PROGRAMMER EN PYTHON
PROJET N°3 : ADAPTATION D'UNE CHARGE À UNE SOURCE RÉELLE

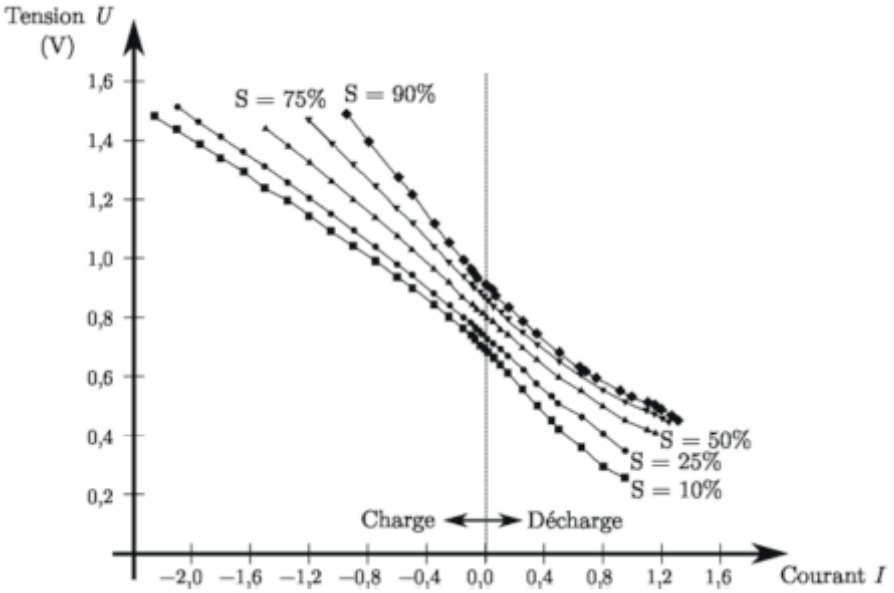
Les problématiques d'adaptation d'une charge à une source électrique sont très présentes dans le monde technologique. Ainsi, l'adaptation du réseau électrique français au parc de centrales françaises permet-il d'éviter certains gaspillages énergétiques. L'activité suivante cherche à illustrer cette notion d'adaptation d'une charge à une source donnée.

Dans un premier temps, le tracé expérimental de la caractéristique tension = f(courant) d'une source réelle est réalisé ce qui permet de proposer une modélisation de la source réelle par l'association en série d'une source idéale de tension de f.é.m E_0 et d'une résistance de sortie R_s (les valeurs de ces paramètres sont obtenues par une régression linéaire).

La «charge» branchée en sortie de cette source réelle est modélisée par une résistance r variable. Enfin, à l'aide d'une résolution numérique, sont déterminées l'intensité du courant if débité par la source et la tension uf aux bornes de la charge pour diverses valeurs de r (il s'agit en fait de la détermination du point de fonctionnement du circuit). Ces deux valeurs permettent de calculer la puissance reçue par la charge pour les différentes valeurs de r. La courbe de la puissance reçue en fonction de la valeur de la résistance présente un maximum pour une certaine valeur de la résistance r, ce qui illustre la notion d'adaptation d'une charge à une source donnée.

Tracé de la caractéristique de la source réelle

La source réelle étudiée présente les caractéristiques suivantes, dans divers états :

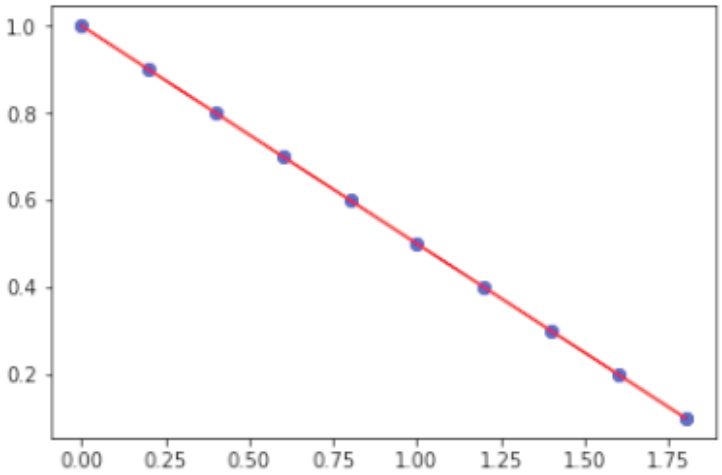


Un pointage des valeurs de U et I pour l'état S = 90 % et leur saisie dans deux tableaux permettent de tracer la caractéristique avec le script suivant :

```
from math import *
```

```
import numpy as np
from scipy import *
import matplotlib.pyplot as plt
from scipy.optimize import bisect
```

```
i=np.array([0,0.2,0.4,0.6,0.8,1,1.2,1.4,1.6,1.8])
u=np.array([1,0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1])
plt.plot(i,u,'bo')
plt.plot(i,u,'r')
```



