HW2

Wylie Standage-Beier

February 4, 2015

1 Home Work 2

Problems: 6.1, 6.5, 6.9, 6.22, 6.25.

Due 2/4/15

Name: Wylie Standage-Beier

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize as optimize
import sympy
%matplotlib inline
Co = 3.0e8 # Speed of light
```

In [2]:

```
def bode_plot(x, y, title='', xlabel='', ylabel1='',ylabel2=''):
    '''Function to wrap plotting of complex numbers vs frequncy

    Parameters
    ----------
    x: ndarray
        x axis data array
    y: ndarray dtype=complex
        y axis data array

    return
    ------
    plot object

    Example
    -------
    >>> x = np.linspace(1e2,1e6,1e3)
    >>> y = 1+1j*1e-5*x+1/(1j*1e-5*x)
    >>> plot = bode_plot(x,y)
    '''
    plt.grid(True)
    plt.subplot(2, 1, 1)
    plt.plot(w, np.abs(Z),'red')
    plt.yscale('log')
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel1)
    plt.subplot(2, 1, 2)
    plt.plot(w, np.angle(Z,deg=True), 'blue' )
    plt.xlabel(xlabel)
    plt.ylabel(ylabel2)
    plt.show()
```

In [3]:

```python
def rollback(Zo,ZL,l,beta):
    ''' rollback of a load

    Parameter
    ---------
    Zo: float, complex, ndarray dtype= float or complex
        Is the instrinsic impedence of the transmission line
    ZL: float, complex, ndarray dtype= float or complex
        The load impedence of the tranmission line
    l: float
        The lenght of the transmission line in meters
    beta: float, complex, ndarray dtype= float or complex
        propigation constant of the transmission line

    retrun
    ------
    Input impedence of the ciruit

    Example
    -------
    >>> Zo, ZL = 50., 0 # Ohms
    >>> l, gamma = 0.25, 1.
    >>> rollback( Zo, ZL, l, gamma)

    '''
    #if type(l) is np.ndarray:
    #    l_len = l.size
    #    l.reshape((1,l_len))
    #elif type(beta) is np.ndarray:
    #    beta_len = beta.size
    #    beta.reshape(beta_len,1)
    prop_const = l*1j*beta #np.dot(l,1j*beta)
    #sin_prop_const = np.sin(prop_const)
    #cos_prop_const = np.cos(prop_const)
    #return Zo*((ZL*cos_prop_const+1j*Zo*sin_prop_const)/
    #           (Zo*cos_prop_const+1j*ZL*sin_prop_const))
    return Zo*(ZL + Zo*np.tanh(prop_const))/(Zo + ZL*np.tanh(prop_const))
```

In [4]:

```python
def get_Q_S(f,S,maxmin):
    ''' Simple function to get the Q of a resonate circuit

    Parameter
    ---------
    f: numpy.ndarray
        array containing the frequencies for Zin calculated
    S: numpy.ndarray
        Reflection of the circuit
    maxmin: bool
        True for resonates at max,
        False for resonates at min

    return
    ------
    Q-factor, resonate frequency'''
    S_mag = abs(S)
    S_max = max(S_mag)
    S_min = min(S_mag)
    index_max = np.argmax(S_mag)
    index_min = np.argmin(S_mag)
    if maxmin == True:
        index_lower = index_max
        index_upper = index_max
        bound = np.sqrt(1./2)*S_max
        while S_mag[index_lower] > bound:
            index_lower += -1
            assert index_lower >= 0, "Lower index out of bounds"
        while S_mag[index_upper] > bound:
            index_upper += 1
```

```
            assert index_upper < S_mag.size, "Upper index out of bounds"
        fo = f[index_max]
    else:
        index_lower = index_min
        index_upper = index_min
        bound = np.sqrt(0.5)*S_max
        while S_mag[index_lower] < bound:
            index_lower += -1
            assert index_lower >= 0, "Lower index out of bounds %d" % index_lower
        while S_mag[index_upper] < bound:
            index_upper += 1
            assert index_upper < S_mag.size, "Upper index out of bounds %d" % index_up
        fo = f[index_min]
    bandwidth = f[index_upper] - f[index_lower]
    Q = fo/bandwidth
    return Q, fo

def get_Q_Z(f,Z,SP):
    ''' Simple function to get the Q of a resonate circuit

    f: numpy.ndarray
        array containing the frequencies for Zin calculated
    Z: numpy.ndarray
        Impedence of the circuit
    SP: bool
        True for series resonates,
        False for parallel resonates at min

