

Circuito RLC serie

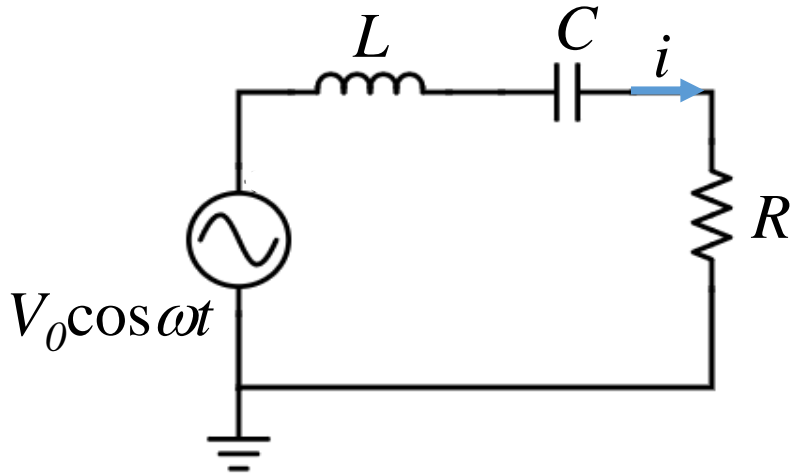
Corriente Alterna

Clase 12



LABORATORIO 3
2do cuatrimestre 2020

Circuito RLC – Corriente alterna



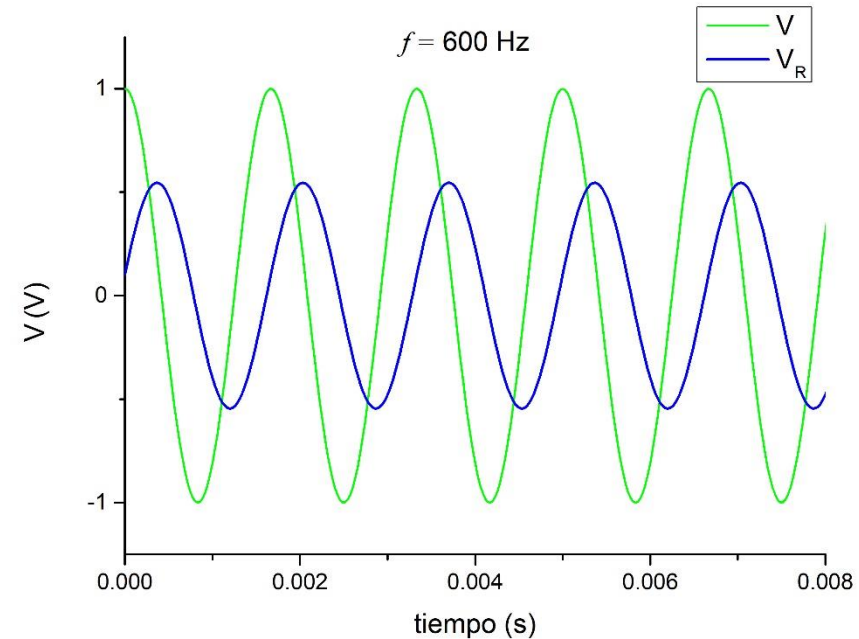
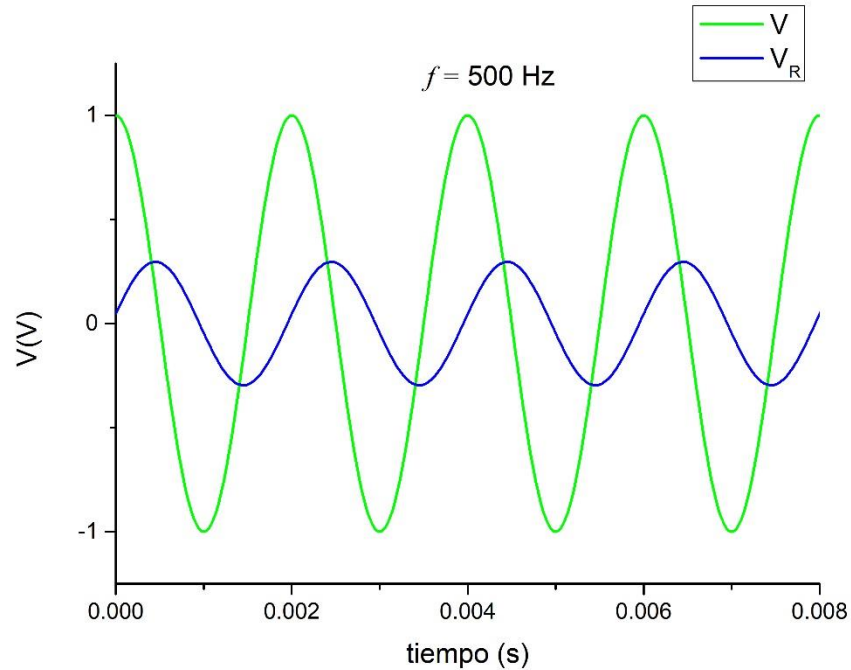
$$v = L \frac{di}{dt} + \frac{1}{C} \int i dt + iR$$

$$V = L(j\omega)I + \frac{1}{C} \left(-\frac{j}{\omega} \right) I + IR$$

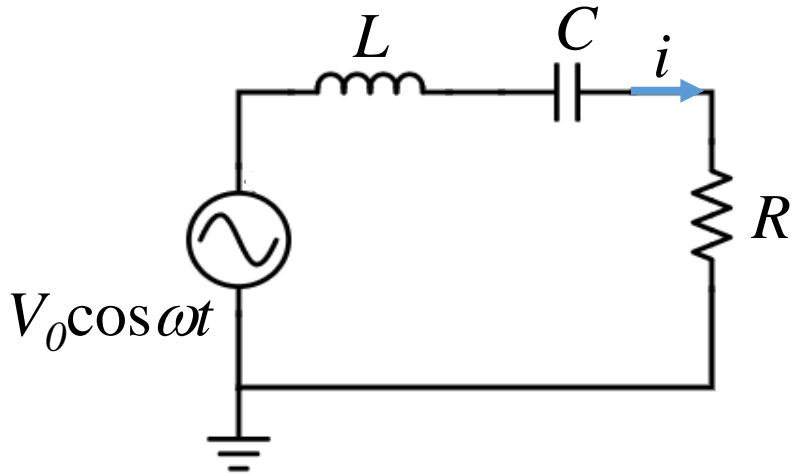
$$V_R = I * R$$

$$i = I_0 \cos(\omega t + \varphi)$$

$$I = \frac{V}{R + j \left(\omega L - \frac{1}{\omega C} \right)}$$



Circuito RLC – Corriente alterna



$$I = \frac{V}{R + j\left(\omega L - \frac{1}{\omega C}\right)}$$

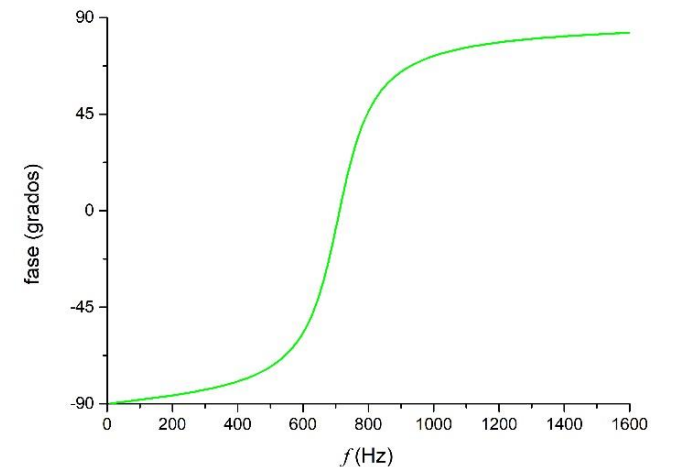
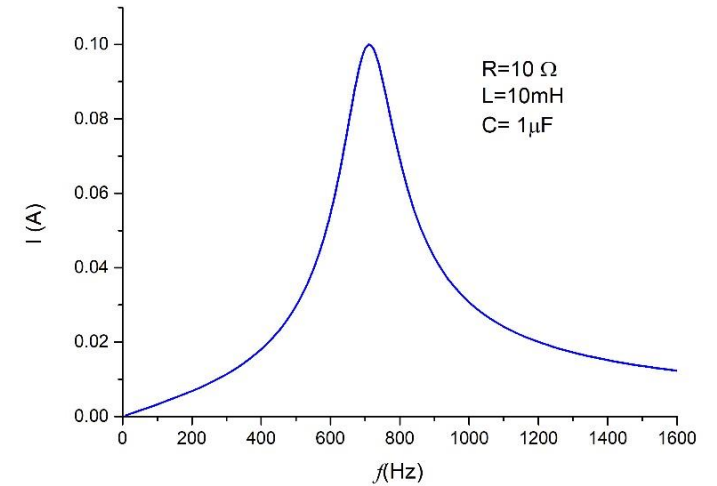
$$|I| = I_0 = \frac{V_0}{\sqrt{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2}}$$

