



### Unité 7 : Utiliser la bibliothèque cmath

### Application : Nombres complexes et sciences

Dans cette application de l'unité 7, vous allez utiliser la bibliothèque **cmath** pour effectuer des calculs et représenter des nombres complexes utilisés en sciences physiques pour l'étude d'un circuit électrique.

#### Objectifs :

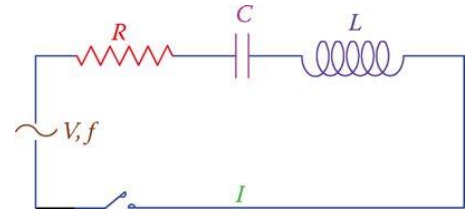
- Découvrir le module **cmath**.
- Utiliser les fonctions de la bibliothèque **cmath**.
- Représenter graphiquement des nombres complexes.
- Analyser un circuit électrique RLC série.

#### Le circuit RLC série.

Un **circuit RLC** en électrocinétique est un circuit linéaire contenant une résistance électrique, une bobine (inductance) et un condensateur (capacité).

Il existe deux types de circuits **RLC série** ou *parallèle*, selon l'interconnexion des trois types de composants. Le comportement d'un circuit RLC est généralement décrit par une équation différentielle du second ordre (là où des circuits RL ou RC modélisés par des équations différentielles du premier ordre).

A l'aide d'un générateur de signaux, on peut injecter dans le circuit des oscillations et observer dans certains cas une résonance, caractérisée par une augmentation du courant (lorsque le signal d'entrée choisi correspond à la pulsation propre du circuit, calculable à partir de l'équation différentielle qui le régit).



Dans un dipôle linéaire, pas forcément élémentaire, mais constitué d'un assemblage d'éléments linéaires passifs R,L,C si l'on prend l'équation qui lie la tension au courant et que l'on y applique une tension  $\bar{U} = U \times e^{j(\omega t - \varphi)}$ , on obtiendra un courant  $\bar{I} = I \times e^{j(\omega t - \varphi - \psi)}$ . On appelle impédance complexe du dipôle la quantité :

$$\bar{Z} = \frac{\bar{U}}{\bar{I}} = Z \times e^{\psi}$$

A quoi sert la notion d'impédance ?

Si on connaît le module Z et un argument  $\varphi$  du dipôle, on peut immédiatement passer de la tension au courant ou réciproquement : le module Z indique le rapport entre l'amplitude de la tension et celle du courant.

Un argument  $\varphi$  donne quant à lui le déphasage entre la tension et le courant.

$\omega$  représente la pulsation du signal électrique.

Pour rappel  $\omega = 2\pi f$  ; f étant la fréquence du signal exprimée en Hertz (Hz).





**Impédance complexe.**

Résistor de résistance R :  $\bar{Z} = R$

Bobine d'inductance L :  $\bar{Z} = jL\omega$

Condensateur de capacité C :  $\bar{Z} = \frac{1}{jC\omega}$  ou bien  $\bar{Z} = \frac{-j}{C\omega}$



**Conseil à l'enseignant :** Une impédance se mesure en ohms ( $\Omega$ ). D'un point de vue physique, on s'intéresse au module de l'impédance. Le déphasage introduit par une inductance pure est :  $\varphi = \frac{\pi}{2}$  et celui introduit par un condensateur pur est :  $\varphi = -\frac{\pi}{2}$ .

**Etude d'un exemple.**

- Créer un script Python afin de déterminer l'impédance complexe d'un circuit RLC série.
- Représenter graphiquement l'impédance de chaque dipôle, puis l'impédance totale.
- En déduire la nature du circuit (dominante inductive ou capacitive).

**Rappel :** Pour des dipôles disposés en série, leurs impédances s'ajoutent. Lorsque les dipôles sont en parallèle, ce sont leurs admittances  $Y$  qui s'ajoutent.  $Y = \frac{1}{Z}$ . (L'admittance est l'inverse de l'impédance).

