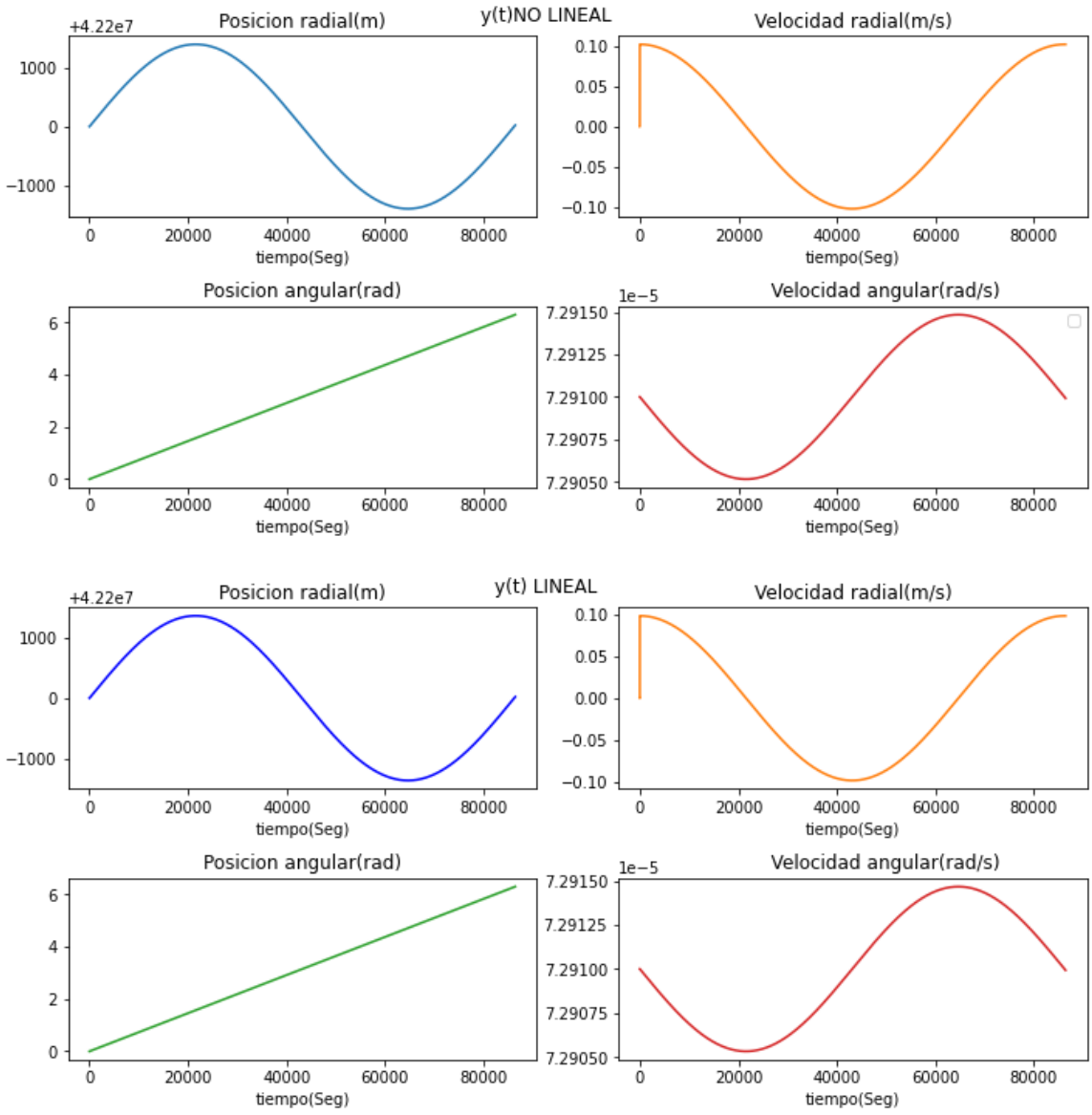
¿Observaciones? ¿Qué debería dar?

Luego hay que aplicar la perturbacion a la entrada para ambos sistemas, lineal y no lineal. Aca está lo interesante puesto que, en los graficos, al no poder superponerlos (viasulente legibles) parecen ser muy similares, y de echo lo son pues no hay mucha diferencia entre ellos por ser una buena aproximacion para la perturbación aplicada.

