Uso de Python como herramienta de enseñanza y aprendizaje de la carta de Smith

Jesús Rodríguez Camacho

Dpto. Física Aplicada

Alfonso Salinas Extremera

Dpto. Electromagnetismo y Física de la Materia

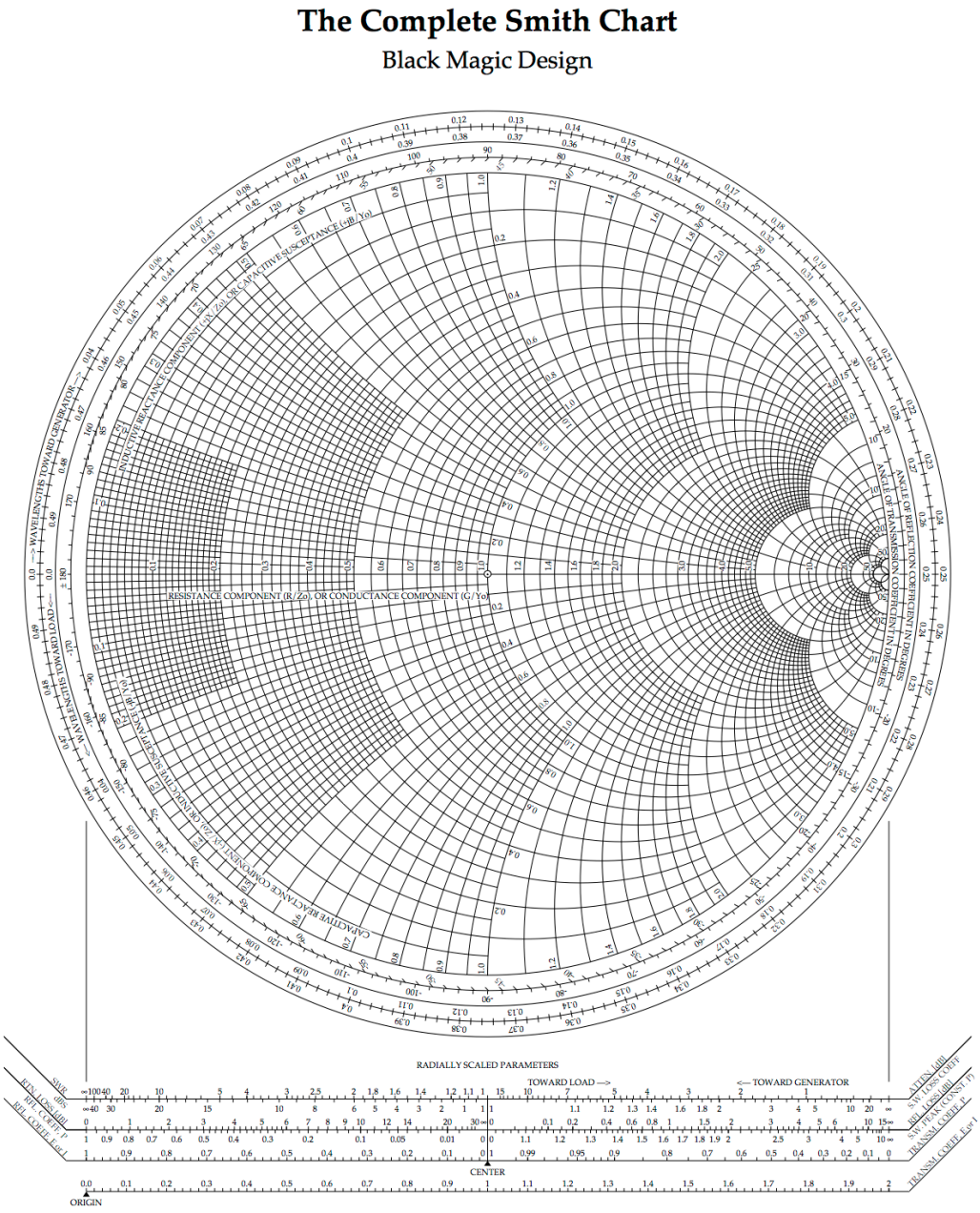
Asignatura: Transmisión de Ondas de 2º de Ingeniería de Tecnologías de Telecomunicación