$$\tan \varphi = \frac{\left(\omega L - \frac{1}{\omega C}\right)}{R}$$

Frecuencia de
resonancia

$$\omega_0 = \frac{1}{\sqrt{LC}}$$

$$f_0 = \frac{\omega_0}{2\pi}$$

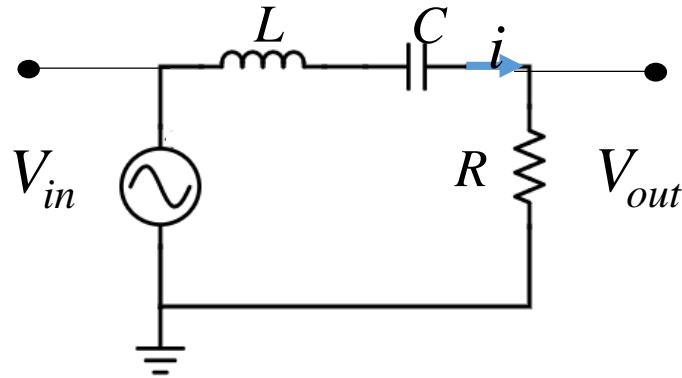
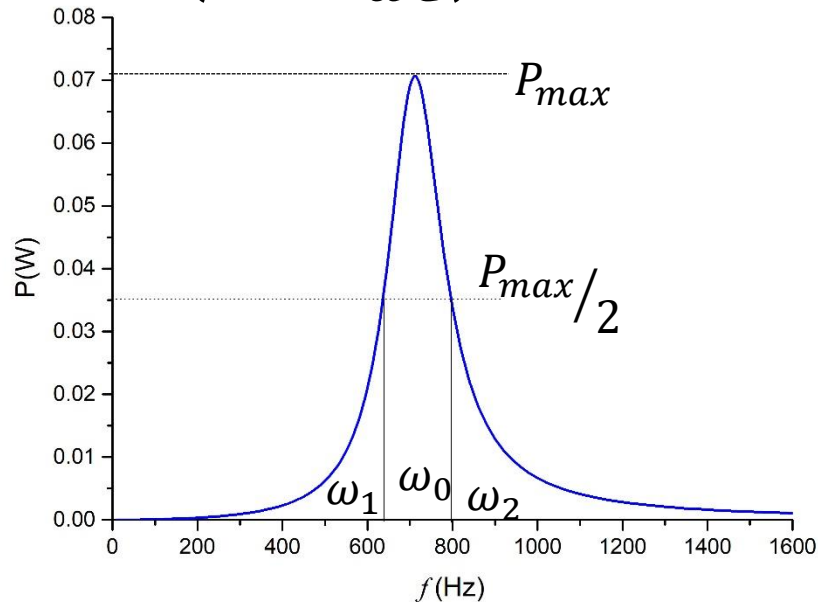


Respuesta en frecuencia

Potencia disipada

$$P = I_{ef}^2 * R = \frac{I^2}{2} * R$$

$$P = \frac{V_{ef}^2 * R}{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2}$$



Transferencia de tensión

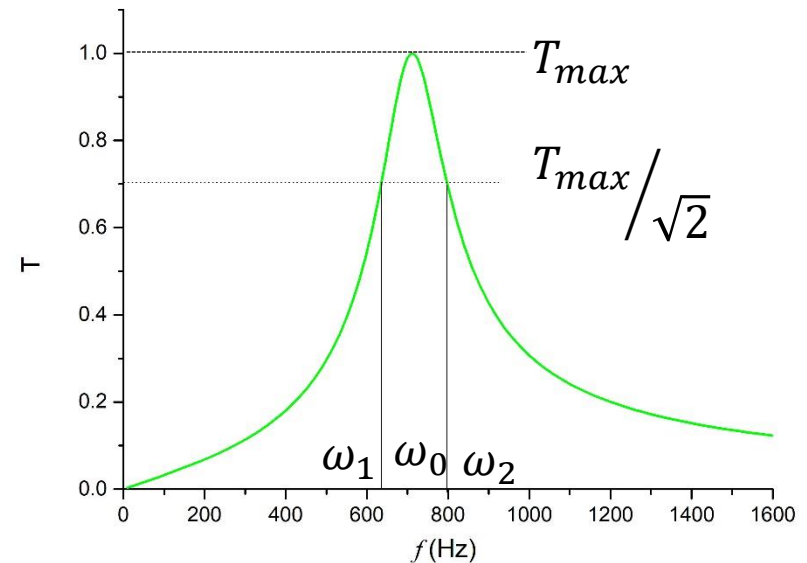
$$T = \left| \frac{V_{out}}{V_{in}} \right| = \frac{|I| * R}{V_0}$$

$$T = \frac{R}{\sqrt{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2}}$$

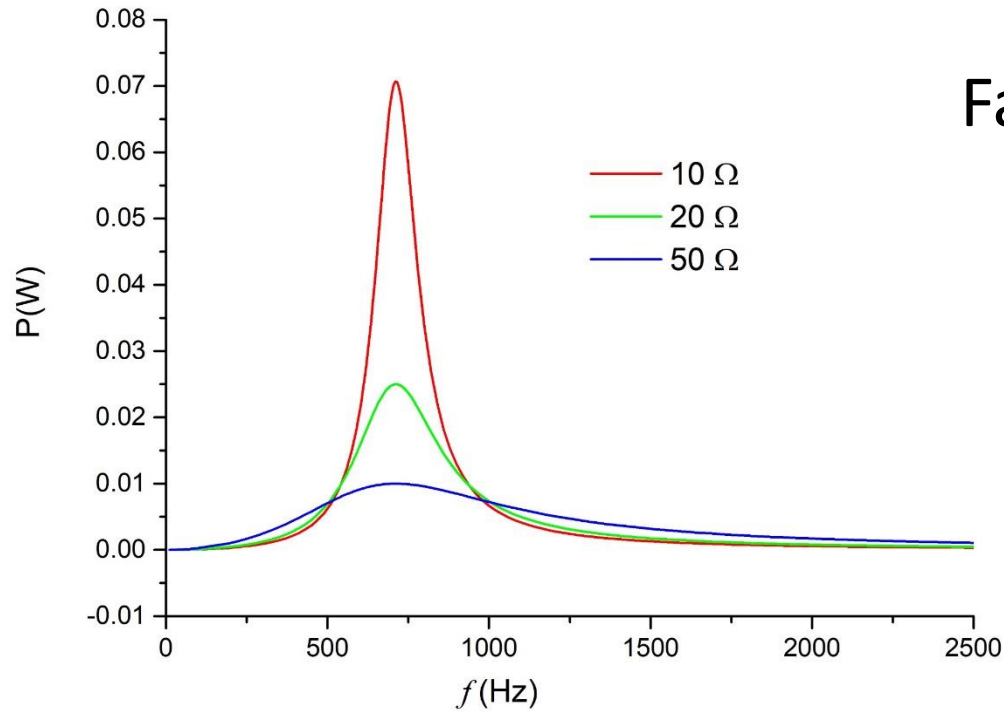
Ancho de la curva

$$\Delta\omega = \omega_2 - \omega_1$$

$$\Delta\omega = \frac{R}{L}$$



Influencia del valor de R



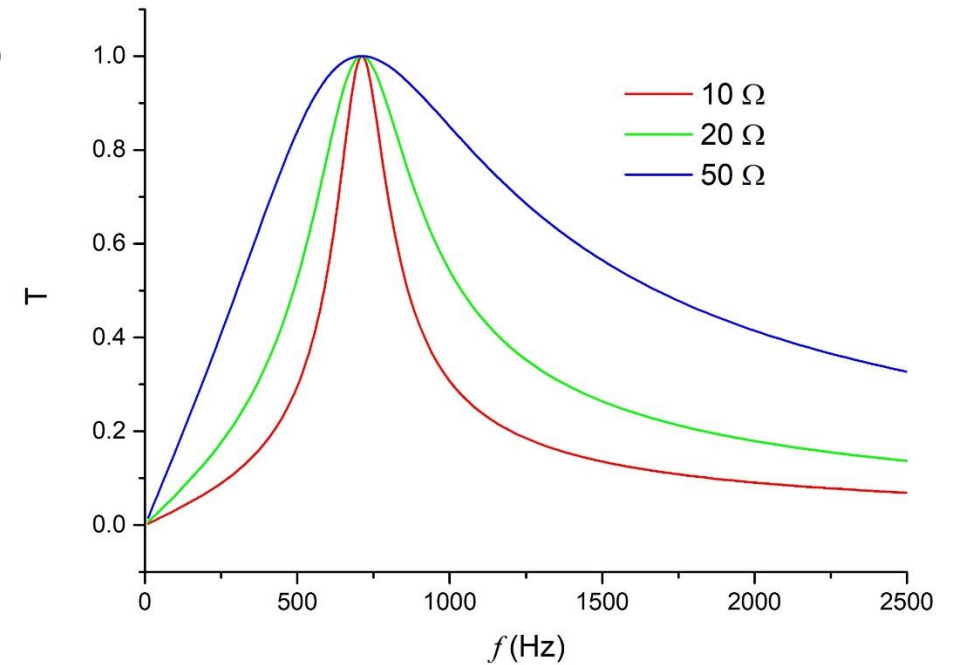
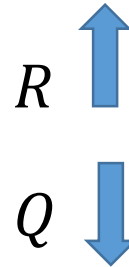
Ancho de banda