In [91]:

```
#VUELVO A GRAFICAR LO MISMO PERO CON PERTURBACION EN LA ENTRADA.
##Graficos de y(t) no lineal, perturbado
fig, axs = plt.subplots(2, 2, figsize=(10,5))
fig.suptitle('y(t)NO LINEAL')
axs[0, 0].plot(tiempo_pert, posicion_r_pert)
axs[0, 0].set_title('Posicion radial(m)')
axs[0, 1].plot(tiempo_pert, velocidad_r_pert , 'tab:orange')
axs[0, 1].set_title('Velocidad radial(m/s)')
axs[1, 0].plot(tiempo_pert, posicion_theta_pert , 'tab:green')
axs[1, 0].set_title('Posicion angular(rad)')
axs[1, 1].plot(tiempo_pert, velocidad_theta_pert , 'tab:red')
axs[1, 1].set_title('           Velocidad angular(rad/s)')

plt.legend()

for ax in axs.flat:
    ax.set(xlabel='tiempo(Seg)')

# Hide x labels and tick labels for top plots and y ticks for right plots.
fig.tight_layout()
########################################
##Graficos de y(t) LINEAL, perturbado.

fig, axs = plt.subplots(2, 2, figsize=(10,5))
fig.suptitle('y(t) LINEAL')
axs[0, 0].plot(tiempo_lin_pert, posicion_r_lin_pert + r0, 'b-',label= 'lin')
axs[0, 0].set_title('Posicion radial(m)')
axs[0, 1].plot(tiempo_lin_pert, velocidad_r_lin_pert, 'tab:orange')
axs[0, 1].set_title('Velocidad radial(m/s)')
axs[1, 0].plot(tiempo_lin_pert, posicion_theta_lin_pert + (w0*tiempo_lin_pert + theta0
), 'tab:green')
axs[1, 0].set_title('Posicion angular(rad)')
axs[1, 1].plot(tiempo_lin_pert, velocidad_theta_lin_pert + w0, 'tab:red')
axs[1, 1].set_title('           Velocidad angular(rad/s)')

for ax in axs.flat:
    ax.set(xlabel='tiempo(Seg)')

# Hide x labels and tick labels for top plots and y ticks for right plots.
fig.tight_layout()
```

```
No handles with labels found to put in legend.
```



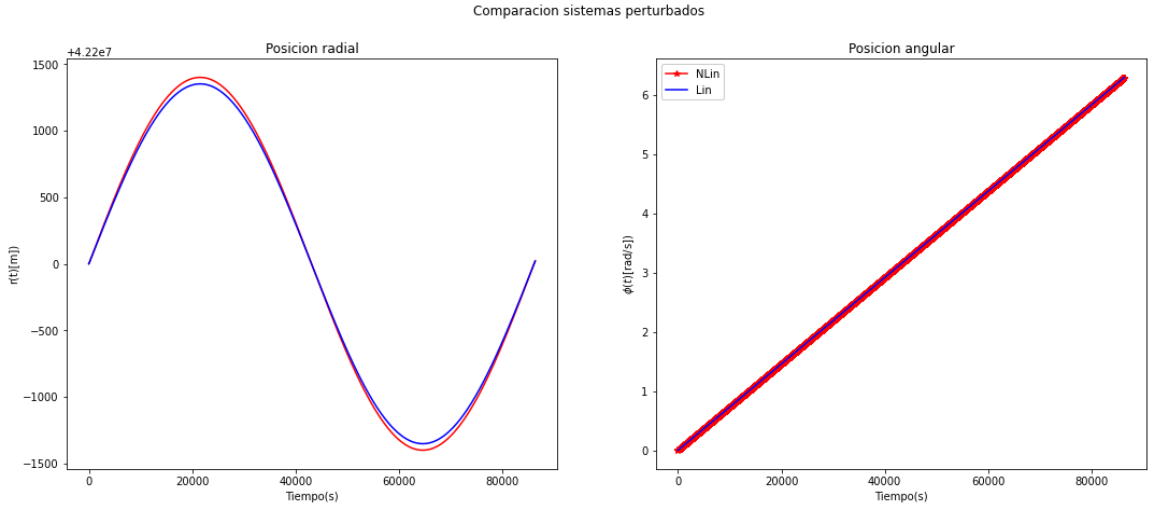
y finalmente se grafican las posciones radiales y angulares para ver si difieren y como se esperaba así lo hacen. ¿Cómo lo ves? A simple vista da igual.

puesto que el sistema lineal es una "linealización" del sistema no lineal, existiran diferencias entre ellas, la lineal trata de aproxima a la no lineal y segun como se la perturbe existiran rangos en que la aproximación ya no valga. Observar también que las respuestas a la perturbaciones oscilan.

aproximación lineal?