El alumnado de esta asignatura:

- Ha manejado programas como Mathematica, Maxima, Matlab, Excel, ...
- Se están iniciando en la programación en C
- Saben manejar el ordenador, pero **no lo usan** como herramienta de cálculo
- Prefieren la calculadora, aún en cálculos largos y laboriosos
- Usan Excel como herramienta para manejo de datos y representación gráfica

- La carta de Smith se desarrolló en 1939 para realizar, de forma gráfica, cálculos complejos en sistemas de Telecomunicación.

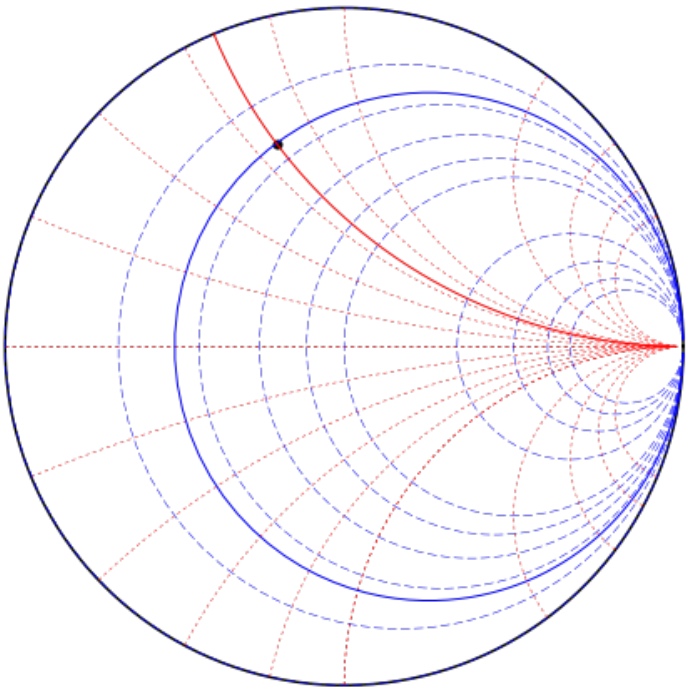
- Actualmente se sigue usando en la salida de aparatos de medida de ondas electromagnéticas



Impedancia de entrada en una línea de transmisión:

$$r_{in} + \jmath\, x_{in} = \frac{(r + \jmath\, x) + \jmath\, \tan(2\pi l_e)}{1 + (r + \jmath\, x)\, \jmath\, \tan(2\pi l_e)}$$

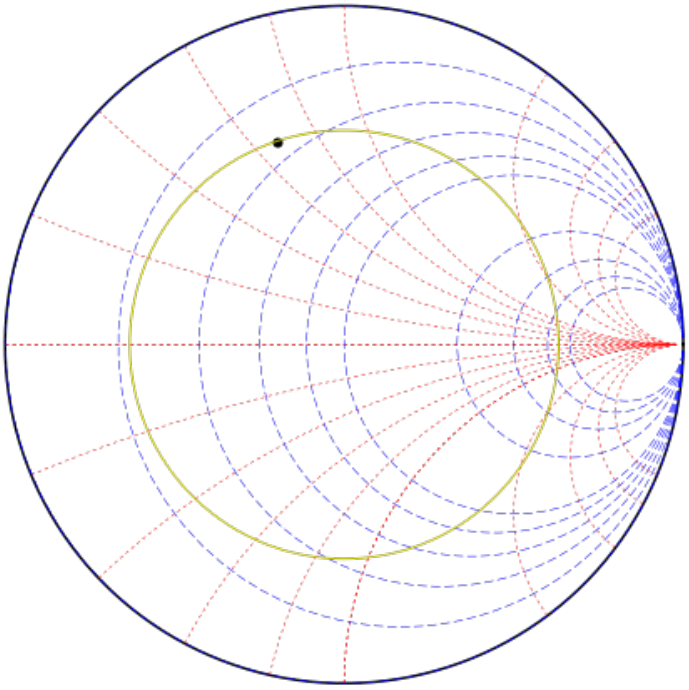
In [4]:
```
fig=plt.figure()
fig.set_size_inches(8.0,8.0)
inicia()
cirr(zln.real,c='b')
cirx(zln.imag,c='r')
punto(ga.real,ga.imag,c='k',s=100)
plt.show()
```



