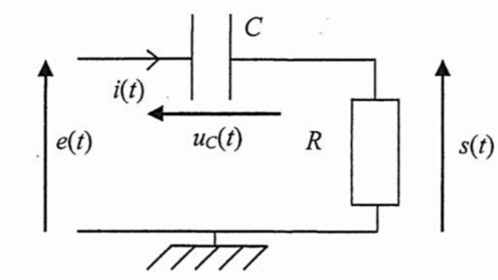
Corrigé du Devoir à la maison 12

Exercice 1 – Mode AC de l'oscilloscope

1. L'oscilloscope étant branché en dérivation sur le dipôle dont on souhaite mesurer la tension, il faut que la résistance d'entrée R de l'oscilloscope soit la plus grande possible (très grande devant l'impédance du dipôle à étudié) pour qu'il ne perturbe pas le circuit étudié (aucun courant absorbé par l'oscilloscope).
2. En mode DC : $\boxed{s(t)=e(t)}$
3. BF : C équivalent à un interrupteur ouvert : $s(t)=Ri(t)=0$

   HF : C équivalent à un interrupteur fermé : $s(t)=e(t)$

   Filtre passe-haut d'ordre 1
4. Loi des mailles :



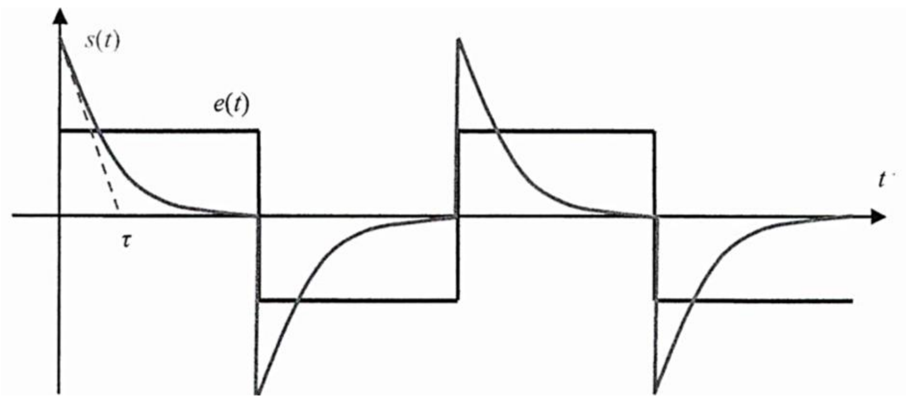
   $$e(t)=u_C(t)+s(t) \quad \text{et} \quad i(t)=C\frac{du_C(t)}{dt}$$

   Dérivation :
   $$\frac{de(t)}{dt}=\frac{du_C(t)}{dt}+\frac{ds(t)}{dt}=\frac{i(t)}{C}+\frac{ds(t)}{dt}$$

   Loi d'Ohm : $s(t)=Ri(t)$

   $$\frac{de(t)}{dt}=\frac{s(t)}{RC}+\frac{ds(t)}{dt}\Leftrightarrow\boxed{\tau\frac{ds(t)}{dt}+s(t)=\tau\frac{de(t)}{dt}\text{ avec }\tau=RC}$$
5. On branche à l'entrée de l'oscilloscope un GBF délivrant une tension $e(t)$ en créneaux, de fréquence suffisamment faible pour que le régime permanent soit atteint. Sur chaque demi-période, $s(t)$ vérifie l'équation différentielle : $\tau\frac{ds(t)}{dt}+s(t)=0$ dont la solution est $s(t)=Ae^{-\frac{t}{\tau}}$. Il faut que la période de $e(t)$ vérifie $T_e \gg \tau$.

   On mesure la constante de temps τ avec la tangente à l'origine ou lorsque $s(\tau)=0{,}37s(0^+)$.



On observe une discontinuité en $t=0$ car le filtre passe-haut laisse passer la discontinuité (variation brutale) de $e(t)$.

6.