In [92]:

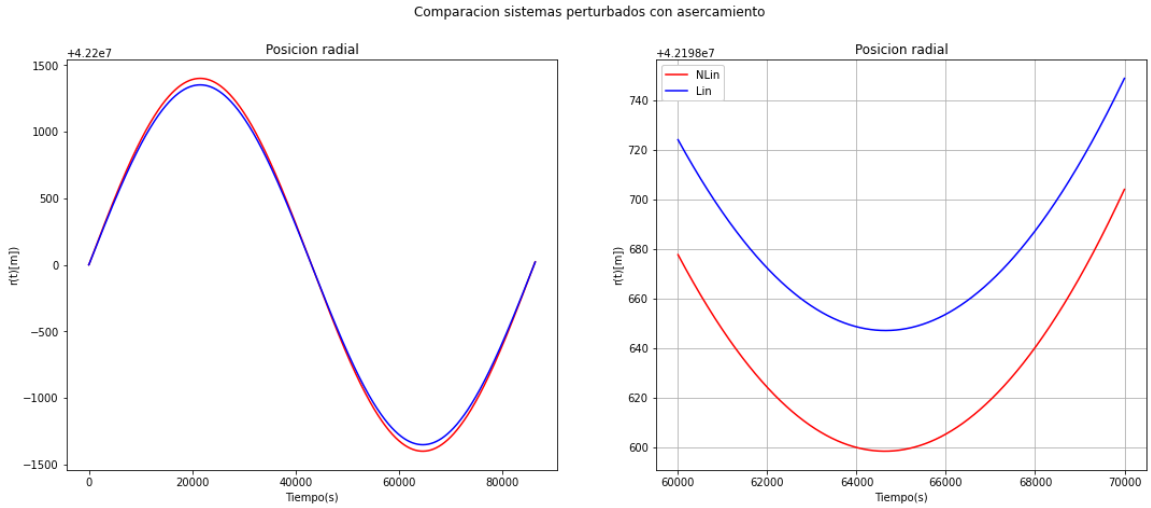
```
#-obtengo las respuesta al sistema lineal
# se suma las condiciones iniciales de la solucion nominal x0m=[r0, 0, w0*t + theta0, w
0]
fig, ((ax1, ax2)) = plt.subplots(1, 2, figsize=(18,7))

fig.suptitle('Comparacion sistemas perturbados')
ax1.plot(tiempo_pert, posicion_r_pert,'r-', label= 'NLin')
ax1.plot(tiempo_lin_pert, posicion_r_lin_pert + r0,'b-', label= 'Lin')
ax1.set_title("Posicion radial")
#plt.legend()
ax2.plot(tiempo_pert, posicion_theta_pert,'r*-', label= 'NLin')
ax2.plot(tiempo_lin_pert, posicion_theta_lin_pert + (w0*tiempo_lin_pert + theta0),'b-',
label= 'Lin')
ax2.set_title("Posicion angular")

ax1.set(xlabel='Tiempo(s)', ylabel='r(t)[m])')
ax2.set(xlabel='Tiempo(s)', ylabel='$\phi(t)$[rad/s])')
plt.legend(loc='upper left',framealpha=1,  frameon=True)
plt.show()
fig.tight_layout()
```



Como es se perturba radialmente, se observa que hay diferencias el la posicion radial para los dos sistemas, entonces se procede a hacer una ampliacion de la zona en done la diferencia es mayor y ver cuanto se separa respecto de la no lineal.