In [ ]:

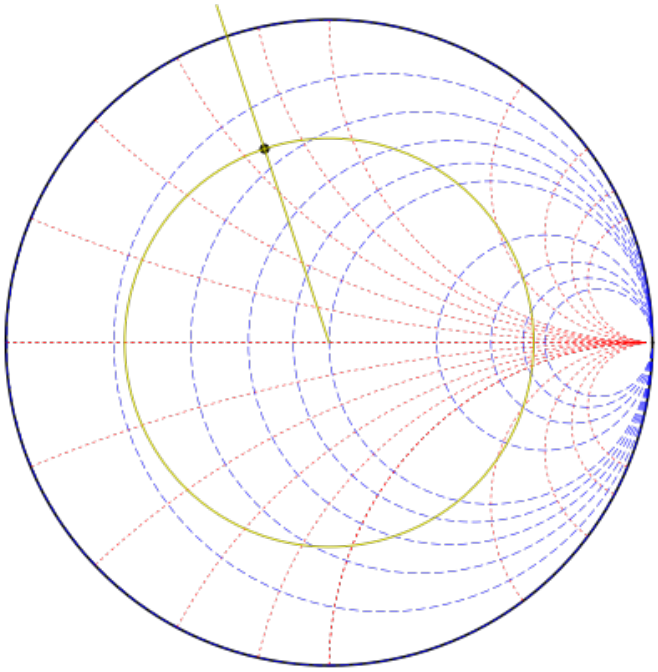
```
def inicia():
    thetas=np.linspace(0,2*np.pi,500)
    xscir=1.0*np.cos(thetas)
    yscir=1.0*np.sin(thetas)
    ax=plt.subplot(111)
    ax.axis('equal')
    ax.axis('off')
    ax.plot(xscir,yscir,'k')  # Circulo unidad
    xlis1=np.arange(-1,1,0.2) # x [-1,-0.8,-0.6,...,0.8]
    xlis2=np.arange(-5,6,dtype=int) # x [-5,-4,...,4,5]
    rlis1=np.arange(0,1,0.2)  # r [0,0.2,...0.8]
    rlis2=np.arange(0,6,dtype=int)  # r [0,1,2,...,5]
    # Dibuja los circulos anteriores
    for i in xlis1:
        ax.plot(*lineax(i),'r',lw=0.5,ls=':')
    for i in xlis2:
        ax.plot(*lineax(i),'r',lw=0.5,ls=':')
    for i in rlis1:
        ax.plot(*linear(i),'b',lw=0.5,ls='--')
    for i in rlis2:
        ax.plot(*linear(i),'b',lw=0.5,ls='--')
```

In [5]:
```
fig=plt.figure()
fig.set_size_inches(8.0,8.0)
inicia()
punto(ga.real,ga.imag,c='k',s=100)
circulo(np.abs(ga),'y')
plt.show()
```