Grandeurs temporelles	Grandeurs complexes
$e(t)=E_M\cos(\omega t)$	$\underline{e}(t)=E_M e^{j\omega t}=\underline{E}e^{j\omega t}$ avec $\underline{E}=E_M$
$s(t)=S_M\cos(\omega t+\varphi)$	$\underline{s}(t)=S_M e^{j(\omega t+\varphi)}=\underline{S}e^{j\omega t}$ avec $\underline{S}=S_M e^{j\varphi}$

Passage de l'équation différentielle en complexe :

$$\tau\frac{d\underline{s}(t)}{dt}+\underline{s}(t)=\tau\frac{d\underline{e}(t)}{dt}\Leftrightarrow(\tau j\omega+1)\underline{s}(t)=\tau j\omega\underline{e}(t)$$

$$\boxed{\underline{H}(j\omega)=\frac{\underline{s}(t)}{\underline{e}(t)}=\frac{j\tau\omega}{1+j\tau\omega}=\frac{j\dfrac{\omega}{\omega_C}}{1+j\dfrac{\omega}{\omega_C}}\text{ avec }\omega_C=\frac{1}{\tau}=\frac{1}{RC}}$$

7. Fréquence de coupure : $f_C=\dfrac{\omega_C}{2\pi}=\dfrac{1}{2\pi RC}$ soit $\boxed{C=\dfrac{1}{2\pi Rf_C}=20\text{ nF}}$.

8. Module ou gain : $G(\omega)=\dfrac{\dfrac{\omega}{\omega_C}}{\sqrt{1+\left(\dfrac{\omega}{\omega_C}\right)^2}}$ soit $G(f)=\dfrac{\dfrac{f}{f_C}}{\sqrt{1+\left(\dfrac{f}{f_C}\right)^2}}$

Argument ou phase : $\varphi(\omega)=\arg\left(j\dfrac{\omega}{\omega_C}\right)-\arg\left(1+j\dfrac{\omega}{\omega_C}\right)=\dfrac{\pi}{2}-\arctan\left(\dfrac{\omega}{\omega_C}\right)$ soit

$$\boxed{\varphi(f)=\frac{\pi}{2}-\arctan\left(\frac{f}{f_C}\right)}$$

9.

```
## Cellule 1 :Importation des bibliothèques utiles
import numpy as np                  # pour la manipulation des tableaux
import matplotlib.pyplot as plt     # pour les représentations graphiques
```

10.

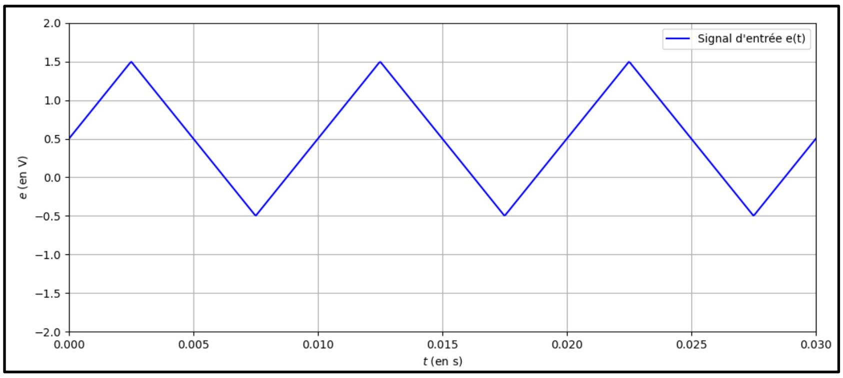
```
## Cellule 2: Données du problème physique
fe = 100                    # fréquence du signal e (en Hz)
E0 = 0.5                        # Valeur moyenne du signal e (en V)
A = 1                        # Amplitude du signal e (en V)
```