In [93]:

```
#Comparacion sistemas perturbados con asercamiento'
# se suma las varibles de estado de la solucion nominal x_m=[r0, 0, w0*t + theta0, w0]
fig, ((ax1, ax2)) = plt.subplots(1, 2, figsize=(18,7))

fig.suptitle('Comparacion sistemas perturbados con asercamiento')
ax1.plot(tiempo_pert, posicion_r_pert,'r-', label= 'NLin')
ax1.plot(tiempo_lin_pert, posicion_r_lin_pert + r0,'b-', label= 'Lin')
ax1.set_title("Posicion radial")
#plt.legend()
ax2.plot(tiempo_pert[60000:70000], posicion_r_pert[60000:70000],'r-', label= 'NLin')
ax2.plot(tiempo_lin_pert[60000:70000], posicion_r_lin_pert[60000:70000] + r0,'b-', labe
l= 'Lin')
ax2.set_title("Posicion radial")

ax1.set(xlabel='Tiempo(s)', ylabel='r(t)[m])')
ax2.set(xlabel='Tiempo(s)', ylabel='r(t)[m])')
plt.legend(loc='upper left',framealpha=1,  frameon=True)
plt.grid()
plt.show()
fig.tight_layout()
```



Se tiene aproximadamente 600m para el sistema lineal y 650m para el no lineal, esto implica una diferencia porcentual del **8.3%** para $u_1 = 0.01\frac{m}{s^2}$.

Lo que es aceptable si se pide una diferencia menor al 10% lo que es usual en la ingenieria, aunque dependiendo el caso de estudio se podria pedir menos.

?

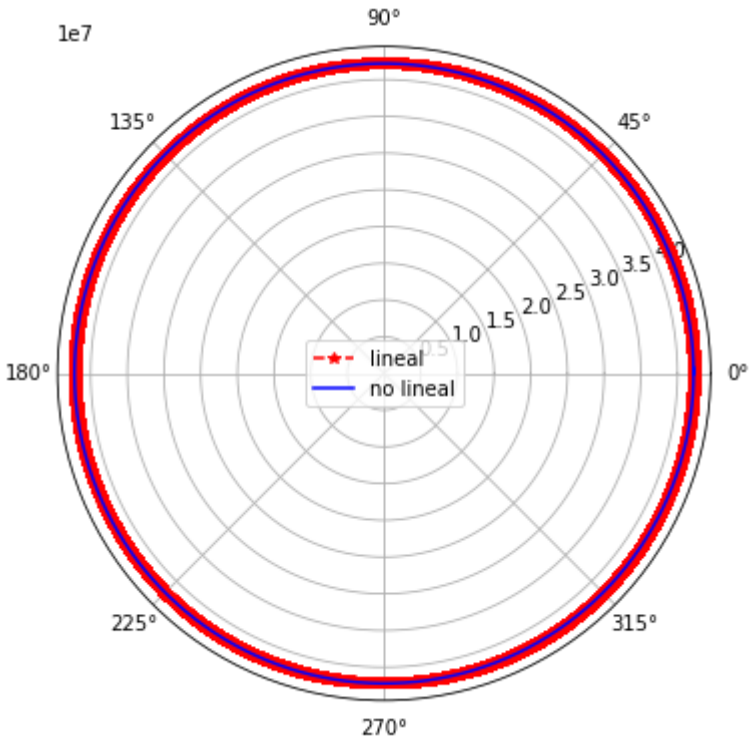
Me parece que depende de qué se esté estudiando.

Se procede a hacer los graficos pedidos en coordenadas polares.

el primero es sin perturbacion en la entrada, notar que no se pueden observa diferencias a simple vista

In [94]:

```
#Grafico la trayectoria del sistema linealizado con el no lineal sin perturbar.
fig = plt.figure(figsize=(10,6))
ax = fig.add_subplot(111, projection="polar")
ax.plot(posicion_theta_lin + (w0*tiempo_lin + theta0), posicion_r_lin + r0, 'r*-',labe
l= 'lineal')
ax.plot(posicion_theta, posicion_r,'b-', label= 'no lineal')
plt.legend()

plt.show()
```