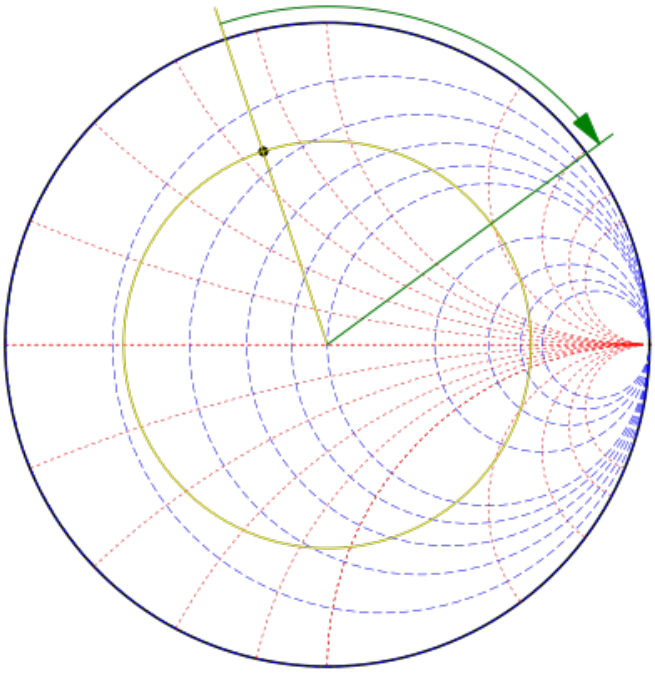
In [ ]:
```
def circulo(r,c='k',ls='-',lw=1):
    """
    (r,c='k',ls='-',lw=1)
    Circulo de radio r.
    Color, trazado('--',':','-.') y grueso.
    """
    ax=plt.subplot(111)
    thes=np.linspace(0,2.*np.pi,300)
    xsar=r*np.cos(thes)
    ysar=r*np.sin(thes)
    ax.plot(xsar,ysar,c=c,ls=ls,lw=lw)
```

In [6]:
```
fig=plt.figure()
fig.set_size_inches(8.0,8.0)
inicia()
punto(ga.real,ga.imag,c='k',s=100)
circulo(np.abs(ga),'y')
linea(0,0,*P2c(1.1,np.angle(ga)),'y')
plt.show()
```



In [ ]:
```
def linea(px1,py1,px2,py2,c='k',ls='-',lw=1):
    """
    (px1,py1,px2,py2,c='k',ls='-',lw=1)
    Linea desde (px1,py1) hasta (px2,py2).
    Color, trazado('--',':','-.') y grueso.
    """
    ax=plt.subplot(111)
    ax.plot([px1,px2],[py1,py2],c=c,ls=ls,lw=lw)
```

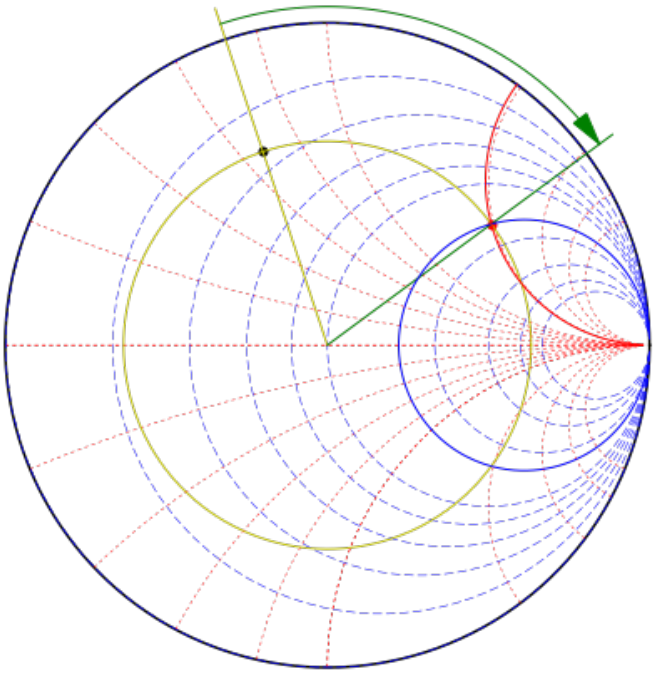
In [7]:
```
fig=plt.figure()
fig.set_size_inches(8.0,8.0)
inicia()
punto(ga.real,ga.imag,c='k',s=100)
circulo(np.abs(ga),'y')
linea(0,0,*P2c(1.1,np.angle(ga)),'y')
arco(1.05,np.angle(ga),np.angle(ga)-2*2*pi*lln1v,'g')
linea(0,0,*P2c(1.1,np.angle(ga)-2*2*pi*lln1v),'g')
plt.show()
```