$$\Delta\omega = \omega_2 - \omega_1 = \frac{R}{L}$$

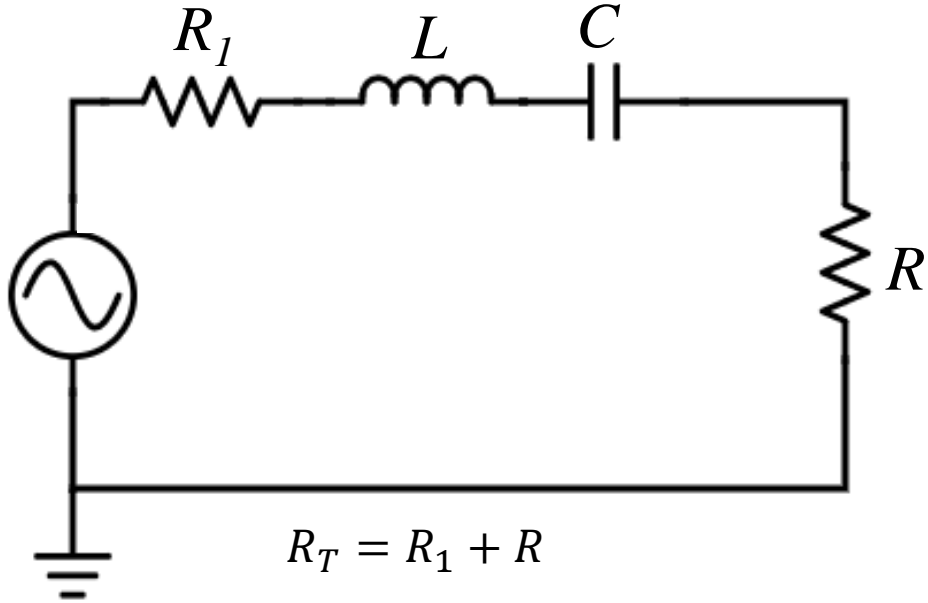
Factor de mérito

$$Q = \frac{\omega_0}{\omega_2 - \omega_1}$$

$$Q = \omega_0 \frac{L}{R}$$

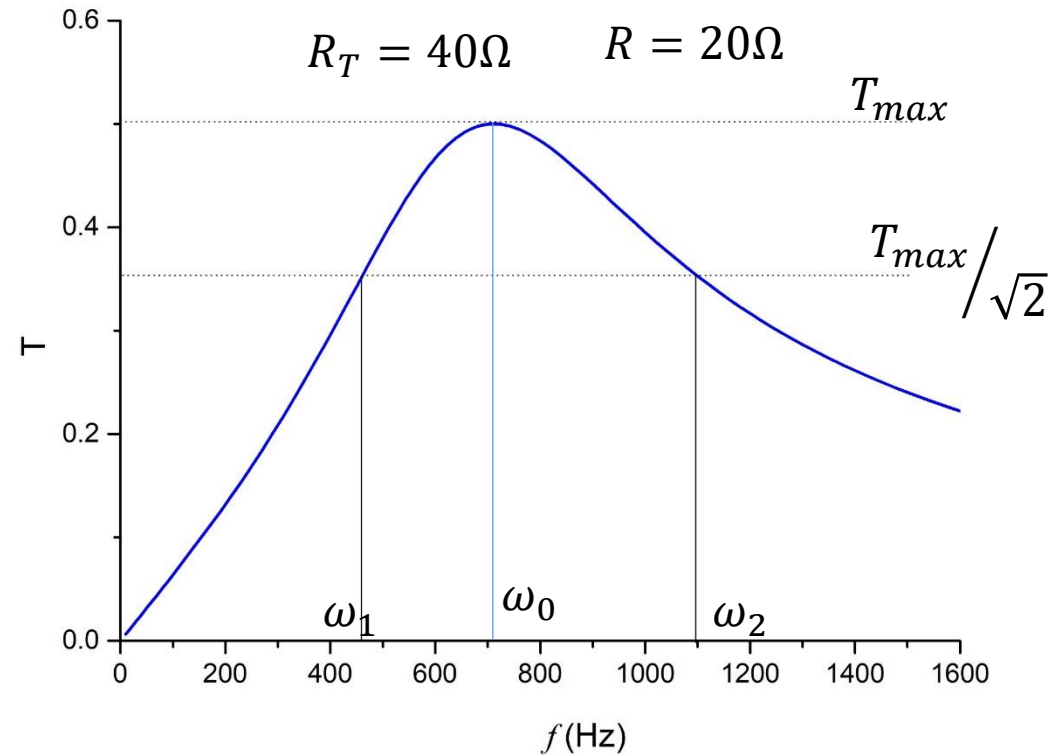


RLC serie con dos resistencias

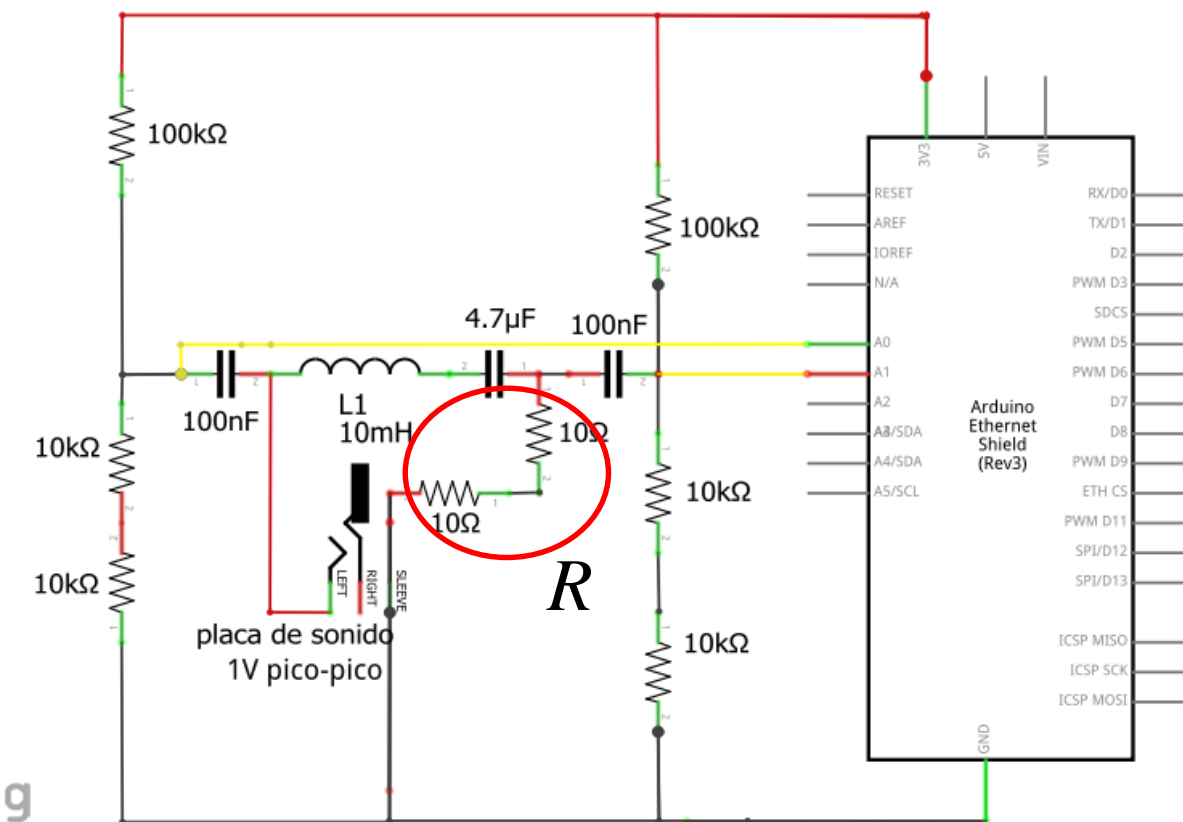
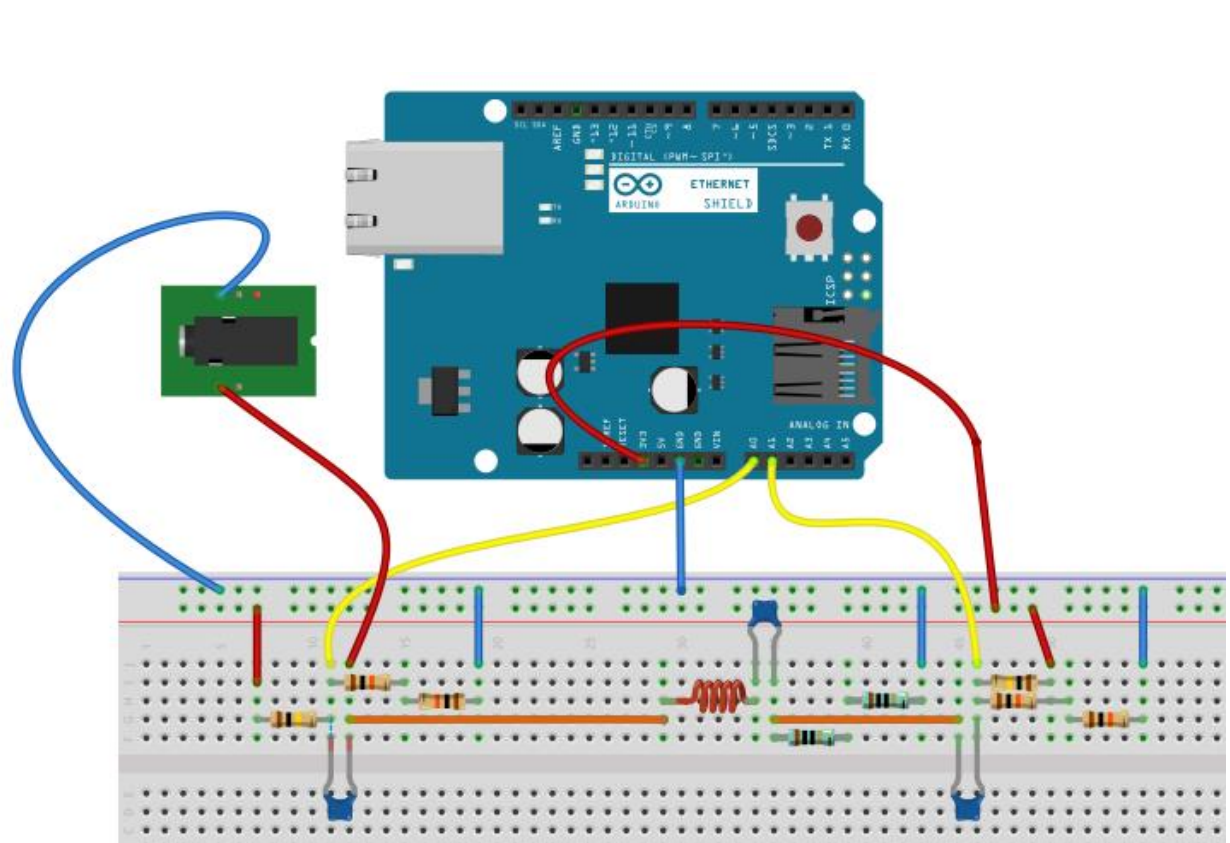


$$T = \frac{T_{max}}{\sqrt{1 + \left(Q \frac{\omega}{\omega_0}\right)^2 \left(1 - \left(\frac{\omega_0}{\omega}\right)^2\right)^2}}$$

$$T = \frac{R}{\sqrt{R_T^2 + \left(\omega L - \frac{1}{\omega C}\right)^2}} \quad T_{max} = \frac{R}{R_T} \quad Q = \omega_0 \frac{L}{R_T}$$