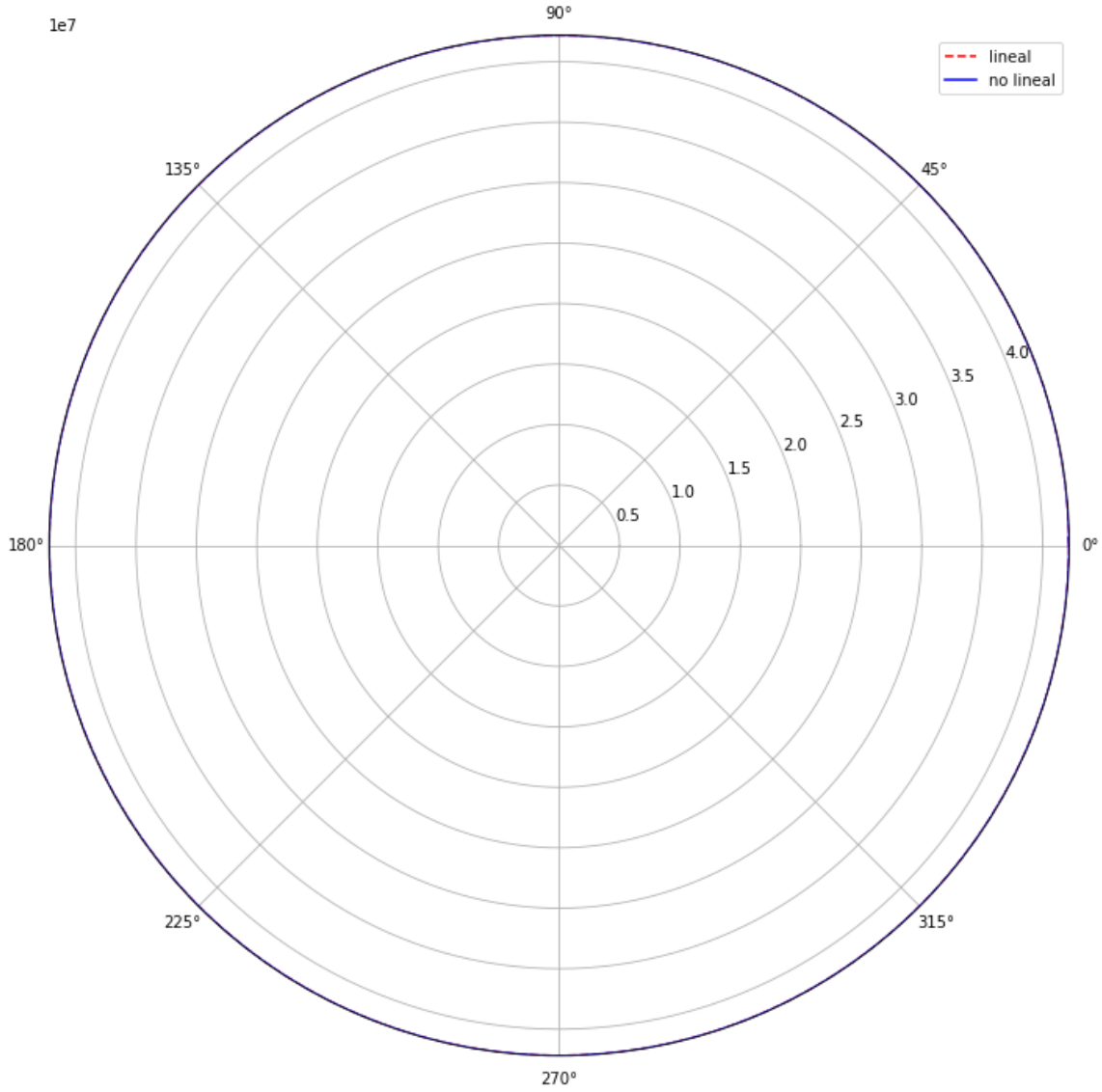


Genera un trazo muy grueso.

Cuando se lo perturba, se pueden diferenciar apenas las distintas curvas, pero ya muestra que no son exactamente igual por lo que es coherente con lo anteriormente mensionado.

In [95]:

```
#Grafico la trayectoria del sistema linealizado con el no lineal perturbado.

fig = plt.figure(figsize=(20,12))
ax = fig.add_subplot(111, projection="polar")
ax.plot(posicion_theta_lin_pert + (w0*tiempo_lin_pert + theta0), posicion_r_lin_pert +
r0, 'r--',label= 'lineal')
ax.plot(posicion_theta_pert, posicion_r_pert,'b-', label= 'no lineal')
plt.legend()
plt.show()
```