In [ ]:

```
def arco(r,thi,thf,c='k',ls='-',lw=1):
    """
    (r,thi,thf,c='k',ls='-',lw=1)
    Dibuja un arco de radio r desde thi rad hasta thf rad.
    Color, trazado('--',':','-.') y grueso.
    """
    ax=plt.subplot(111)
    thes=np.linspace(thi,thf,100) # Lista de angulos
    xsar=r*np.cos(thes)  # Proyeccion al eje x
    ysar=r*np.sin(thes)  # Proyeccion al eje y
    # Punto inicial de la flecha: desde el valor -3, dx, dy
    flecha=(xsar[-3],ysar[-3],xsar[-1]-xsar[-3],ysar[-1]-ysar[-3]
)
    ax.plot(xsar,ysar,c=c,ls=ls,lw=lw)
    # Se dibuja la flecha con los valores normales
    ax.arrow(*flecha,head_width=0.05, head_length=0.1,fc=c,ec=c,\
            length_includes_head=True)
```

In [8]:
```
fig=plt.figure()
fig.set_size_inches(8.0,8.0)
inicia()
punto(ga.real,ga.imag,c='k',s=100)
circulo(np.abs(ga),'y')
linea(0,0,*P2c(1.1,np.angle(ga)),'y')
arco(1.05,np.angle(ga),np.angle(ga)-2*2*pi*lln1v,'g')
linea(0,0,*P2c(1.1,np.angle(ga)-2*2*pi*lln1v),'g')
cirr(zinn.real,c='b')
cirx(zinn.imag,c='r')
punto(*P2c(abs(ga),np.angle(ga)-2*2*pi*lln1v),c='r',s=100)
plt.show()
```



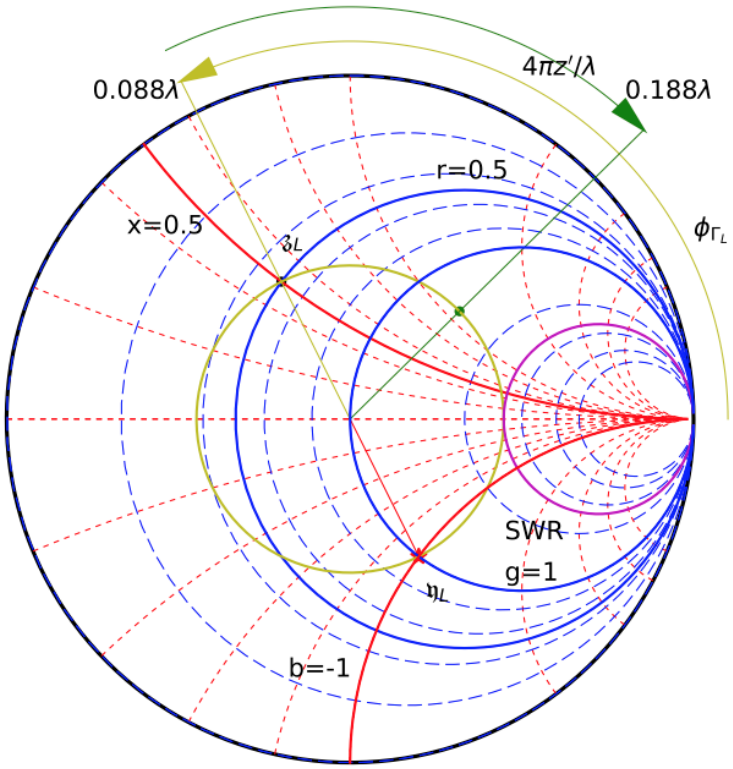
In [ ]:
```
def cirx(x,c='k',ls='-',lw=1):
    """
    (x,c='k',ls='-',lw=1)
    Dibuja el circulo de reactancia x en la carta de Smith.
    Color, trazado('--',':','-.') y grueso.
    """
    ax=plt.subplot(111)
    ax.plot(*lineax(x),c=c,ls=ls,lw=lw)

def cirr(r,c='k',ls='-',lw=1):
    """
    (r,c='k',ls='-',lw=1)
    Dibuja el circulo de resistencia r en la carta de Smith.
    Color, trazado('--',':','-.') y grueso.
    """
    ax=plt.subplot(111)
    ax.plot(*linear(r),c=c,ls=ls,lw=lw)
```