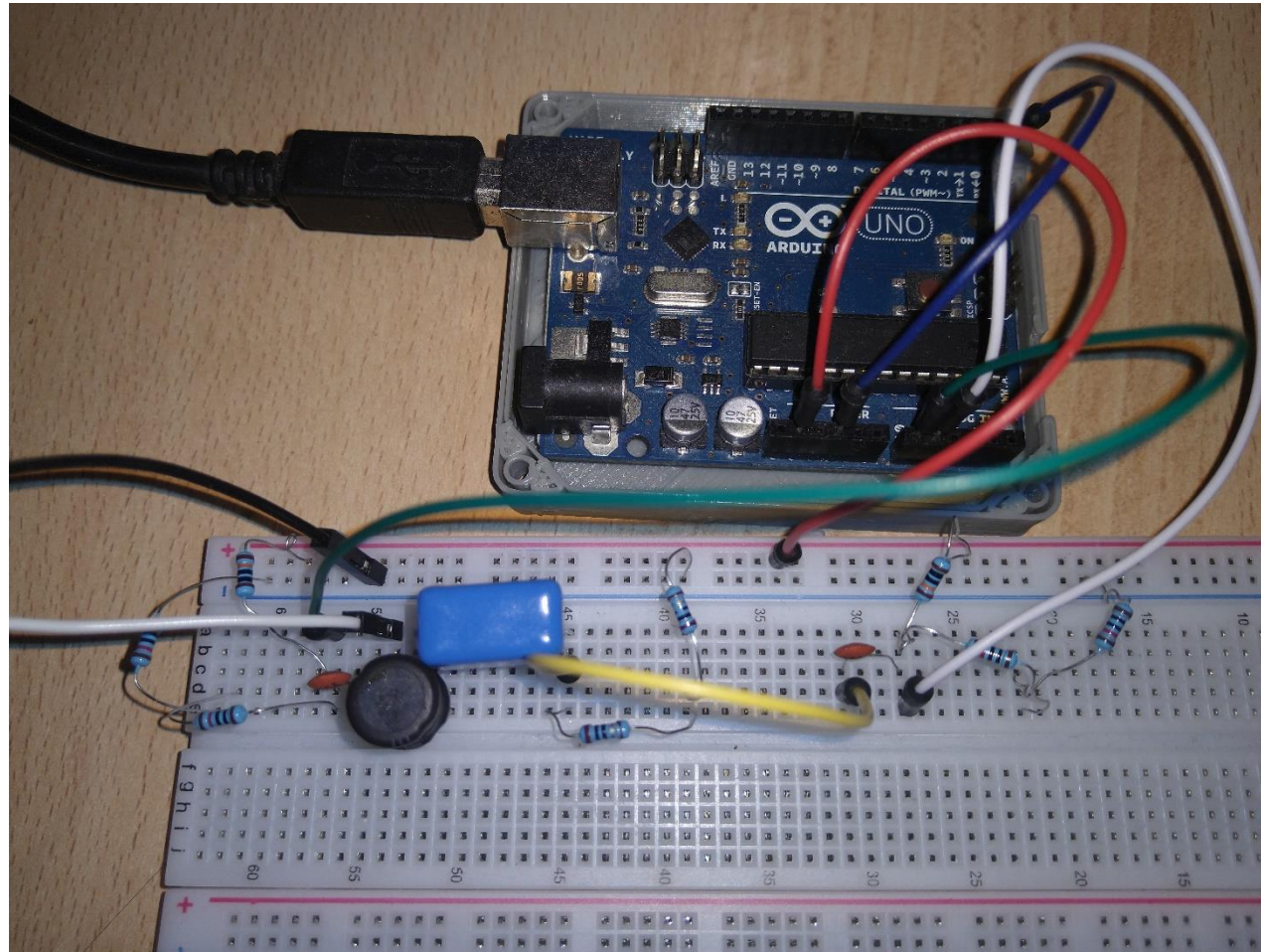


Experimento RLC serie con alimentación AC



Realizar el experimento con $R = 20 \Omega$ y $R = 51.1 \Omega$

Experimento RLC serie con alimentación AC



Emplear el IDE

Doscanales_frec_ampli_fase_v2

Se registra la señal de entrada y la salida sobre la resistencia

Tenemos 6 columnas de salida: tiempo, V1, V2, Amplitud1, Amplitud2, frecuencia1

Registramos y guardamos la señal con Serial Plot



```
Doscanales_frec_ampli_fase Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda

* poco ruido y se logran adquirir 2 máximos y 2 mínimos (> 2 periodos) con un mínimo
* de 10 puntos por período.
*
* Hay dos formas de modificar la frecuencia de adquisición:
* -1) Cambiar el valor de "espera" según se indica más abajo, particularmente útil para
* medir señales de frecuencias "bajas" (< 400 Hz).
* -2) Cambiar el prescaler del microprocesador que es un factor de división de la frecuencia
* del "clock" de 16 MHz del Arduino UNO. Esto es útil para aumentar la velocidad de adquisición
* para frecuencias > 2 kHz. Ojo! Para frecuencias de adquisición elevadas (prescaler bajo) se
* pueden perder bits de resolución en el ADC. Valor x defecto 04-->/16 aumentar a 03-->/8 para
* adquirir señales de frecuencia > 2kHz
*/

int espera=0; // Cambiar para ajustar la ventana de datos a la frecuencia medida
// espera=0--> 1 medición cada 55 microseg // 30000 para frec=1-2 Hz
// 0 para frec=451-1800 Hz // 100 161-450 Hz // 300 74-160 Hz // 1000 26-73 Hz // 3000 9-25 Hz // 10000 3-8 Hz

int canal1[130];
int canal2[130];
long tiempo[130]; // para que no resetee cada 32 ms (aprox)!
float V1;
float V2;

float maxV, minV, maxV11, minV11, maxV12, minV12, maxV21, minV21, maxV22, minV22;
int maxIndice, minIndice, maxIndice11, maxIndice12, minIndice11, minIndice12, Flagmax1, Flagmin1;
int maxIndice21, maxIndice22, minIndice21, minIndice22, Flagmax2, Flagmin2, maxilocal, minilocal;
float Amplitud1, Amplitud2, defasaje, frec1, frec2;
int Flag;
int ventana=8; // fija la ventana local (# de puntos) para determinar los picos, frecuencias, etc
```

Comienzan midiendo con la referencia en Default para ajustar el máximo de la tensión de entrada por debajo de 1.1V

Con esta referencia el canal mide entre 0-5V

```
Doscanales_frec_ampli_fase Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda

Serial.begin(57600); // setea la velocidad de transferencia de datos al puerto serie

// Estas instrucciones son para setear parámetros del microprocesador
// Se modifica el prescaler, se habilita la conversión y se fija el valor máximo
// de las señales a adquirir.

ADCSRA = (ADCSRA & 0xf8) | 0x04; // 0x --> Hexa, f8--> 1111 1000, los 000 resetean los bits del prescaler
// cambiar a 0x03 sólo para adquirir señales entre 1.5 -4 kHz y mantener
ADCSRA |= 1<<ADEN; // habilita la conversión ADC
analogReference(DEFAULT); //para señales de hasta 5 V
//analogReference(INTERNAL); //la señal debe ser inferior a 1.1V mid con 10bits en 1.1 V
}

for (int i=0; i<130; i++){
  Serial.print(tiempo[i]/1000000.00000,5); // tiempo en segundos con 5 dígitos de resolución
  Serial.print(',');
  V1 = canal1[i]*5.0/1023;
  Serial.print(V1,3) ;
  Serial.print(',');
  V2 = canal2[i]*5.0/1023;
  Serial.print(V2,3);
  Serial.print(',');
  Serial.print(Amplitud1,3);
  Serial.print(',');
  Serial.print(Amplitud2,3);
  Serial.print(',');
  Serial.println(frec1,1);
  //Serial.print(',');
  // Serial.print(Amplitud2/Amplitud1,3);
  // Serial.print(',');
  // Serial.println(defasaje,1);
}
```

Modificamos la referencia a Internal. La tensión de entrada debe ser inferior a 1.1V