11.

```
## Cellule 3 : Synthèse spectrale du signal e
N = 50                      # Nombre d'harmoniques non nuls impairs pris en compte

def e(t):
    """ Fonction qui évalue le signal e à l'instant t """
    e = E0                          # prise en compte de la composante continue
    for n in range(0, N+1):         # prise en compte des harmoniques impairs pour n compris entre 0 et N
        Bn = 8*A* (-1)**n / np.pi**2 / (2*n+1)**2 # Amplitude de l'harmonique de rang impair 2n+1
        en = Bn * np.sin(2*np.pi*(2*n+1)*fe*t) # Expression de l'harmonique de rang impair 2n+1
        e = e + en                  # Expression de e par sommation des composantes
    return e
```

12.

```python
## Cellule 4: Représentation graphique du signal e
"""
t = np.linspace(tmin, tmax, N)
Renvoie un tableau de N points régulièrement espacés entre tmin (inclus) et tmax (inclus)
"""
t = np.linspace(0, 3/fe, 600)            # Tableau des valeurs de t représentées

plt.figure(figsize = (12,5))
plt.plot(t, e(t), 'b-', label="Signal d'entrée e(t)") # Tracé du signal e(t) en trait bleu
plt.xlim(0, 3/fe)
plt.ylim(-2,2)
plt.xlabel(r"$t$ (en s)")
plt.ylabel(r"$e$ (en V)")
plt.grid()
plt.legend(loc = 'upper right')
plt.show()
```



13.

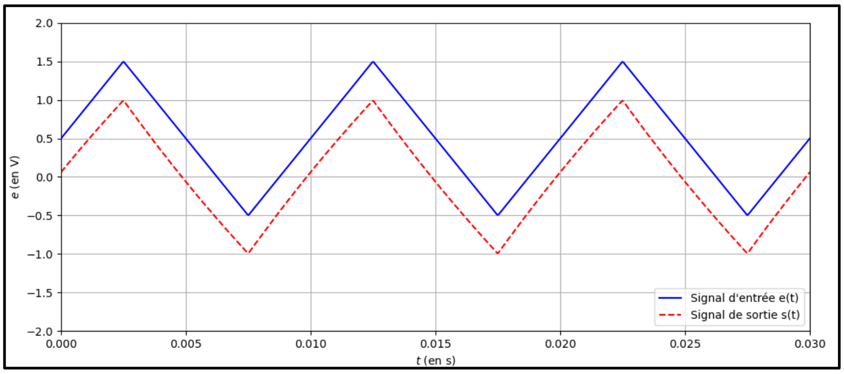
```python
## Cellule 5 :Définition du gain et du déphasage introduits par le filtre
def G(f, fc):
    """ Fonction qui renvoie le gain du filtre à la fréquence f (fc : fréquence de coupure du filtre) """
    return (f/fc)/np.sqrt(1+(f/fc)**2)

def phi(f, fc):
    """ Fonction qui renvoie la phase du filtre à la fréquence f (fc : fréquence de coupure du filtre) """
    return np.pi/2 - np.arctan(f/fc)
```

14.

```python
## Cellule 6 : Synthèse spectrale du signal de sortie du filtre en réponse au signal e
def s(t, fc):
    """ Fonction qui évalue le signal s, à l'instant t, en sortie du filtre passe-bas
    dont la fréquence de coupure est fc. """
    s = G(0,fc)*E0*np.sin(phi(0,fc))            # calcul de la composante continue de s(t)
    for n in range(0, N+1):            # calcul des composantes spectrales suivantes
        Bn = 8*A* (-1)**n / np.pi**2 / (2*n+1)**2 # amplitude de l'harmonique de rang 2n+1 pour e(t)
        Sn = G((2*n+1)*fe, fc)*Bn            # amplitude de l'harmonique de rang 2n+1 pour s(t)
        phin = phi((2*n+1)*fe, fc)           # phase à l'origine de l'harmonique de rang 2n+1 pour s(t)
        sn = Sn* np.sin(2*np.pi*(2*n+1)*fe*t + phin) # Expression de l'harmonique de rang impair 2n+1
        s = s + sn
    return s
```

15.