Nicholas Negroponte, fundador y director del MIT Media Lab:

Si los niños pudieran escribir programas informáticos, podrían aprender sobre el aprendizaje, entenderían lo que significa hacerse con una idea, convertirla en un algoritmo, ejecutar un código, observar el comportamiento y depurar el código.

Es un aprendizaje que es muy distinto a tener un montón de información en mi cabeza que quiero poner en la tuya y ahora pasaremos tiempo en una clase para transmitir esa información a tu cabeza, después te examinaré para ver si está en tu cabeza, y si está en tu cabeza pasaremos a otro tema.

Esto es una manera muy limitada de ver el aprendizaje, comparándola con el objetivo de que los niños pudieran aprender a aprender.



Funciones definidas

- Importación de paquetes y constantes

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

# Constantes
pi=np.pi # Numero pi
Np2db=20.*np.log10(np.e) # Factor neper a decibelios
Db2np=1./Np2db # Factor decibelios a Neper
D2r=pi/180. # Factor grados a radianes
R2d=180./pi # Factor radianes a grados
```

Funciones definidas

- Importación de paquetes y constantes
- Salida por pantalla

In [ ]:
```
# Funciones de salida por pantalla
def Pr(lis):
    """
    Imprime la lista de reales: Pr((pi,))
    Usa diferentes lineas. Formato 0:9.2E
    """
    for i in lis:
        print('{0:9.2E}'.format(i))

def Prc(co,va):
    """
    Imprime la lista de cabeceras y de reales: Pr(('Pi',),(pi,))
    Usa diferentes lineas. Formato 0:9.2E
    """
    for i, ca in enumerate(co):
        print(ca+'='+'{0:9.2E}'.format(va[i]))

def Prv(co,va):
    """
    Imprime una cabecera y la lista de reales: Pr(('Val',),(1.,2.
,3.))
    Usa una sola linea. Formato 0:9.2E
    """
    print(co+'=',end='')
    for i in va:
        print('{0:9.2E}'.format(i),end='')
    print()
```

Funciones definidas

- Importación de paquetes y constantes
- Salida por pantalla
- Complejos

In [ ]:
```
def C2p(x):
    """
    Complejo a tupla de polares.
    """
    return (np.abs(x),np.angle(x))

def C2pD(x):
    """
    Complejo a tupla de polares con fase en grados.
    """
    return (np.abs(x),np.angle(x)*R2d)

def P2c(r,th):
    """
    Modulo y fase a tupla de cartesianas.
    """
    return (r*np.cos(th),r*np.sin(th))

def P2cD(r,th):
    """
    Modulo y fase en grados a tupla de cartesianas.
    """
    thr=th*D2r
    return P2c(r,thr)
```

Funciones definidas

- Importación de paquetes y constantes
- Salida por pantalla
- Complejos
- Funciones específicas de las líneas de transmisión

In [ ]:
```
# Funciones de las lineas
def Gr(zl,z0):
    """
    Coeficiente de reflexion en la carga.
    Impedancia de carga e impedancia intrinseca. Complejas.
    """
    return (zl-z0)/(zl+z0)