Mejoramos la resolución en la medición de la señal: tenemos 10 bits en 1.1V

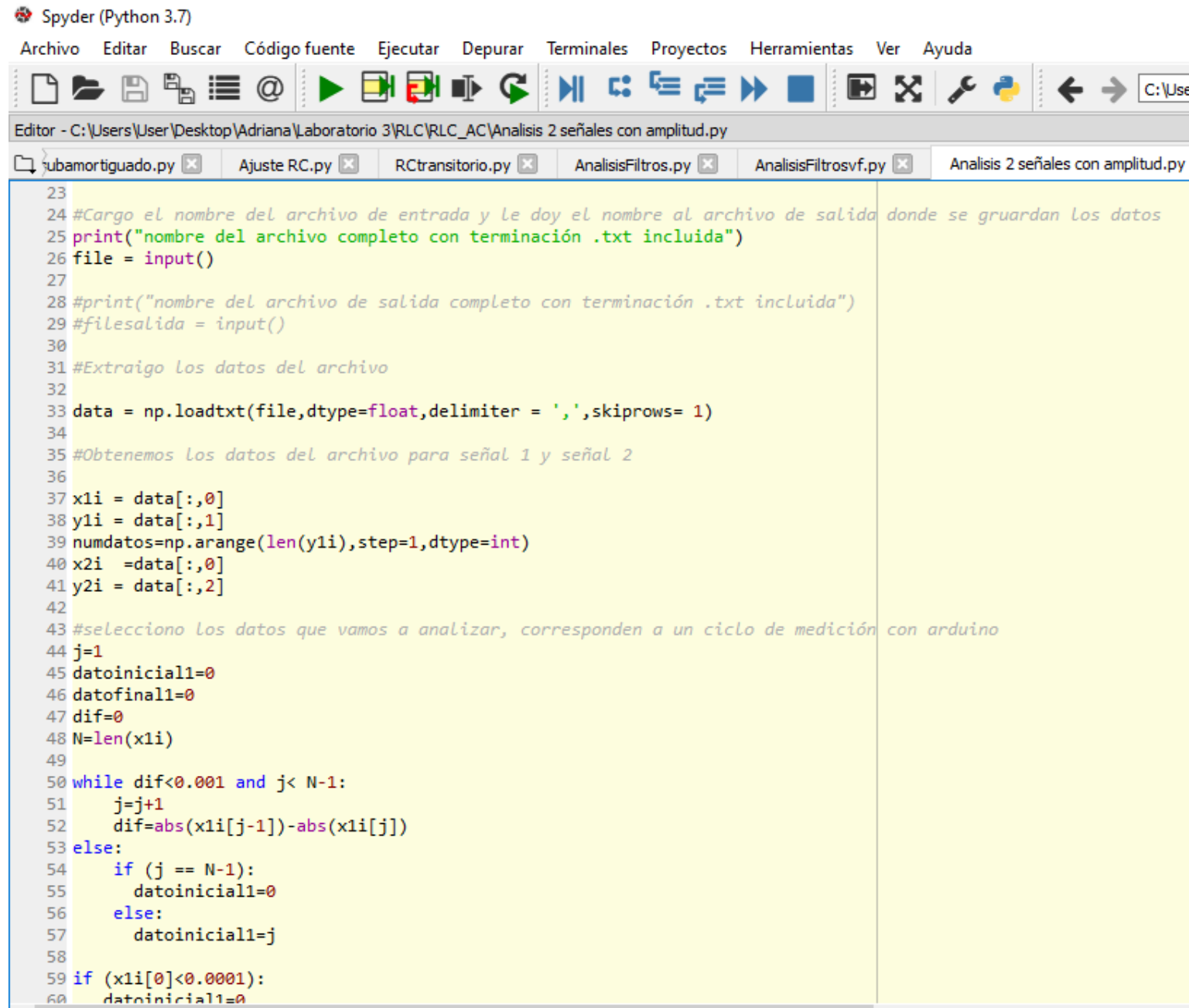
```
ADCSRA = (ADCSRA & 0xf8) | 0x04; // 0x --> Hexa, f8--> 1111 1000, los 000 resetean los bits del prescaler
// cambiar a 0x03 sólo para adquirir señales entre 1.5 -4 kHz y mantener la resolución a 10 bits
ADCSRA |= 1<<ADEN; // habilita la conversión ADC
//analogReference(DEFAULT); //para señales de hasta 5 V
analogReference(INTERNAL); //la señal debe ser inferior a 1.1V mid con 10bits en 1.1 V
}

void loop() {
    // aquí se ponen las instrucciones que se ejecutarán en forma de loop (indefinidamente)

    for (int i=0; i<130; i++){
        Serial.print(tiempo[i]/1000000.00000,5); // tiempo en segundos con 5 dígitos de resolución
        Serial.print(',');
        V1 = canal1[i]*1.1/1023;
        Serial.print(V1,3) ;
        Serial.print(',');
        V2 = canal2[i]*1.1/1023;
        Serial.print(V2,3);
        Serial.print(',');
        Serial.print(Amplitud1,3);
        Serial.print(',');
        Serial.print(Amplitud2,3);
        Serial.print(',');
        Serial.println(frec1,1);
        //Serial.print(',');
        // Serial.print(Amplitud2/Amplitud1,3);
        // Serial.print(',');
        // Serial.println(defasaje,1);
    }
}
```

Analizamos la señales con Python

Script Análisis 2 señales con amplitud



The image shows a screenshot of the Spyder Python IDE (Python 3.7) with a script titled 'Análisis 2 señales con amplitud.py'. The script is written in Python and uses NumPy for data processing. It prompts the user for an input file name and then processes the data to analyze two signals. The script includes comments in Spanish explaining each step, from loading the data to calculating differences between signal points.

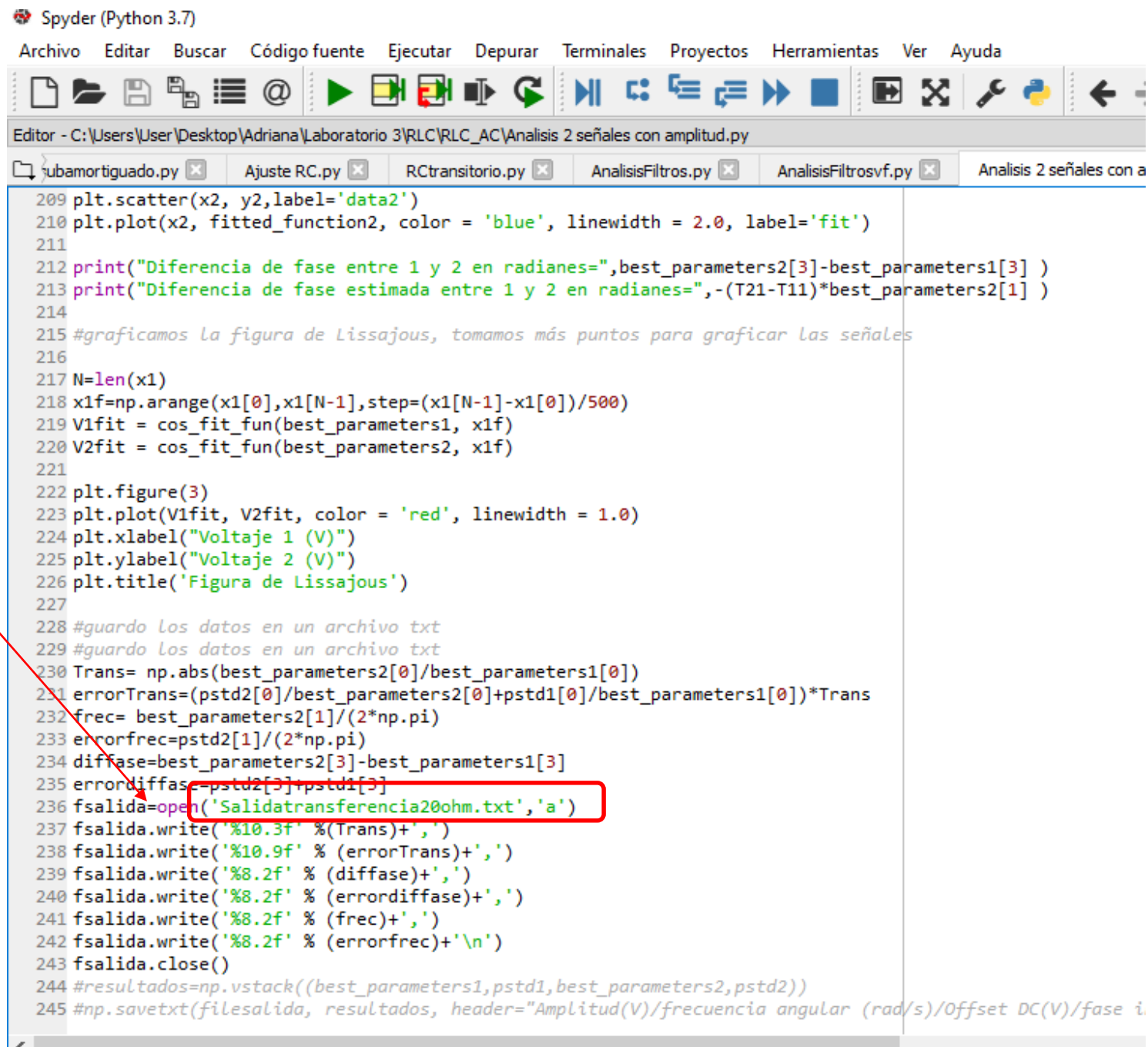
```
23
24 #Cargo el nombre del archivo de entrada y le doy el nombre al archivo de salida donde se guardan los datos
25 print("nombre del archivo completo con terminación .txt incluida")
26 file = input()
27
28 #print("nombre del archivo de salida completo con terminación .txt incluida")
29 #filesalida = input()
30
31 #Extraigo los datos del archivo
32
33 data = np.loadtxt(file, dtype=float, delimiter = ',', skiprows= 1)
34
35 #Obtenemos los datos del archivo para señal 1 y señal 2
36
37 x1i = data[:,0]
38 y1i = data[:,1]
39 numdatos=np.arange(len(y1i), step=1, dtype=int)
40 x2i = data[:,0]
41 y2i = data[:,2]
42
43 #selecciono los datos que vamos a analizar, corresponden a un ciclo de medición con arduino
44 j=1
45 datoinicial1=0
46 datofinal1=0
47 dif=0
48 N=len(x1i)
49
50 while dif<0.001 and j< N-1:
51     j=j+1
52     dif=abs(x1i[j-1])-abs(x1i[j])
53 else:
54     if (j == N-1):
55         datoinicial1=0
56     else:
57         datoinicial1=j
58
59 if (x1i[0]<0.0001):
60     datoinicial1=0
```