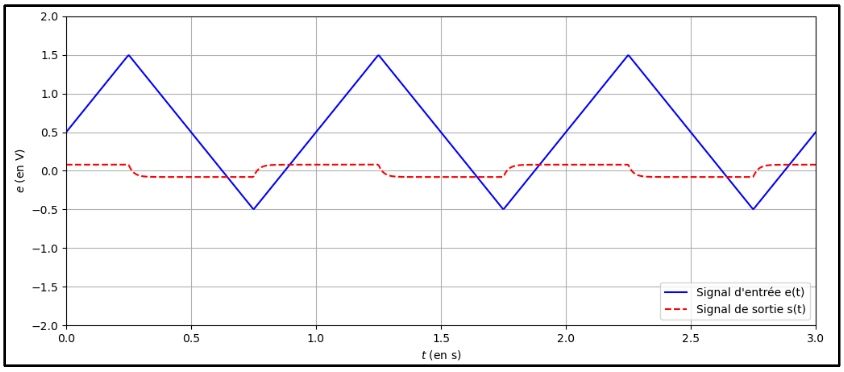
```python
## Cellule 7 : Représentation graphique du signal s
fc = 8                  # Fréquence de coupure du filtre

plt.figure(figsize = (12,5))
plt.plot(t, e(t), 'b-', label="Signal d'entrée e(t)") # Tracé du signal e(t) en trait bleu
plt.plot(t, s(t, fc), 'r--', label = "Signal de sortie s(t)") # Tracé du signal s(t) en pointillés (--)
rouges
plt.xlim(0, 3/fe)
plt.ylim(-2,2)
plt.xlabel(r"$t$ (en s)")
plt.ylabel(r"$e$ (en V)")
plt.grid()
plt.legend(loc = 'lower right')
plt.show()
```



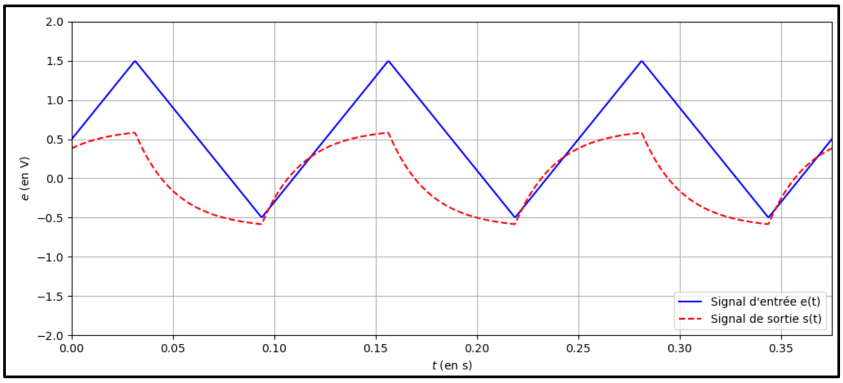
16. Le filtre passe-haut du <u>mode AC</u> <u>supprime la composante continue</u> de $e(t)$ L'allure de $s(t)$ est triangulaire : le filtre ne modifie pas les autres composantes de $e(t)$. Le mode AC permet de <u>ne visualiser que la composante alternative</u> de $e(t)$.
17. Pour $f_e = 1 \text{ Hz} << f_C$. Le signal $s(t)$ est de forme créneau. Pour chaque portion rectiligne de $e(t)$, s(t) est constante : le filtre se comporte comme un <u>dérivateur</u> dans le domaine temporel.

    <u>Justification</u> : $f << f_C$ ou $\omega << \omega_C$: $\underline{H}(j\omega) = \dfrac{\underline{s}(t)}{\underline{e}(t)} \simeq j\dfrac{\omega}{\omega_C} \Leftrightarrow \underline{s}(t) = j\dfrac{\omega}{\omega_C}\underline{e}(t)$

    Dans le domaine temporel : $\boxed{s(t) = \dfrac{1}{\omega_C}\dfrac{de(t)}{dt}}$



18. Pour $f_e = 8 \text{ Hz} = f_C$, l'allure du signal $s(t)$ n'est pas triangulaire : le signal est <u>déformé</u>. Le filtre atténue le fondamental (l'amplitude est multipliée par $\dfrac{1}{\sqrt{2}}$) mais laisse passer les harmoniques de rang supérieur.