def Zw(zp,zl,z0,coe):
    """
    Impedancia de onda. Caso con perdidas.
    (zp,zl,z0,const. de propagacion, complejo).
    """
    zw=z0*(zl+z0*np.tanh(coe*zp))/(z0+zl*np.tanh(coe*zp))
    return zw

def ZwSP(zp,zl,z0,be):
    """
    Impedancia de onda. Caso sin perdidas.
    (zp,zl,z0,const. fase).
    """
    zw=z0*(zl+z0*1.j*np.tan(be*zp))/(z0+zl*1.j*np.tan(be*zp))
    return zw
```

Funciones definidas

- Importación de paquetes y constantes
- Salida por pantalla
- Complejos
- Funciones específicas de las líneas de transmisión
- Funciones de dibujo para la carta de Smith

In [ ]:

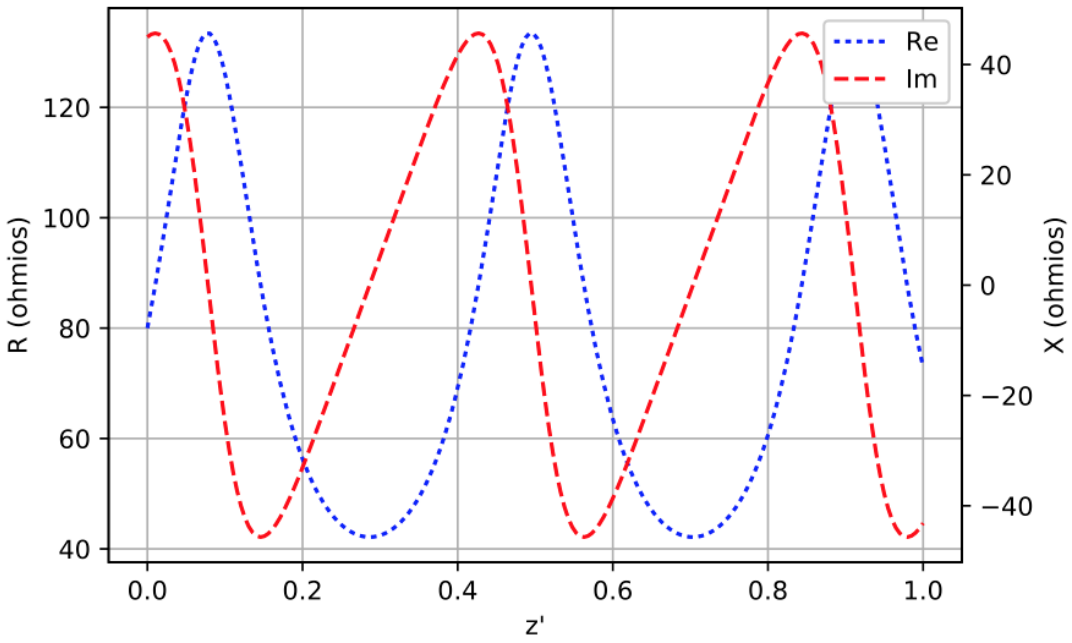
```
## Funciones generales de dibujo
def punto(x,y,c='k',marker='.',s=20):
    """
    (x,y,c='k',marker='.'
    Dibuja un punto en las coordenadas cartesianas x, y.
    Color y marker(x,+,*,s,v).
    """
    ax=plt.subplot(111)  # Todo el paquete dibuja en ax
    ax.scatter(x,y,c=c,marker=marker,s=s)