La primera vez que corren el script crea un archivo en el mismo directorio de los archivos de entrada

Colocar el nombre del archivo de salida

Cada vez que analizan una señal guarda los datos de salida en el archivo de salida

Chequear que el ajuste es correcto, si no lo es borrar los datos en el archivo de salida



```
Spyder (Python 3.7)
Archivo  Editar  Buscar  Código fuente  Ejecutar  Depurar  Terminales  Proyectos  Herramientas  Ver  Ayuda

Editor - C:\Users\User\Desktop\Adriana\Laboratorio 3\RLC\RLC_AC\Análisis 2 señales con amplitud.py

subamortiguado.py x  Ajuste RC.py x  RCTransitorio.py x  AnalisisFiltros.py x  AnalisisFiltrosvf.py x  Analisis 2 señales con a

209 plt.scatter(x2, y2, label='data2')
210 plt.plot(x2, fitted_function2, color = 'blue', linewidth = 2.0, label='fit')
211
212 print("Diferencia de fase entre 1 y 2 en radianes=", best_parameters2[3]-best_parameters1[3] )
213 print("Diferencia de fase estimada entre 1 y 2 en radianes=", -(T21-T11)*best_parameters2[1] )
214
215 #graficamos la figura de Lissajous, tomamos más puntos para graficar las señales
216
217 N=len(x1)
218 x1f=np.arange(x1[0],x1[N-1],step=(x1[N-1]-x1[0])/500)
219 V1fit = cos_fit_fun(best_parameters1, x1f)
220 V2fit = cos_fit_fun(best_parameters2, x1f)
221
222 plt.figure(3)
223 plt.plot(V1fit, V2fit, color = 'red', linewidth = 1.0)
224 plt.xlabel("Voltaje 1 (V)")
225 plt.ylabel("Voltaje 2 (V)")
226 plt.title('Figura de Lissajous')
227
228 #guardo los datos en un archivo txt
229 #guardo los datos en un archivo txt
230 Trans= np.abs(best_parameters2[0]/best_parameters1[0])
231 errorTrans=(pstd2[0]/best_parameters2[0]+pstd1[0]/best_parameters1[0])*Trans
232 frec= best_parameters2[1]/(2*np.pi)
233 errorfrec=pstd2[1]/(2*np.pi)
234 diffase=best_parameters2[3]-best_parameters1[3]
235 errordiffase=pstd2[3]+pstd1[3]
236 fsalida=open('Salidatransferencia20ohm.txt','a')
237 fsalida.write('%10.3f' %(Trans)+',')
238 fsalida.write('%10.9f' % (errorTrans)+',')
239 fsalida.write('%8.2f' % (diffase)+',')
240 fsalida.write('%8.2f' % (errordiffase)+',')
241 fsalida.write('%8.2f' % (frec)+',')
242 fsalida.write('%8.2f' % (errorfrec)+'\n')
243 fsalida.close()
244 #resultados=np.vstack((best_parameters1,pstd1,best_parameters2,pstd2))
245 #np.savetxt(filesalida, resultados, header="Amplitud(V)/frecuencia angular (rad/s)/Offset DC(V)/fase i
```