19. À partir de fréquence $\boxed{f_e = 10 f_C}$, le mode AC de l'oscilloscope ne modifie pas la composante alternative du signal d'entrée $e(t)$: elle reste de forme triangulaire non déformée.

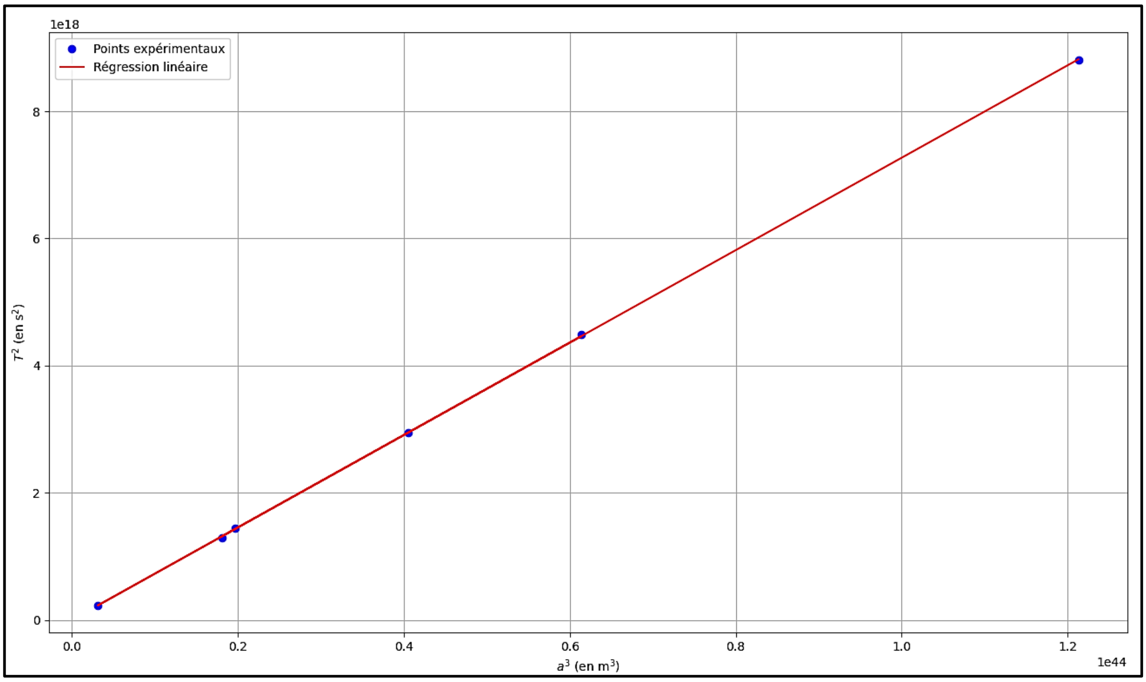
Exercice 2 – Vérification de la 3ème loi de Kepler

❖ **Simulation numérique en Python : Vérification de la 3ème loi de Kepler**

Astuce Python : Pour effectuer la même opération mathématique sur chaque élément d'un tableau, il suffit d'effectuer cette opération directement sur le tableau ! C'est le cas ici pour les conversions d'unités, et l'élévation au carré ou au cube…

Commentaire : on vérifie la loi de Kepler car la régression linéaire de T^2 en fonction de a^3 passe par les points expérimentaux

➢ Graphe



❖ **Simulation numérique en Python : Masse du trou noir**

Loi de Kepler : $\dfrac{T^2}{a^3} = \dfrac{4\pi^2}{GM}$ soit $\boxed{M = \dfrac{4\pi^2}{G}\dfrac{a^3}{T^2}}$

Masse du trou noir (obtenue par étude statistique) : $\boxed{M = \dfrac{4\pi^2}{G}\dfrac{1}{K}}$

Incertitude-type composée : $\dfrac{u(M)}{M} = \dfrac{u(K)}{K} \Leftrightarrow \boxed{u(M) = M\dfrac{u(K)}{K}}$

Valeurs numériques : $\boxed{M = 8{,}1581.10^{36}\ \text{kg}}$ et $\boxed{u(M) = 0{,}0307.10^{36}\ \text{kg}}$

➢ Écart normalisé : $\boxed{EN = \dfrac{|M - M_t|}{\sqrt{u^2(M) + u^2(M_t)}} = 2{,}3}$: de l'ordre de 2, donc l'estimation est correcte.