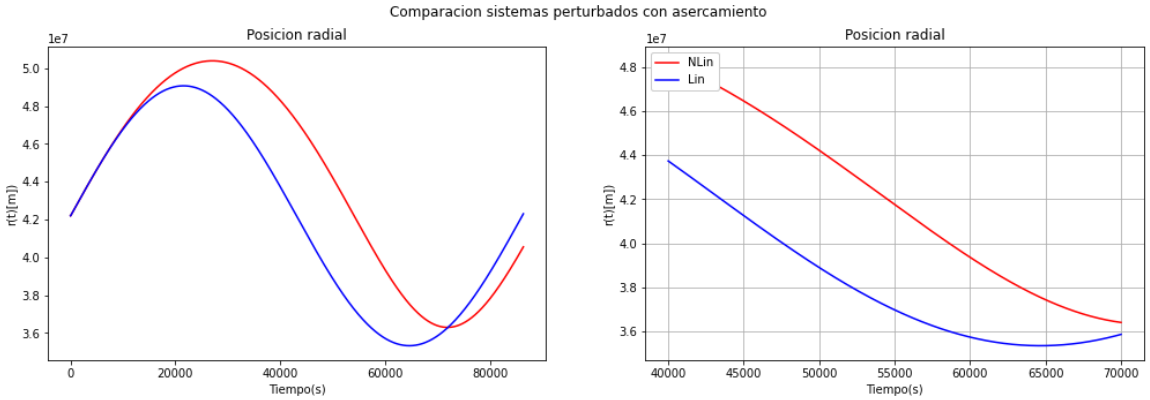


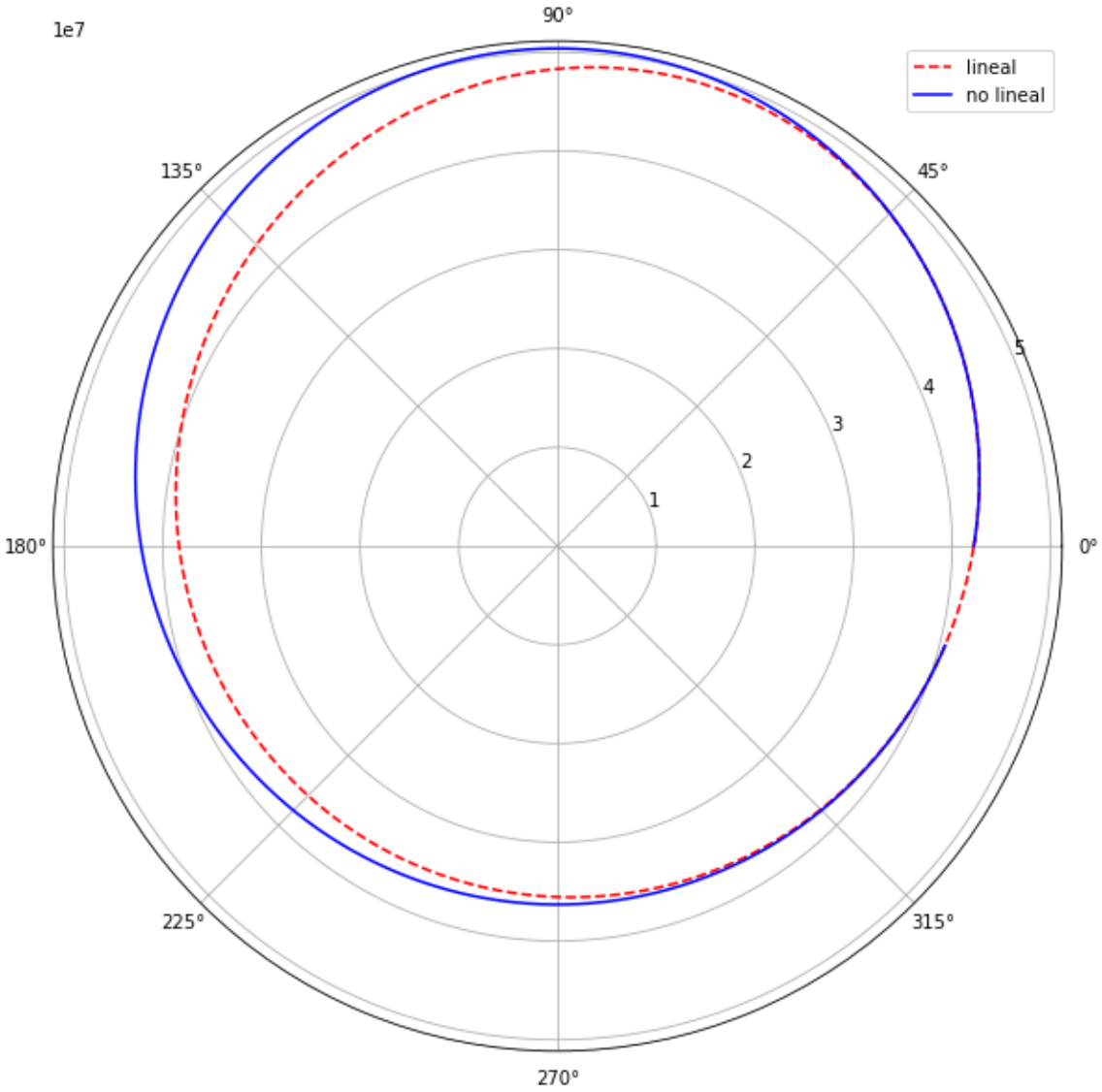
1. Cambiá la amplitud del pulso y volvé a simular y graficar para un caso en que creas que ya no es válida la aproximación lineal. **RTA:**