Analizamos el archivo de salida del script Analisis 2 señales

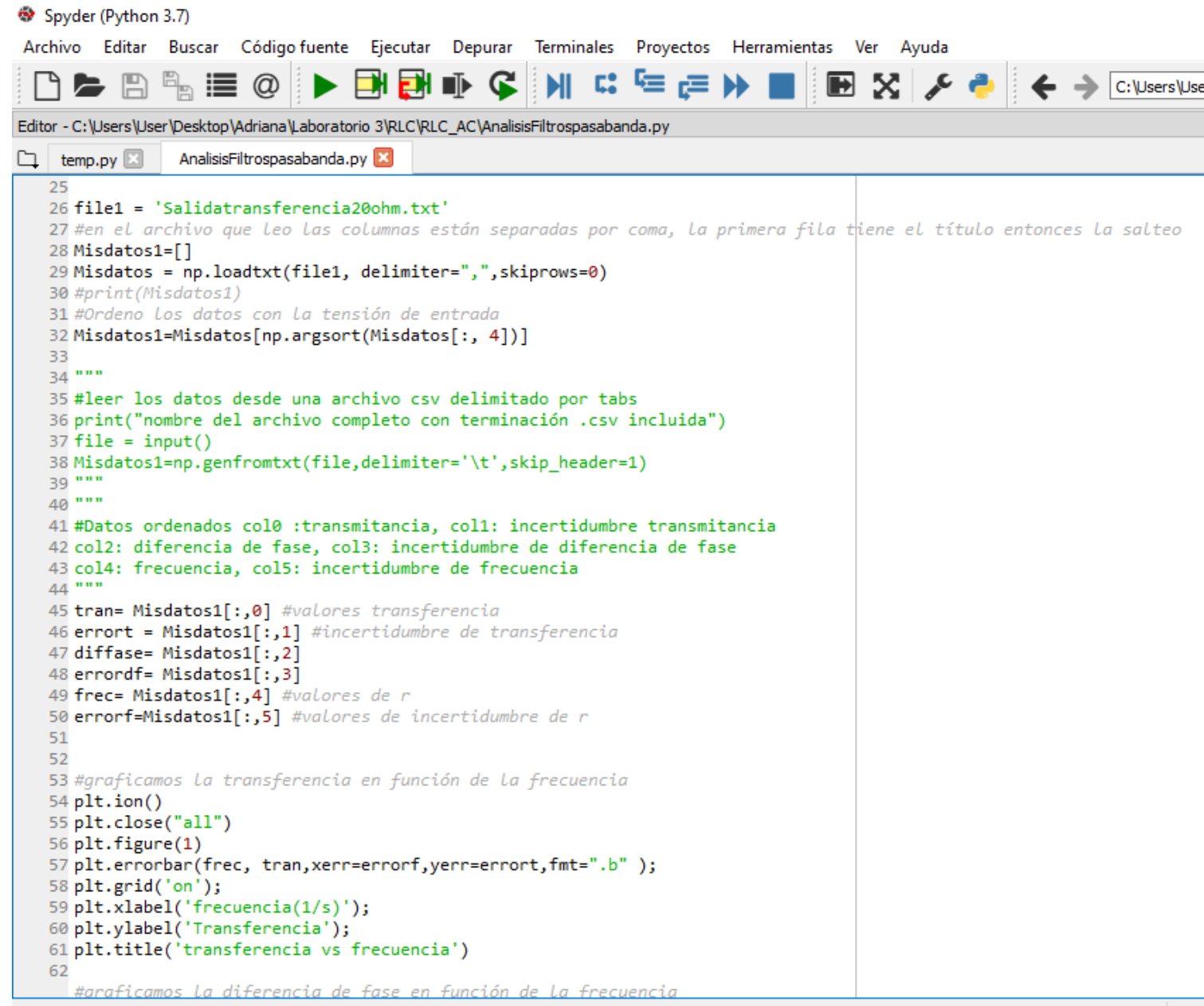
Script

Analisisfiltrospasabanda

Graficamos la transferencia y la diferencia de fase en función de la frecuencia

Del gráfico estimamos

T_{max} ω_0 ω_1 ω_2



```
Spyder (Python 3.7)
Archivo  Editar  Buscar  Código fuente  Ejecutar  Depurar  Terminales  Proyectos  Herramientas  Ver  Ayuda

Editor - C:\Users\User\Desktop\Adriana\Laboratorio 3\RLC\RLC_AC\AnalisisFiltrospasabanda.py

temp.py x AnalisisFiltrospasabanda.py x

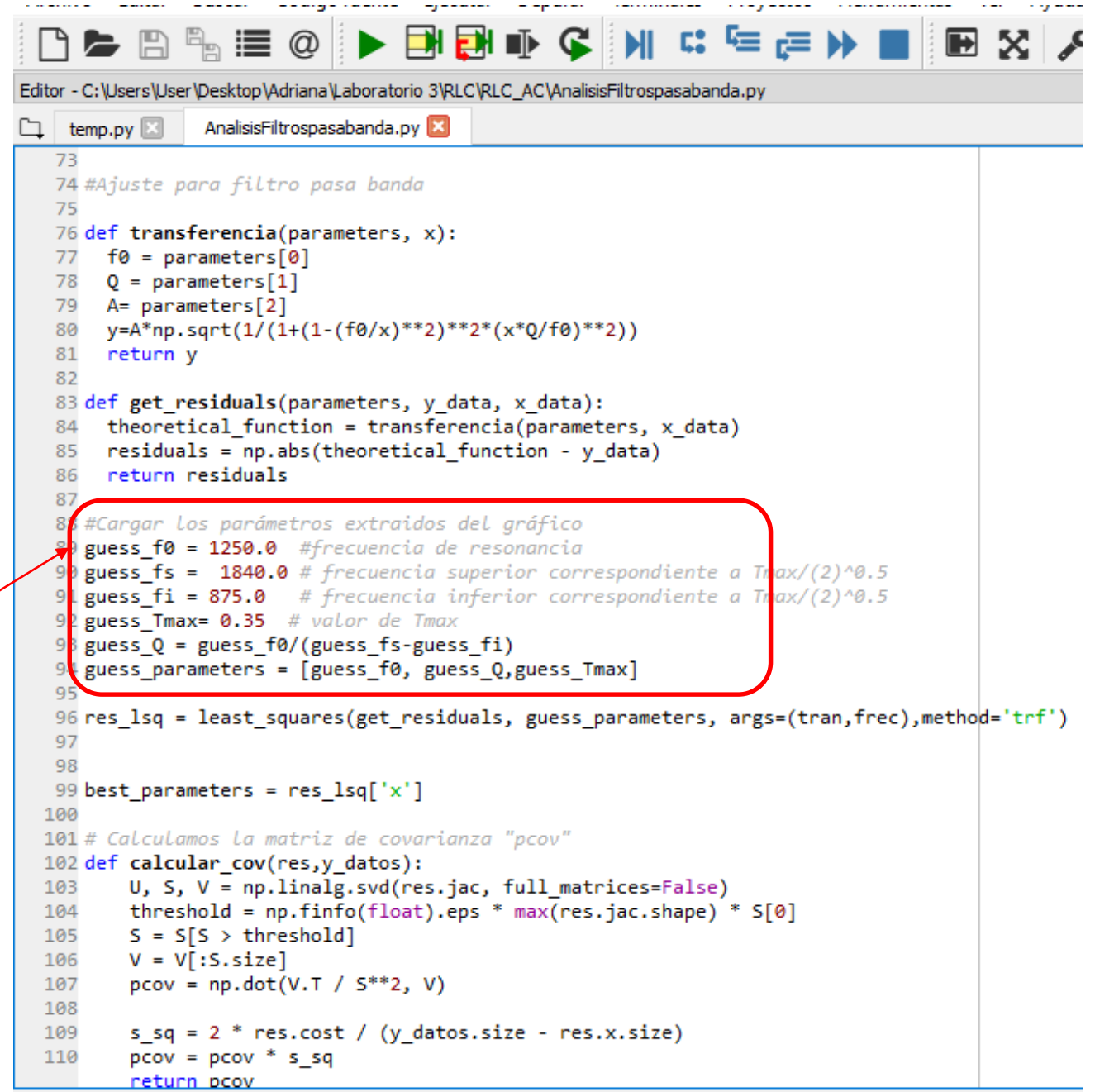
25
26 file1 = 'Salidatransferencia20ohm.txt'
27 #en el archivo que leo las columnas están separadas por coma, la primera fila tiene el título entonces la salteo
28 Misdatos1=[]
29 Misdatos = np.loadtxt(file1, delimiter=",", skiprows=0)
30 #print(Misdatos1)
31 #Ordeno los datos con la tensión de entrada
32 Misdatos1=Misdatos[np.argsort(Misdatos[:, 4])]
33
34 """
35 #leer los datos desde una archivo csv delimitado por tabs
36 print("nombre del archivo completo con terminación .csv incluida")
37 file = input()
38 Misdatos1=np.genfromtxt(file,delimiter='\t',skip_header=1)
39 """
40 """
41 #Datos ordenados col0 :transmitancia, col1: incertidumbre transmitancia
42 col2: diferencia de fase, col3: incertidumbre de diferencia de fase
43 col4: frecuencia, col5: incertidumbre de frecuencia
44 """
45 tran= Misdatos1[:,0] #valores transferencia
46 errorrt = Misdatos1[:,1] #incertidumbre de transferencia
47 diffase= Misdatos1[:,2]
48 errorrdf= Misdatos1[:,3]
49 frec= Misdatos1[:,4] #valores de r
50 errorrf=Misdatos1[:,5] #valores de incertidumbre de r
51
52
53 #graficamos la transferencia en función de la frecuencia
54 plt.ion()
55 plt.close("all")
56 plt.figure(1)
57 plt.errorbar(frec, tran,xerr=errorrf,yerr=errorrt,fmt=".b" );
58 plt.grid('on');
59 plt.xlabel('frecuencia(1/s)');
60 plt.ylabel('Transferencia');
61 plt.title('transferencia vs frecuencia')
62
#graficamos la diferencia de fase en función de la frecuencia
```


Corremos la segunda etapa del programa

Proponemos un ajuste para la transferencia vs frecuencia

Cargar los valores tomados del gráfico en el script.

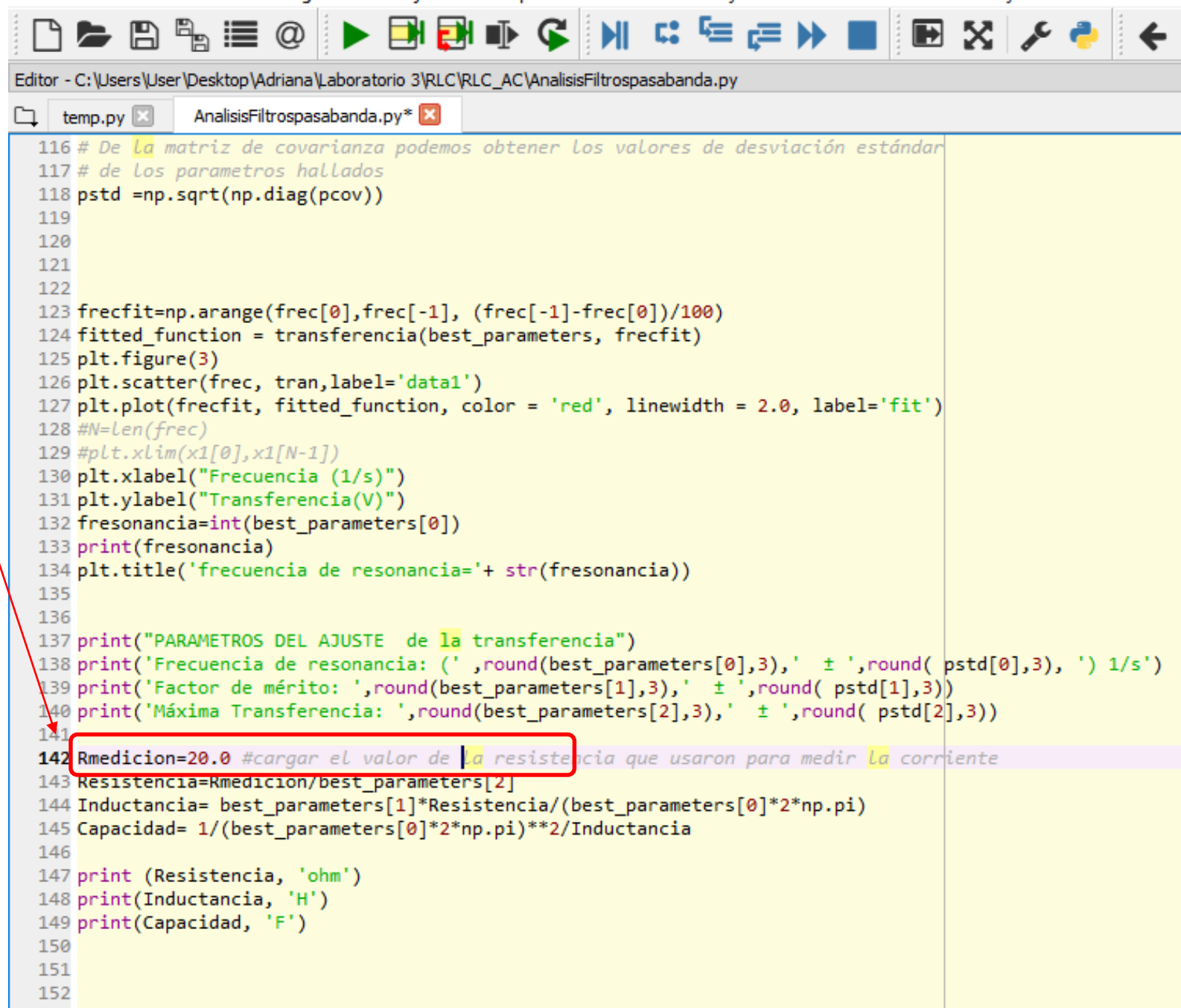
Los usamos para estimar valores iniciales para el ajuste.



```
73
74 #Ajuste para filtro pasa banda
75
76 def transferencia(parameters, x):
77     f0 = parameters[0]
78     Q = parameters[1]
79     A= parameters[2]
80     y=A*np.sqrt(1/(1+(1-(f0/x)**2)**2*(x*Q/f0)**2))
81     return y
82
83 def get_residuals(parameters, y_data, x_data):
84     theoretical_function = transferencia(parameters, x_data)
85     residuals = np.abs(theoretical_function - y_data)
86     return residuals
87
88 #Cargar los parámetros extraídos del gráfico
89 guess_f0 = 1250.0 #frecuencia de resonancia
90 guess_fs = 1840.0 # frecuencia superior correspondiente a Tmax/(2)0.5
91 guess_fi = 875.0 # frecuencia inferior correspondiente a Tmax/(2)0.5
92 guess_Tmax= 0.35 # valor de Tmax
93 guess_Q = guess_f0/(guess_fs-guess_fi)
94 guess_parameters = [guess_f0, guess_Q, guess_Tmax]
95
96 res_lsq = least_squares(get_residuals, guess_parameters, args=(tran,frec),method='trf')
97
98
99 best_parameters = res_lsq['x']
100
101 # Calculamos la matriz de covarianza "pcov"
102 def calcular_cov(res,y_datos):
103     U, S, V = np.linalg.svd(res.jac, full_matrices=False)
104     threshold = np.finfo(float).eps * max(res.jac.shape) * S[0]
105     S = S[S > threshold]
106     V = V[:,S.size]
107     pcov = np.dot(V.T / S**2, V)
108
109     s_sq = 2 * res.cost / (y_datos.size - res.x.size)
110     pcov = pcov * s_sq
111     return pcov
```