    return
    ------
    Q-factor, resonate frequency'''
    Z_mag = abs(Z)
    Z_max = max(Z_mag)
    Z_min = min(Z_mag)
    index_max = np.argmax(Z_mag)
    index_min = np.argmin(Z_mag)
    if SP == True:
        index_lower = index_min
        index_upper = index_min
        bound = Z_min*np.sqrt(2)
        while Z_mag[index_lower] < bound:
            index_lower += -1
            assert index_lower >= 0, "Lower index out of bounds %d" % index_lower
        while Z_mag[index_upper] < bound:
            index_upper += 1
            assert index_upper < Z_mag.size, "Upper index out of bounds %d" % index_up
        fo = f[index_min]
    else:
        index_lower = index_max
        index_upper = index_max
        bound = Z_max/np.sqrt(2)
        while Z_mag[index_lower] > bound:
            index_lower += -1
            assert index_lower >= 0, "Lower index out of bounds %d" % index_lower
        while Z_mag[index_upper] > bound:
            index_upper += 1
            assert index_upper < Z_mag.size, "Upper index out of bounds %d" % index_up
        fo = f[index_max]
    bandwidth = f[index_upper] - f[index_lower]
    Q = fo/bandwidth
    return Q, fo
```

In [5]:

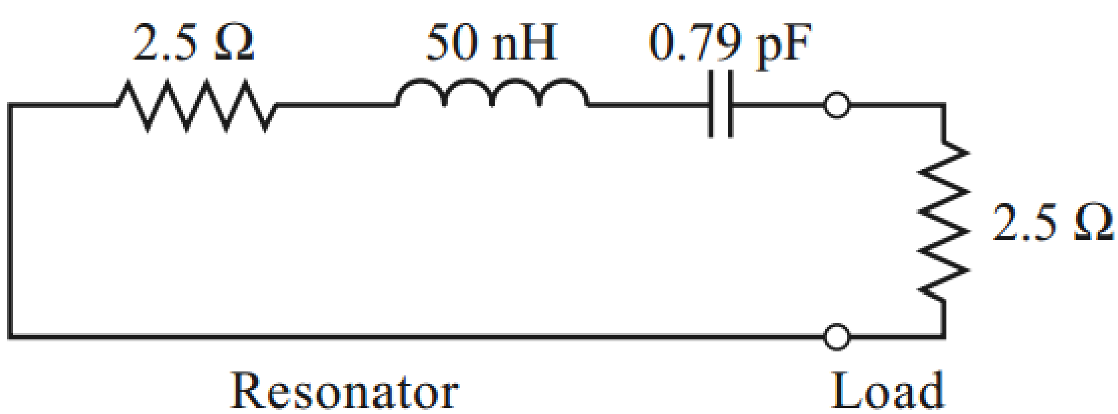
```
def real_sympy(exp):
    '''function to return just the real part of a sympy expression

    Parameter
    ---------
```

```
    exp: sympy
        sympy expression or symbol

    return
    ------
    real expression'''
    return (sympy.functions.conjugate(exp)+exp)/2
def imag_sympy(exp):
    '''function to return just the imaginary part of a sympy expression

    Parameter
    ---------
    exp: sympy
        sympy expression or symbol

    return
    ------
    imaginary expression'''
    return (exp - sympy.functions.conjugate(exp))/2
```

1.1 Problem 1

6.1

A series RLC resonator with an external load is shown below. Find the resonant frequency, the unloaded Q, and the loaded Q.



In [6]:
```
R1 = 2.5        # Ohms
L  = 50e-9      # Henerrys
C  = 0.79e-12   # Farads
RL = 2.5        # Ohms
```

In [7]:
```
wo = np.sqrt(1./(L*C))
print("Ideal lossless frequency of the LC circuit %f MHz" %
      float(wo/(2*np.pi)*1e-6))
```

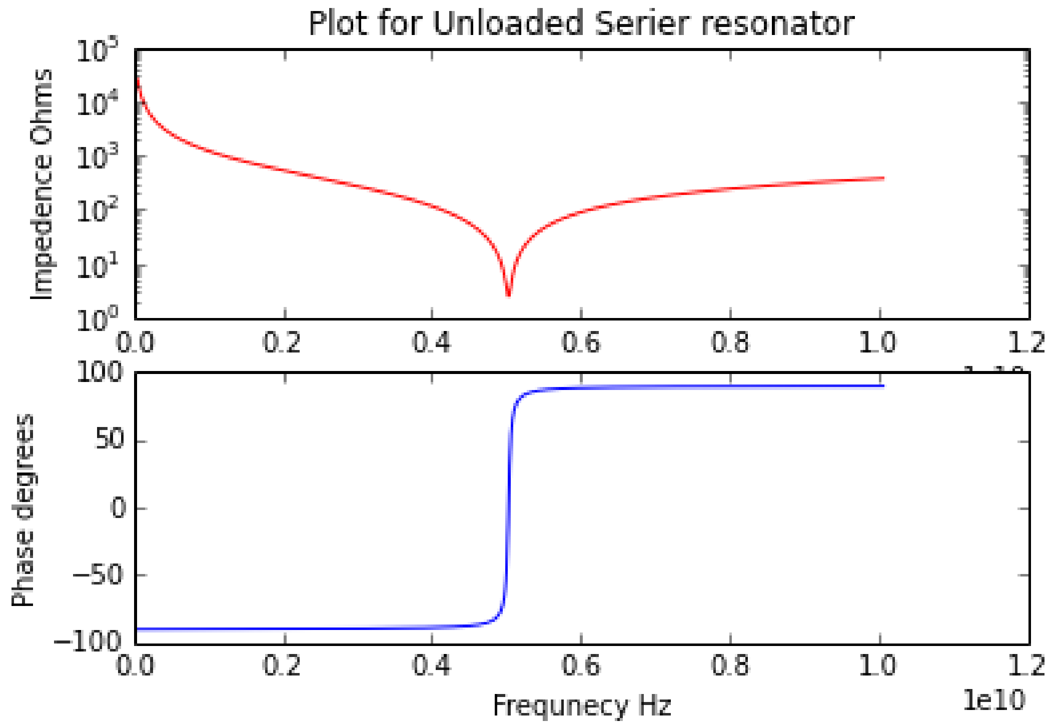
```
Ideal lossless frequency of the LC circuit 800.795426 MHz
```

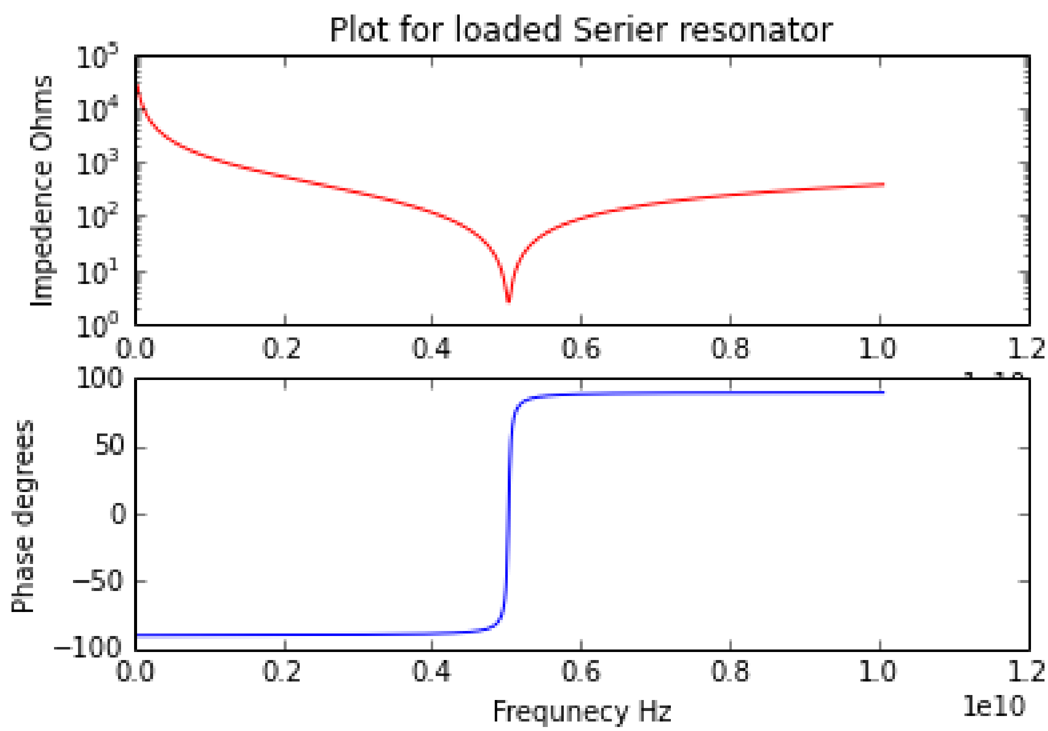
In [8]:
```
w = np.linspace(wo*0.01,wo*2,1001)
```

In [9]:
```
Z = R1 + 1j*w*L + 1/(1j*w*C)
ZL = R1 + 1j*w*L + 1/(1j*w*C) + RL
```

In [10]:

```
bode_plot(w,Z,title="Plot for Unloaded Serier resonator",
          xlabel="Frequnecy Hz",ylabel1='Impedence Ohms',
          ylabel2="Phase degrees")
bode_plot(w,ZL,title="Plot for loaded Serier resonator",
          xlabel="Frequnecy Hz",ylabel1='Impedence Ohms',
          ylabel2="Phase degrees")
```





In [11]:
```
Qo = wo*L/R1
print('Q-factor for unloaded %.3f' %float(Qo))
```
```
Q-factor for unloaded 100.631
```

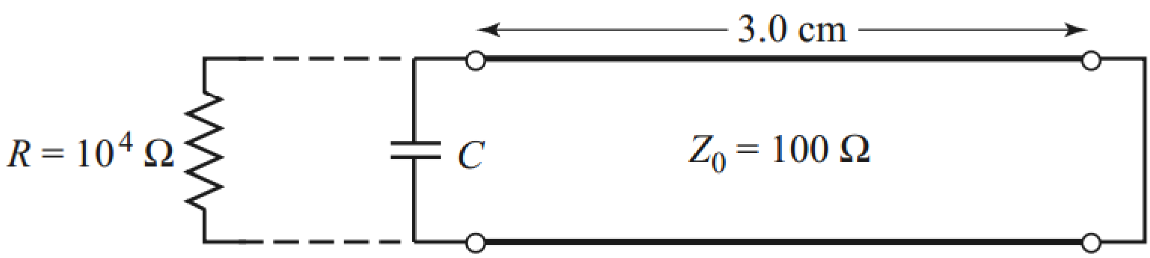
In [12]:
```
Qe = wo*L/RL
print('Q-factor for external %.3f' %float(Qe))
```
```
Q-factor for external 100.631
```

In [13]:
```
QL = (Qo**-1 + Qe**-1)**-1
print('Q-factor for loaded %.3f' %float(QL))
```
```
Q-factor for loaded 50.315
```

1.2 Problem 2

6.5

A resonator is constructed from a 3.0 cm length of 100 Ohms air-filled coaxial line, shorted at one end and terminated with a capacitor at the other end, as shown below.



(a) Determine the capacitor value to achieve the lowest order resonance at 6.0 GHz.

(b) Now assume that loss is introduced by placing a 10,000 Ohms resistor in parallel with the capacitor. Calculate the unloaded Q.

In [14]:
```
l  = 0.03 # 3.00 cm
Zo = 100. # Ohms
R  = 1e4  # Ohms
fo = 6e9  # Hz
epsilonr = 1.   # unitless
mur      = 1.   # unitless
```

2 a.

Determine the capacitor value to achieve the lowest order resonance at 6.0 GHz.

In [15]:
```
wo = 2*np.pi*fo
beta = ( wo * np.sqrt(epsilonr*mur)) / Co
```

In [16]:
```
Zin = rollback(Zo,0.,l,beta)
```

In [17]:
```
Capacitor = -1./(wo*Zin.conj().imag)
Lo = Zin.imag/wo
```

In [18]:
```
print('The equivalent Inductance of the transmission line and short %.3f nH' %
      (Lo*1e9))
print('Matching Capacitor %.3f pF' % (Capacitor*1e12))
```

```
The equivalent Inductance of the transmission line and short 1.927 nH
Matching Capacitor 0.365 pF
```

2 b.

Now assume that loss is introduced by placing a 10,000 Ohms resistor in parallel with the capacitor. Calculate the unloaded Q.

Simple approximation

This is a simple approximation modeling the coax roll back as a inductor. This is only valid for a single or a very narrow frequency range.

In [19]:

```python
Q_LRC = R / (wo*Lo)
```

In [20]:

```python
print('Q-factor for circuit using the LRC model %.3f' % float(Q_LRC))
```

```
Q-factor for circuit using the LRC model 137.638
```

More Accurate approximation

The above model is not 100% correct as the impedance of the Coax roll back is not function as a inductor with frequency, only at the signal or a VERY narrow frequency. This is a numeric approimation using the Scipy minimize tools.

In [21]:

```python
def resonator_helper(Zo,Zl,l,C,R=0.,epsilonr=1.,mur=1.):
    ''' Generates impedance of the transmission line circuit

    Parameter
    ---------
    Zo:
        line impedance
    Zl:
        terminated load impedance
    l:
        length of the line in meters
    epsilonr: float or complex
        epsilon of the media
    mur: float or complex
        mu of the media

    return
    ------
    function with only argument
    '''
    def resonator_impedance(frequency):
        ''' Function to get the single frequency impedance of the resonator

        Parameter
        ---------
        frequency: float
            the frequency for the impedance to be calculated

        return
        ------
        The roll impedance at that frequency
        '''
        w = 2*np.pi*frequency
        beta = ( w * np.sqrt(epsilonr*mur)) / Co
        Zc = 1./(1j*w*C)
```

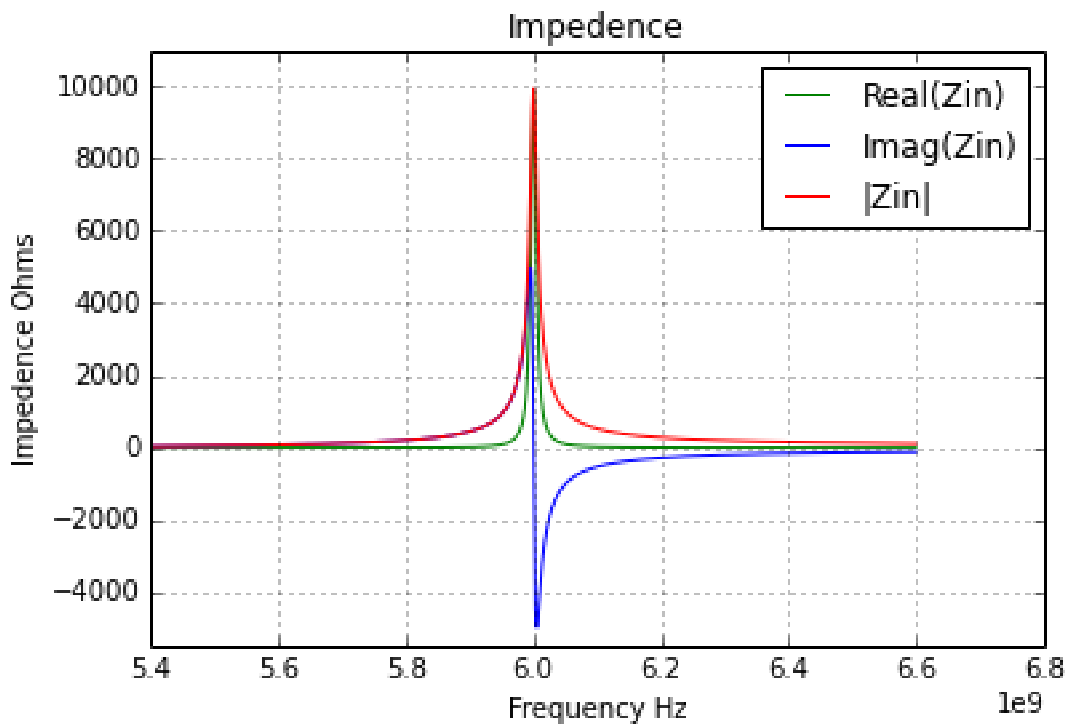
```
        Z_line = rollback( Zo, Zl, l, beta)
        if R == 0:
            Zin = ( Z_line**-1 + Zc**-1)**-1
        else:
            Zin = ( Z_line**-1 + Zc**-1 + R**-1)**-1
        return Zin
    def resonator_impedance_mag(frequency):
        ''' Function to get the impedance magitude

        Parameter
        ---------
        frequency: float
            frequency driving the resonator

        return
        ------
        magitude of the impedance
        '''
        return np.abs(resonator_impedance(frequency))
    def resonator_impedance_imag(frequency):
        ''' Function to get the imaginary impedance of the resonator

        Parameter
        ---------
        frequnecy: float
            frequnecy driving the the resonator

        return
        ------
        imaginary part of the impedance
        '''
        return np.imag(resonator_impedance(frequency))
    def resonator_reflection(frequency):
        ''' Function to get the reflection coeficent for the resonator

        Parameter
        ---------
        frequency: float
            frequency driving the resonator

        return
        ------
        relfection coeficent
        '''
        Zin = resonator_impedance(frequency)
        return (Zin-Zo)/(Zin+Zo)
    def resonator_reflection_mag(frequency):
        ''' Function to get the magitude of the resonator reflection

        Parameter
        ---------
        frequency: float
            frequency driving the resonator

        return
        ------
        reflection magitude
        '''
        return np.abs(resonator_reflection(frequency))
    return resonator_impedance, resonator_impedance_mag,\
            resonator_impedance_imag, resonator_reflection,\
            resonator_reflection_mag
```

In [22]:

```
def roller(Zo,ZL,l,w):
    Llambda = l*1j*w/ Co
    Zin = Zo*((ZL+Zo*np.tanh(Llambda))/\
```

```
                (Zo+ZL*np.tanh(Llambda)))
    return Zin
```

In [23]:
```
f = np.linspace(0.90*fo,1.1*fo,1e3)
w = 2*np.pi*f
beta =  w / Co
lamdba = l*1j*beta
ZL = 0