➢ Script Python :

```python
"""Exécuter les cellules les unes après les autres après les avoir complétées
en cliquant sur Run puis Execute Cell (ou CTRL + Entrée)
Pour exécuter l'ensemble du fichier : CTRL + E
"""
# 3ème loi de Kepler

## Cellule 1 :Importation des bibliothèques utiles
import numpy as np                  # pour la manipulation des tableaux
import matplotlib.pyplot as plt     # pour les représentations graphiques

## Cellule 2: Vérification de la 3ème loi de Kepler
tabT = np.array([94.1,15.24,67.2,54.4,36.,38])*365*24*3600   # Tableau des périodes T en secondes!
taba = np.array([3300.,980.,2630.,2290.,1750.,1800.])*1.5*10**11 # Tableau des demi-grand axes a en mètres!
x = taba**3                 # Tableau des abscisses a^3
y = tabT**2                 # Tableau des ordonnées T^2
"""
p = np.polyfit(x, y, n)
Modélise la courbe y = f(x) par un polynôme de degré n
Arguments:
    x : tableau des abscisses
    y : tableau des ordonnées
    n : degré du polynôme (pour n = 1 : régression linéaire)
Renvoie:
    p : tableau des coefficients du polynôme tel que :
    p[0] : coefficient de degré n, p[1] : coefficient de degré n-1...
    p[n] : coefficient de degré 0

"""
p = np.polyfit(x, y, 1)  # Tableau des coefficients de la régression linéaire de y en fonction de x
ymodele = p[0]*x + p[1]  # Equation de la régression linéaire ymodele en fonction de x

plt.figure(figsize = (16,9))
plt.plot(x, y, 'bo', label="Points expérimentaux") # Tracé des points expérimentaux avec des ronds bleus
plt.plot(x, ymodele, 'r-', label="Régression linéaire") # Tracé de la régression linéaire en trait rouge
plt.xlabel(r"$a^3$ (en m$^3$)")
plt.ylabel(r"$T^2$ (en s$^2$)")
plt.grid()
plt.legend(loc = 'upper left')
plt.show()

## Cellule 3 : Masse du trou noir
tabK = tabT**2 / taba**3      # Tableau des valeurs T^2 / a^3
K = np.mean(tabK)             #Valeur moyenne
sK = np.std(tabK,ddof=1)      #Ecart-type expérimental
uK = sK / np.sqrt(len(tabK))  #Incertitude-type

G = 6.67*1e-11  # Constante de gravitation
M = 4 * np.pi**2 / (G*K)   # Expression de la masse M
uM = M * uK / K            # Expression de l'incertitude-type uM
print(f"Masse du trou noir : M = {M:.4e}  kg") #Affichage de la masse du trou noir
print(f"Incertitude-type : u(M) = {uM:.2e} kg ") #Affichage de l'incertitude-type

## Cellule 4 : Ecart normalisé
Mt = 8.2541*1e36            #Valeur tabulée de M
uMt = 0.0278*1e36           #Valeur tabulée de uM
EN = abs(M - Mt) / np.sqrt(uM**2 + uMt**2) #Calcul de l'écart normalisé
print(f"Ecart normalisé : EN = {EN:.1f}") #Affichage de l'écart normalisé
```

➢ Affichage des résultats

```
Masse du trou noir : M = 8.1581e+36  kg
Incertitude-type : u(M) = 3.07e+34 kg
Ecart normalisé : EN = 2.3
```