Cargar el valor de la resistencia que emplearon para medir la corriente

Del ajuste obtenemos los parámetros del circuito



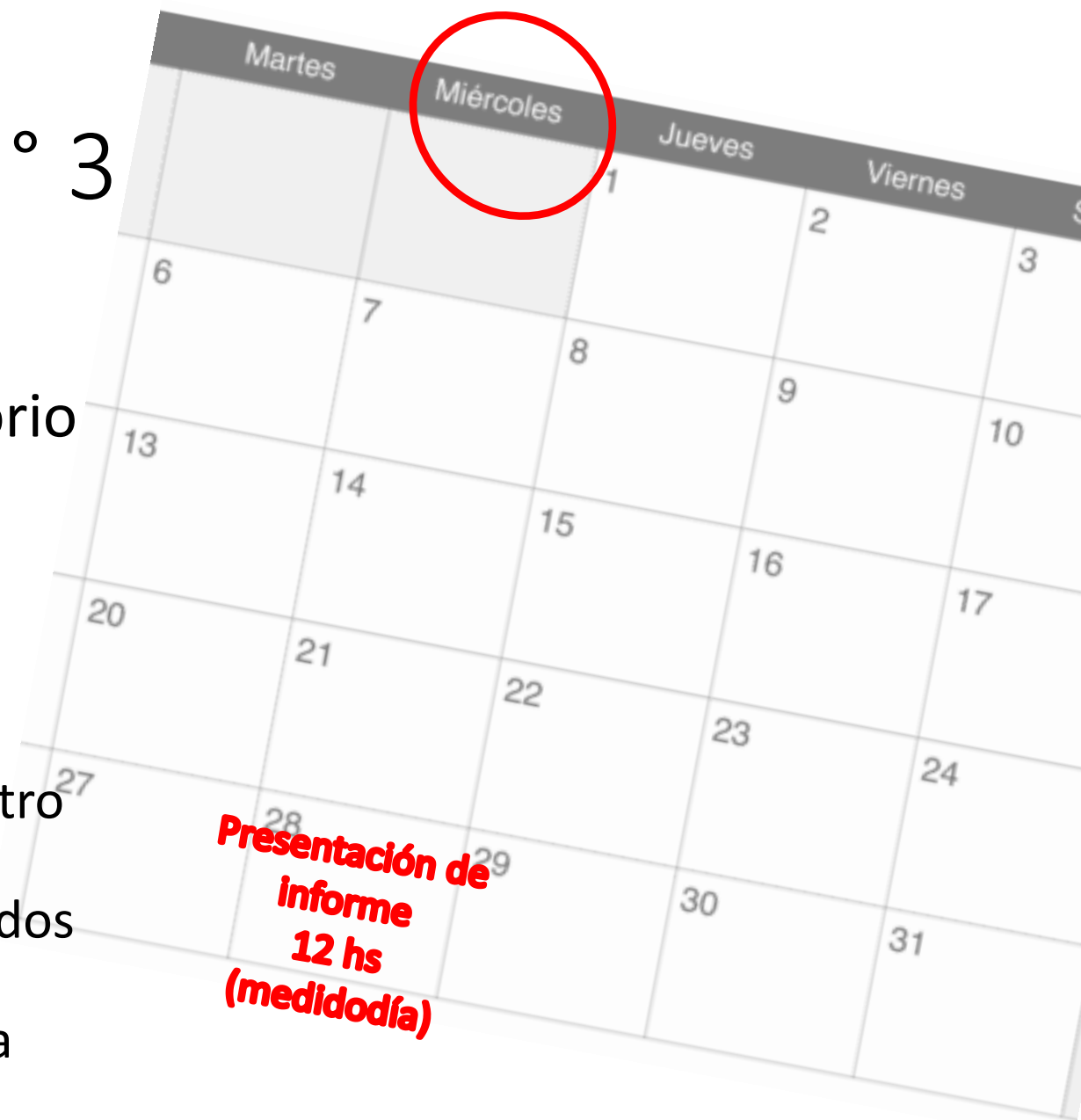
```
Editor - C:\Users\User\Desktop\Adriana\Laboratorio 3\RLC\RLC_AC\AnálisisFiltrosPasabanda.py
temp.py x AnalisisFiltrosPasabanda.py* x
116 # De la matriz de covarianza podemos obtener los valores de desviación estándar
117 # de los parámetros hallados
118 pstd = np.sqrt(np.diag(pcov))
119
120
121
122
123 frecfit=np.arange(frec[0],frec[-1], (frec[-1]-frec[0])/100)
124 fitted_function = transferencia(best_parameters, frecfit)
125 plt.figure(3)
126 plt.scatter(frec, tran,label='data1')
127 plt.plot(frecfit, fitted_function, color = 'red', linewidth = 2.0, label='fit')
128 #N=len(frec)
129 #plt.xlim(x1[0],x1[N-1])
130 plt.xlabel("Frecuencia (1/s)")
131 plt.ylabel("Transferencia(V)")
132 fresonancia=int(best_parameters[0])
133 print(fresonancia)
134 plt.title('frecuencia de resonancia='+ str(fresonancia))
135
136
137 print("PARAMETROS DEL AJUSTE de la transferencia")
138 print('Frecuencia de resonancia: (',round(best_parameters[0],3),' ± ',round(pstd[0],3), ' ) 1/s')
139 print('Factor de mérito: ',round(best_parameters[1],3),' ± ',round(pstd[1],3))
140 print('Máxima Transferencia: ',round(best_parameters[2],3),' ± ',round(pstd[2],3))
141
142 Rmedicion=20.0 #cargar el valor de la resistencia que usaron para medir la corriente
143 Resistencia=Rmedicion/best_parameters[2]
144 Inductancia= best_parameters[1]*Resistencia/(best_parameters[0]*2*np.pi)
145 Capacidad= 1/(best_parameters[0]*2*np.pi)**2/Inductancia
146
147 print (Resistencia, 'ohm')
148 print(Inductancia, 'H')
149 print(Capacidad, 'F')
150
151
152
---
```

Punto de control

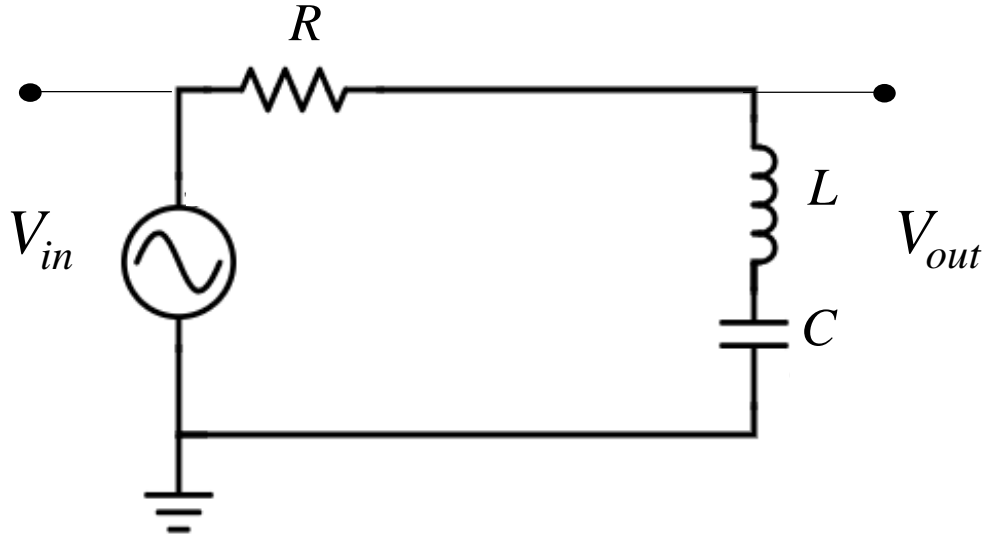
- Estudiar la transferencia y la diferencia de fase del circuito RLC serie midiendo la tensión de salida sobre R
- Realizar el estudio para $R= 20 \, \Omega$ y $R= 51.1 \, \Omega$
- Determinar frecuencia de resonancia, ancho de banda y factor de mérito para cada circuito

Temas para el informe N° 3

- RLC serie estudio de la respuesta transitoria comportamiento oscilatorio subamortiguado
 - Determinación de R y L
- RLC serie con tensión AC
 - Caracterizar la respuesta del filtro pasa banda
 - Comparar la respuesta con las dos resistencias
 - Simulación filtro rechaza banda



RLC serie medición sobre R – L



$$T = \frac{\left| \omega L - \frac{1}{\omega C} \right|}{\sqrt{R^2 + \left(\omega L - \frac{1}{\omega C} \right)^2}}$$

$$V_{out} = \frac{j \left(\omega L - \frac{1}{\omega C} \right) V}{R + j \left(\omega L - \frac{1}{\omega C} \right)}$$

$$T = \left| \frac{V_{out}}{V_{in}} \right|$$

$$T = \left| \frac{j \left(\omega L - \frac{1}{\omega C} \right) I}{V_0} \right|$$

$$\tan \varphi = \frac{R}{\left(\omega L - \frac{1}{\omega C} \right)}$$

Respuesta en función de la frecuencia

$$T = \frac{\left| \omega L - \frac{1}{\omega C} \right|}{\sqrt{R^2 + \left(\omega L - \frac{1}{\omega C} \right)^2}}$$

Frecuencia de rechazo

$$\omega_0 = \frac{1}{\sqrt{LC}}$$

Ancho de banda de rechazo

$$\Delta\omega = \omega_2 - \omega_1$$

$$\tan \varphi = \frac{R}{\left(\omega L - \frac{1}{\omega C} \right)}$$