Zc = 1/(1j*w*Capacitor)
zstub = roller( 100., 0, l, w)
Zin = (zstub**-1+Zc**-1+R**-1)**-1
gamma1 = (Zin-50)/(Zin+50)
#gamma1
plt.title("Impedence")
plt.grid(True)
plt.xlabel("Frequency Hz")
plt.ylabel("Impedence Ohms")
plt.ylim((1.1*min(np.imag(Zin)),1.1*R))
plt.plot(f,np.real(Zin),'g',label="Real(Zin)")
plt.plot(f,np.imag(Zin),'b',label="Imag(Zin)")
plt.plot(f,np.abs(Zin),'r',label="|Zin|")
plt.legend()
plt.show()
```



In [24]:
```
#zfunc = Z_res(freq)
#gamma_func = (zfunc-50.)/(zfunc+50.)
#plt.title("Reflection Coeficent")
#plt.plot(freq,np.real(gamma(freq)),'b')
#plt.plot(freq,np.imag(gamma(freq)),'r')
#plt.plot(freq,np.abs(gamma(freq) ),'black')
```

In [25]:
```
Qcalc, focalc = get_Q_Z(f,Zin,False)
print( '''Calculated center frequency of resonator %.3f GHz,
Calculated Q-factor of the resonator %.3f''' %
       ( focalc/1e9, Qcalc))
```

```
Calculated center frequency of resonator 6.001 GHz,
Calculated Q-factor of the resonator 555.056
```

1.3 Problem 3

6.9

A rectangular cavity resonator is constructed from a 2.0 cm length of aluminum X-band waveguide. The cavity is air filled. Find the resonant frequency and unloaded Q of the TE101 and TE102 resonant modes.

In [26]:
```
a, b, d = 0.0285, 0.01262, 0.02 # demention in meters
modes = [(1,0,1), (1,0,2)]
epsilonr = 1.
mur      = 1.
```

Sigma of the cavity

No material was given for the cavity, therefore I am going with copper

In [27]:
```
sigma = 5.813e7
```

In [28]:
```
def get_k( mode, dim ):
    ''' function to return the k magitude of the cavity

    Parameter
    ---------
    mode: tuple
        (m, n, l) This is the mode numbers of the cavity
    dim: tuple
        (a, b, c) This is the physical dementions of the cavity

    return
    ------
    Magitude of the k vector'''
    m, n, l = mode
    a, b, c = dim
    return np.sqrt(((m*np.pi)/a)**2+((n*np.pi)/b)**2+((l*np.pi)/c)**2)

def fo_from_k(k,epsilonr=epsilonr,mur=mur):
    ''' converts k mag to frequency for the given cavity

    Parameter
    ---------
    k:
        k vector mag for the cavity
    epsilonr:
        is the relitive epsilon of the material in the cavity
    mur:
        is the relitive mu of the material in the cavity

    return
    ------
    frequnecy of the k vector mag'''
    return Co*k/(np.pi*np.sqrt(epsilonr*mur))

def Q_cavity(mode, dim, fo, sigma):
    '''Calculates the Q-factor of the cavity from the Rs of the cavity

    Parameter
    ---------
    mode: tuple
        (m, n, l) This is the mode numbers of the cavity
```

```
    dim: tuple
        (a, b, c) This is the physical dementions of the cavity
    fo:
        frequnecy of resoninates
    sigma:
        surface conductivity of the walls of the cavity

    return
    ------
    Q-factor of the cavity'''
    a, b, d = dim
    m, n, l = mode
    muo = np.pi*4e-7
    epsilono =  8.854187817e-12
    eta = np.sqrt(muo/epsilono)
    wo = 2*np.pi*fo
    Rs = np.sqrt((wo*muo)/(2*sigma))
    return (((k*a*d)**3*b*eta)/(2*np.pi**2*Rs))/\
            (2*l**2*a**3*b+2*b*d**3+l**2*a**3*d+a*d**3)
```

In [29]:

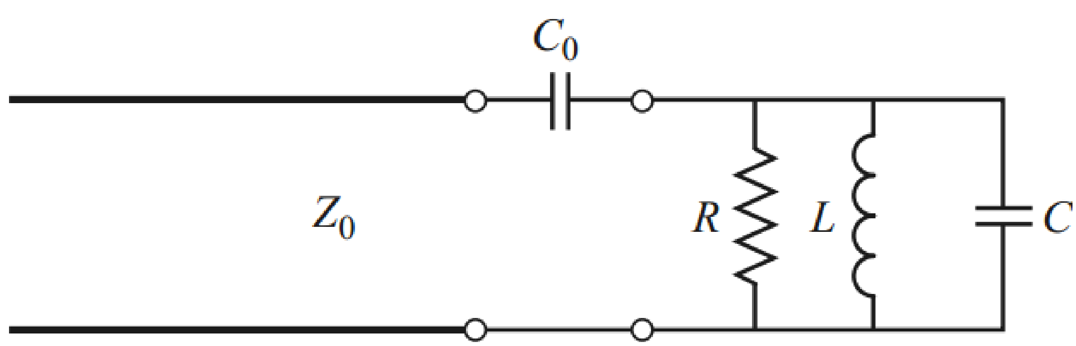
```
for m in modes:
    k = get_k(m, (a,b,d))
    fo = fo_from_k(k)
    Q = Q_cavity(m, (a,b,d), fo, sigma)
    print('Mode: %s, frequency %.6f GHz, Q-Factor %.3f' % (str(m), fo/1e9, Q))
```

```
Mode: (1, 0, 1), frequency 18.324937 GHz, Q-Factor 6048.697
Mode: (1, 0, 2), frequency 31.793133 GHz, Q-Factor 7669.686
```

1.4 Problem 4

6.22

A parallel RLC circuit, with R = 1000 Ohms , L = 1.26 nH, C = 0.804 pF, is coupled with a series capacitor, C0, to a 50 ohms transmission line, as shown below. Determine C0 for critical coupling to the line. What is the resonant frequency?



In [30]:

```
R = 1000.      # Ohms
L = 1.26e-9    # Henry's
C = 0.804e-12  # Farads
Zo = 50
```

Parallel Resonance circuit

1. $$Z_{||RLC} = (R^{-1} + (j * \omega * L)^{-1} + (\frac{1}{j * \omega * C})^{-1})^{-1}$$

2.

$$Z_C{}^* = Z_L$$

3.

$$\frac{1}{\omega_o * C} = j * \omega_o * L$$

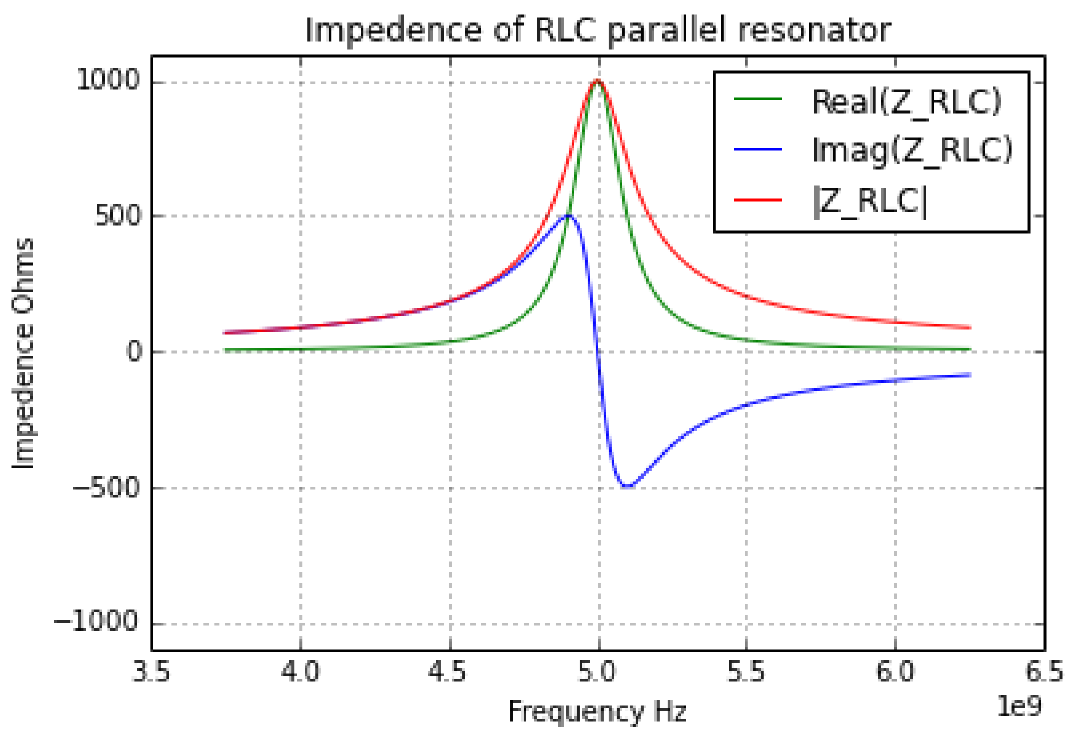
4.

$$\omega_o = \frac{1}{\sqrt{L * C}}$$

In [31]:
```
wo = np.sqrt(1/(L*C))
fo = wo / (2*np.pi)
print('The Unloaded Resonance frequency of the parallel resonance is %.6f GHz'
      % float(fo * 1e-9 ))
```

The Unloaded Resonance frequency of the parallel resonance is 5.000424 GHz

In [32]:
```
w = np.linspace(0.75*wo,1.25*wo,1000)
Z_RLCo = ((R)**-1 + (1j*wo*L)**-1 + (1./(1j*wo*C))**-1)**-1
Z_RLC = ((R)**-1 + (1j*w*L)**-1 + (1./(1j*w*C))**-1)**-1
f = w/(2*np.pi)
plt.title("Impedence of RLC parallel resonator")
plt.xlabel("Frequency Hz")
plt.ylabel("Impedence Ohms")
plt.grid(True)
plt.ylim((-1.1*R,1.1*R))
plt.plot(f,np.real(Z_RLC),'g',label="Real(Z_RLC)")
plt.plot(f,np.imag(Z_RLC),'b',label="Imag(Z_RLC)")
plt.plot(f,np.abs(Z_RLC),'r',label="|Z_RLC|")
plt.legend()
plt.show()
```



Q factor of resonance cicruit

1.

$$g = \frac{Q_0}{Q_e}$$

2.
$$1 = \frac{Q_0}{Q_e}$$

3.
$$Q_e = Q_0$$

4.
$$Q_0 = \frac{Z_0}{\omega_o L}$$

5.
$$Q_e = \frac{R_L}{\omega_o L}$$

Input Impedance

1.
$$Z_{||RLC} = (R^{-1} + (j * \omega * L)^{-1} + (\frac{1}{j * \omega * C})^{-1})^{-1}$$

2.
$$Z_{||RLC} = (R^{-1} + (j * \omega * L)^{-1} + j * \omega * C)^{-1}$$

Max Power transfor happens when the imaginary parts of the impedences are equal in magitube and opposite in sign.

1.
$$Z_{in} = Z_{C_0} + Z_{||RLC}$$

2.
$$Imag(Z_{C_o}) = Imag(Z_{||RLC}{}^*)$$

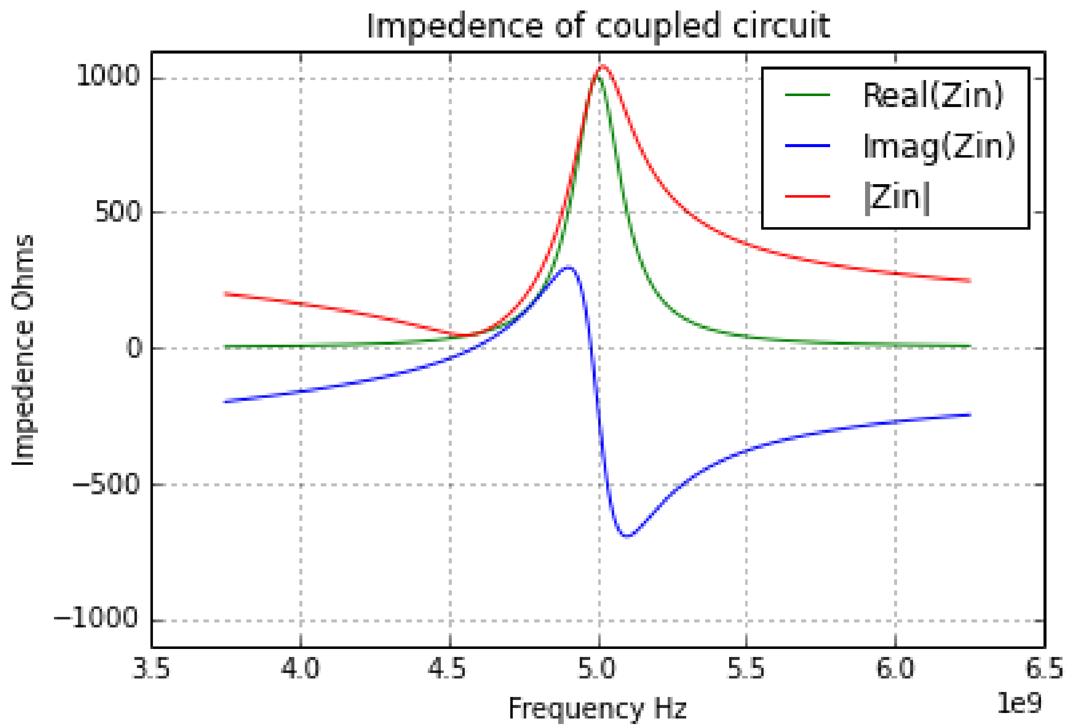
3.
$$\frac{-1}{\omega * C_o} = Imag(Z_{||RLC}{}^*)$$

4.
$$C_o = \frac{-1}{\omega * Imag(Z_{||RLC}{}^*)}$$

5.
$$C_o = \frac{1}{\omega * Imag(Z_{||RLC})}$$

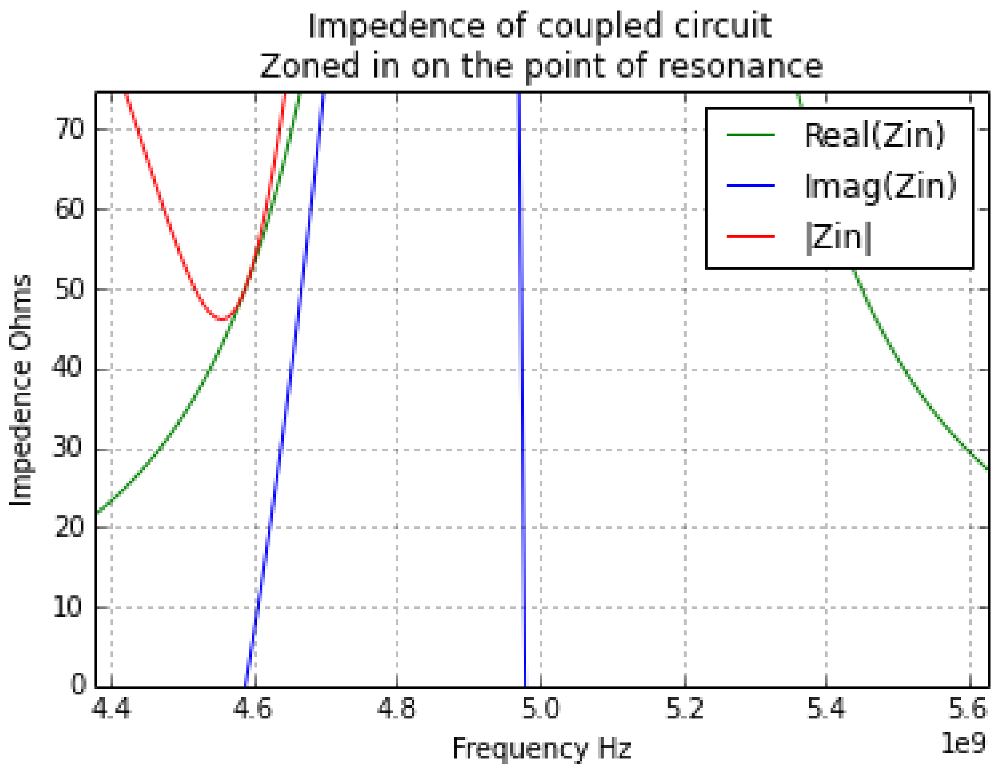
The above angular frequency is not that of the idea resonate frequency. It is that of the new or shifted resonate frequency.

In [38]:
```
C0 = 0.159101e-12      # Calculated coupling capactitor value
ZC0 = 1/(1j*w*C0)
Zin = Z_RLC+ZC0
plt.title("Impedence of coupled circuit")
plt.grid(True)
plt.xlabel("Frequency Hz")
plt.ylabel("Impedence Ohms")
plt.ylim((-1.1*R,1.1*R))
plt.plot(f,np.real(Zin),'g',label="Real(Zin)")
plt.plot(f,np.imag(Zin),'b',label="Imag(Zin)")
plt.plot(f,np.abs(Zin),'r',label="|Zin|")
plt.legend()
f[np.argmax(np.real(Zin))]
plt.show()
```



In [39]:

```
plt.title('''Impedence of coupled circuit
Zoned in on the point of resonance''')
plt.grid(True)
plt.xlabel("Frequency Hz")
plt.ylabel("Impedence Ohms")
plt.ylim((0,75))
plt.xlim((0.875*fo,1.125*fo))
plt.plot(f,np.real(Zin),'g',label="Real(Zin)")
plt.plot(f,np.imag(Zin),'b',label="Imag(Zin)")
plt.plot(f,np.abs(Zin),'r',label="|Zin|")
plt.legend()
plt.show()
```



Note

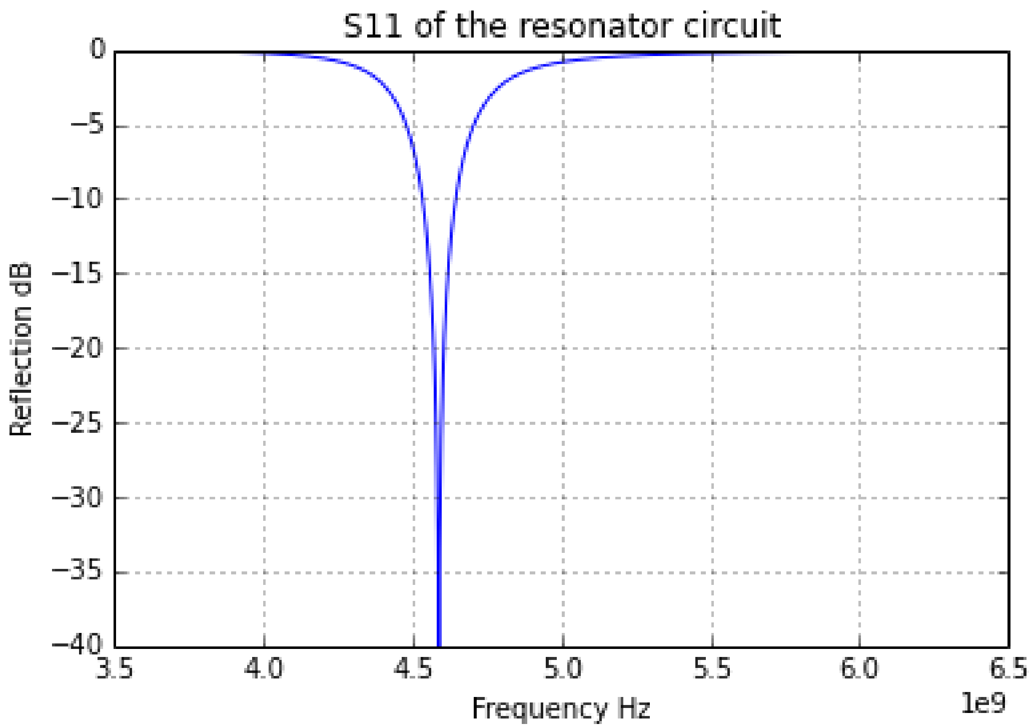
In the above plot the matching 50 Ohms at less then resonate frequncy. This is the point of maximum power transfor. See page 298 and 299 of pozar.

In [35]:

```
S = (Zin-Zo)/(Zin+Zo)
plt.title("S11 of the resonator circuit")
plt.xlabel("Frequency Hz")
plt.ylabel("Reflection dB")
plt.grid(True)
plt.ylim((-40,0))
plt.plot(f,20*np.log10(abs(S)))
```

Out [35]:

```
[<matplotlib.lines.Line2D at 0x11b8c940>]
```



In [36]:
```
# Calculated using the reflection values
Qcalc, focalc = get_Q_S(f,S,False)
print( '''Calculated center frequency of resonator %.3f GHz,
Calculated Q-factor of the resonator %.3f,
for the coupling capacitor value %.3f pF''' %
       ( focalc/1e9, Qcalc, C0*1e12))
```

```
Calculated center frequency of resonator 4.589 GHz,
Calculated Q-factor of the resonator 13.890,
for the coupling capacitor value 0.159 pF
```

In [37]:
```
# Calculated using the Impedence values
Qcalc, focalc = get_Q_Z(f,Zin,True)
print( '''Calculated center frequency of resonator %.3f GHz,
Calculated Q-factor of the resonator %.3f,
for the coupling capacitor value %.3f pF''' %
       ( focalc/1e9, Qcalc, C0*1e12))
```

```
Calculated center frequency of resonator 4.554 GHz,
Calculated Q-factor of the resonator 25.993,
for the coupling capacitor value 0.159 pF
```

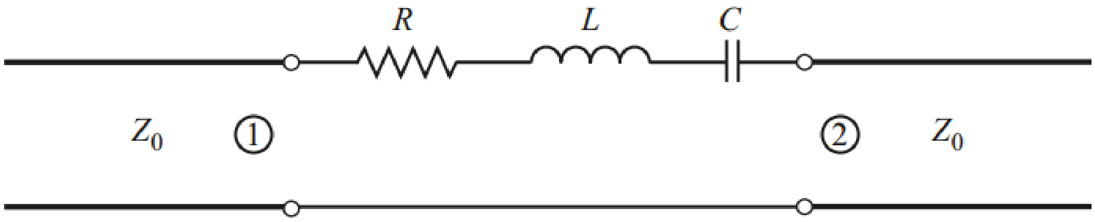
2 ansers, what is correct

Above there are 2 answers, one calculated from the port 1 scattering parameters and 2 from the input impedence of the circuit. The differences happen in the value from the impedence of the circuit does not count the loading of it being connected to a transmission line. The method using the scattering parameters does and this explains the lower value and the small shift in the frequency.

1.5 Problem 5

6.25

A microwave resonator is measured in a two-port configuration like that shown in Figure 6.21. The minimum insertion loss is measured as 1.94 dB at 3.0000 GHz. The insertion loss is 4.95 dB at 2.9925 GHz and at 3.0075 GHz. What is the unloaded Q of the resonator?



Note the about 3 dB difference is the values of the insertion loss at resonances of 3.0000 GHz and the off resonances at 2.9925 ad 3.0075 GHz. This correspones the the Bandwidth of the resonator.

1. $$BW = \frac{F_{max} - F_{min}}{F_{center}}$$

2. $$BW = \frac{3.0075GHz - 2.99925GHz}{3.0000GHz}$$

3. $$BW = \frac{15.0MHz}{3.0000GHz}$$

4. $$BW = \frac{1}{200}$$

eq. 6.21 on page 277

4. $$BW = \frac{1}{Q}$$

5. $$Q = \frac{1}{BW}$$

6. $$Q = \frac{3.0000GHz}{15.0MHz}$$

7. $$Q = 200.0$$