def arco(r,thi,thf,c='k',ls='-',lw=1):
    """
    (r,thi,thf,c='k',ls='-',lw=1)
    Dibuja un arco de radio r desde thi rad hasta thf rad.
    Color, trazado('--',':','-.') y grueso.
    """
    ax=plt.subplot(111)
    thes=np.linspace(thi,thf,100) # Lista de angulos
    xsar=r*np.cos(thes)  # Proyeccion al eje x
    ysar=r*np.sin(thes)  # Proyeccion al eje y
    # Punto inicial de la flecha: desde el valor -3, dx, dy
    flecha=(xsar[-3],ysar[-3],xsar[-1]-xsar[-3],ysar[-1]-ysar[-3]
)
    ax.plot(xsar,ysar,c=c,ls=ls,lw=lw)
    # Se dibuja la flecha con los valores normales
    ax.arrow(*flecha,head_width=0.05, head_length=0.1,fc=c,ec=c,\
            length_includes_head=True)

def circulo(r,c='k',ls='-',lw=1):
    """
    (r,c='k',ls='-',lw=1)
    Circulo de radio r.
    Color, trazado('--',':','-.') y grueso.
    """
    ax=plt.subplot(111)
    thes=np.linspace(0,2.*np.pi,300)
    xsar=r*np.cos(thes)
    ysar=r*np.sin(thes)
    ax.plot(xsar,ysar,c=c,ls=ls,lw=lw)

def linea(px1,py1,px2,py2,c='k',ls='-',lw=1):
    """
    (px1,py1,px2,py2,c='k',ls='-',lw=1)
    Linea desde (px1,py1) hasta (px2,py2).
    Color, trazado('--',':','-.') y grueso.
    """
    ax=plt.subplot(111)
```

Funciones definidas

- Importación de paquetes y constantes
- Salida por pantalla
- Complejos
- Funciones específicas de las líneas de transmisión
- Funciones de dibujo para la carta de Smith
- Funciones para cálculo numérico

In [ ]:

```
# Funcion de calculo numerico de ceros
# Se importa el metodo 'brentq'
from scipy.optimize import brentq
def Zwfc(zln,a,b,c,reim):
    """
    (zln,a,b,c,reim)
    Punto donde Z_w para zln vale c.
    [a,b] intervalo de busqueda
    reim='re', 'im'. Sin perdidas
    """
    if reim=='re':
        f=lambda x:c-(ZwSP(x,zln,1,2*pi)).real
    elif reim=='im':
        f=lambda x:c-(ZwSP(x,zln,1,2*pi)).imag
    else:
        print('Defina re o im')
        return
    z1=brentq(f,a,b)
    return z1
def Ywfc(zln,a,b,c,reim):
    """
    (zln,a,b,c,reim)
    Punto donde Y_w para zln vale c.
    [a,b] intervalo de busqueda
    reim='re', 'im'. Sin perdidas
    """
    if reim=='re':
        f=lambda x:c-(1/ZwSP(x,zln,1,2*pi)).real
    elif reim=='im':
        f=lambda x:c-(1/ZwSP(x,zln,1,2*pi)).imag
    else:
        print('Defina re o im')
        return
    z1=brentq(f,a,b)
    return z1
```



¿Por qué Python?

- Mathematica: Perfecto pero no es software libre
- Maxima con WxMaxima: Muy potente pero le falta notebooks y es poco atractivo
- Python con numpy y matplotlib

- Es el lenguaje de programación más popular en los cursos introductorios en las universidades de Estados Unidos.
- Es el cuarto lenguaje más usado de acuerdo con una encuesta realizada por el IEEE (Institute of Electrical and Electronics Engineering): Java, C, C++ y Python.
- Es fácil de usar, potente y versátil, siendo la mejor opción para la iniciación en la programación.
- Se usa en sistemas con un gran requerimiento informático como Dropbox, Google, Spotify o Netflix. Los ingenieros de Google marcaron su estrategia de desarrollo de software con la frase "Python where we can, C++ where we must".
- No es necesario "reinventar la rueda". No hay prácticamente ningún campo donde no haya desarrollada una librería en Python.
- Es software libre y de código abierto.
- Numpy, scipy, matplotlib y Jupyter-notebooks.
- Independiente de la plataforma (Linux, Windows y Mac).

Muchas gracias por vuestra atención

El código y un notebook explicativo está disponible para el que lo solicite.