- Insérer une nouvelle application et choisir le menu **A Ajouter Python**.
- Créer un nouveau script et le nommer U7Apps.
- Importer les bibliothèques **math** et **cmath**.
- Créer une fonction comportant 3 arguments, lesquels étant les valeurs des impédances de chaque dipôle dans l'ordre  $Z_R$ ,  $Z_L$  et  $Z_C$ .
- La fonction devra retourner l'impédance complexe totale, le module de l'impédance totale, un argument, arrondi au dixième de degré.
- 

```

1.1 1.2 *Classeur RAD 8/11
U7Apps.py
from cmath import *
from math import *
# Calcul de l'impédance totale (RLC série)
def impT(zr,zl,zc):
    zt=complex(zr,zl-zc)
    module=round(abs(zt),2)
    arg=round(degrees(phase(zt)),1)
    return zt,module,arg
    
```

**Conseil à l'enseignant :** Si vous le souhaitez, vous pouvez également créer une fonction tenant compte de la fréquence du signal. La fonction aura alors directement pour arguments, les valeurs des dipôles R, L et C, respectivement exprimés en ohms ( $\Omega$ ), henry (H) et farads (F).





- Exécuter le script et déterminer l'impédance totale d'un circuit RLC série tel que :

$$zr = 2\Omega ; zl = 3\Omega ; zc = 1\Omega$$


- La phase est de  $45^\circ$ , le comportement du circuit est à dominante inductive.

```

1.1 1.2 *Classeur RAD 3/3
Shell Python
>>>impT(2,3,1)
((2+2j), 2.83, 45.0)
>>>|

```

Représenter graphiquement le diagramme d'impédances.

- Il s'agit de représenter dans le plan, la somme de trois vecteurs. En électricité, une pratique consiste à représenter à partir de l'origine, le vecteur  $\vec{U}_R(zr, 0)$  puis à partir de son extrémité, le vecteur  $\vec{U}_L(zl, \frac{\pi}{2})$  et enfin également à partir de son extrémité, le vecteur  $\vec{U}_C(zc, -\frac{\pi}{2})$ .
- Importer dans votre script, les bibliothèques **TI PlotLib**. Modifier le script afin qu'il renvoie la valeur de l'impédance complexe après la représentation graphique. L'appui sur la touche  permettra de stopper l'affichage de la représentation graphique.

```

1.1 1.2 *Classeur RAD 2/14
U7Apps.py
from cmath import *
from math import *
import ti_plotlib as plt
from time import *
# Calcul de l'impédance totale (RLC série)
def impT(zr,zl,zc):
    *zt=complex(zr,zl-zc)
    *module=round(abs(zt),2)
    *arg=round(degrees(phase(zt)),1)
    *return zt,module,arg
#Représentation graphique

```

- Le vecteur correspondant à l'impédance totale sera en couleur magenta à l'aide de l'instruction **plt.color(255,0,255)**.

```

1.1 1.2 *Classeur RAD 11/29
U7Apps.py
#Représentation graphique
*plt.cls()
*plt.window(-1,5,-1,5)
*plt.title("Diagramme d'impédance")
*plt.grid(1,1,"dashed")
*plt.pen("medium","solid")
*plt.color(0,0,255)
*plt.line(0,0,zr,0,"arrow")
*plt.color(0,255,0)
*plt.line(zr,0,zr,zl,"arrow")
*plt.color(255,0,0)
*plt.line(zr,zl,zr,zl-zc,"arrow")
*plt.color(255,0,255)
*plt.line(0,0,zr,zl-zc,"arrow")
*plt.show_plot()
*return zt,module,arg


```





## 10 Minutes de Code

### TI - NSPIRE™ CX II & TI - PYTHON

- Demander à nouveau l'exécution de votre script.
- Donner les arguments à la fonction permettant de calculer l'impédance totale **impT(2, 3, 1)**.
- Observer la représentation graphique du diagramme d'impédance.
- La touche  permet de retrouver le résultat du calcul précédent.

```
Shell Python
>>>impT(2,3,1)
((2+2j), 2.83, 45.0)
>>>|
```

## UNITE 7 : APPLICATION

### NOTES DU PROFESSEUR