Claramente se puede observar que con una perturbación radial de $u_1 = 50[m/s^2]$ ya el modelo lineal no es válido. incluso con el grafico polar el cual por temas de escalas no se lograba apreciar diferencias anteriormente, en este caso si nota claramente si se observan los valores hay una diferencia porcentual del 12%, mas alla de que diferencial porcentual se elija como aceptable, un 12% es mucho y se vé que el modelo lineal ya no es valido.

In [97]:

```
amplitud_perturbacion = 50 # Probar valores

sol_lin_pert = sp.integrate.solve_ivp(f_lin_pert, (0, dia_seg), x0d, t_eval= np.arange(
0, dia_seg, 1), dense_output=True, rtol=1e-5, atol=1e-5)
sol_pert = sp.integrate.solve_ivp(f_nl_pert, (0, dia_seg), x0, t_eval= np.arange(0, dia
_seg, 1), dense_output=True, rtol=1e-5, atol=1e-5)

#Se obtienen las variables de fase para graficar.

posicion_r_lin_pert=       sol_lin_pert.y[0]
posicion_theta_lin_pert=   sol_lin_pert.y[2]
velocidad_r_lin_pert=      sol_lin_pert.y[1]
velocidad_theta_lin_pert=  sol_lin_pert.y[3]
tiempo_lin_pert =          sol_lin_pert.t

posicion_r_pert=      sol_pert.y[0]
posicion_theta_pert=  sol_pert.y[2]
velocidad_r_pert=     sol_pert.y[1]
velocidad_theta_pert= sol_pert.y[3]
tiempo_pert =         sol_pert.t
#_____________________________________-

#-obtengo las respuesta al sistema lineal
# se suma las condiciones iniciales de la solucion nominal x0m=[r0, 0, w0*t + theta0, w
0]
fig, ((ax1, ax2)) = plt.subplots(1, 2, figsize=(17,5))

fig.suptitle('Comparacion sistemas perturbados con asercamiento')
ax1.plot(tiempo_pert, posicion_r_pert,'r-', label= 'NLin')
ax1.plot(tiempo_lin_pert, posicion_r_lin_pert + r0,'b-', label= 'Lin')
ax1.set_title("Posicion radial")
#plt.legend()
ax2.plot(tiempo_pert[40000:70000], posicion_r_pert[40000:70000],'r-', label= 'NLin')
ax2.plot(tiempo_lin_pert[40000:70000], posicion_r_lin_pert[40000:70000] + r0,'b-', labe
l= 'Lin')
ax2.set_title("Posicion radial")

ax1.set(xlabel='Tiempo(s)', ylabel='r(t)[m])')
ax2.set(xlabel='Tiempo(s)', ylabel='r(t)[m])')
plt.legend(loc='upper left',framealpha=1,  frameon=True)
plt.grid()
plt.show()
fig.tight_layout()

##########################################################################
##########################################################################
#Grafico la trayectoria del sistema linealizado con el no lineal perturbado.

fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection="polar")
ax.plot(posicion_theta_lin_pert + (w0*tiempo_lin_pert + theta0), posicion_r_lin_pert +
r0, 'r--',label= 'lineal')
ax.plot(posicion_theta_pert, posicion_r_pert,'b-', label= 'no lineal')
plt.legend()
plt.show()
```





1. ¿Qué propondrías hacer para medir si la aproximación es buena en cada caso?

   RTA:

   Por lo general se adopta un 10% de error entre la solucion y l aproximacion lineal, por lo que si viendo el grafico se puede ver que en algun punto hay una diferencia porcentual mayor al 10% ese es un punto donde ya no es aceptable la aproximacion lineal. Porsupuesto esto depende del caso de estudio, 10% para un satelite puede ser inaceptable.