Modélisation de cette caractéristique par une association classique

La source réelle est alors modélisée par l'association en série d'une source idéale de tension de f.é.m E_0 et d'une résistance de sortie R_s, dont les valeurs sont déterminées à l'aide d'une régression linéaire. Le script est le suivant :

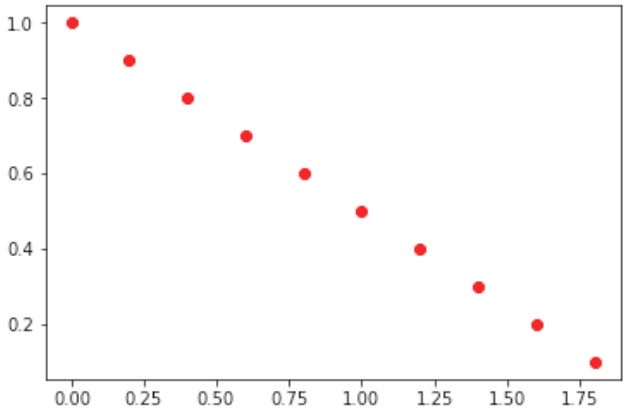
```
rl=np.polyfit(i,u,1)
print(rl)
def urmod(i):
    return rl[1]+rl[0]*i

plt.plot(i,urmod(i),'ro')
```

La première ligne renvoie une modélisation affine des points u en fonction de i (le polynôme est de degré 1 comme l'indique l'argument 1). La deuxième permet de visualiser le résultat de la modélisation. Les lignes suivantes permettent de définir une fonction modèle urmod=f(i) à partir des résultats donnés par polyfit (cela est rendu nécessaire par le fait que polyfit renvoie une liste de nombres et non une fonction).

Le résultat renvoyé par polyfit est le suivant :

```
[-0.5  1. ] .
```

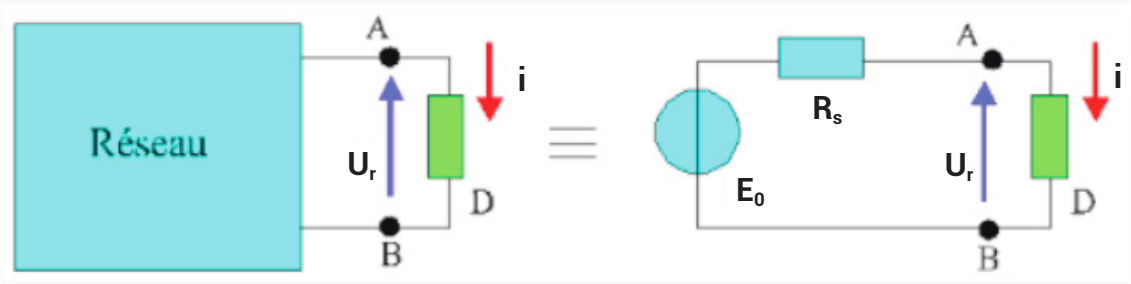
Ce résultat indique une fém à vide E_0 = 1V et une résistance de sortie de R_s = 0,5 Ω, qui sont des ordres de grandeurs cohérents pour des batteries portables.



Détermination du point de fonctionnement du circuit pour diverses valeurs de résistance de charge

La charge branchée en sortie de la source réelle étant modélisée par une résistance r variable, l'intensité du courant i_f débité par la source et la tension u_f aux bornes de la charge pour diverses valeurs de r sont déterminées, d'abord graphiquement, puis numériquement.

Le circuit équivalent représentant la situation est le suivant :



Dans un premier temps, une liste de valeurs pertinentes de la résistance de charge r est créée (typiquement avec des valeurs encadrant celle de la résistance de sortie R_s).

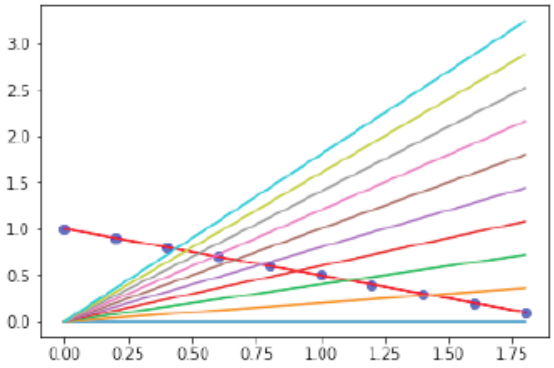
Puis, une liste de valeurs d'intensité est créée pour servir d'abscisse au futur graphique.

Enfin, une boucle for permet de tracer sur le même graphe les caractéristiques de la charge et de la source. Pour cela, il faut définir la fonction ur = r×i et faire calculer ses valeurs pour chaque valeur de r et de i.

On doit donc définir la fonction ur = r×i et la faire calculer et tracer pour diverses valeurs de r et pour toutes les valeurs de i du tableau.

Cela donne :

```
rliste=np.arange(10)/5
plt.plot(i,u,'bo')
plt.plot(i,u,'r')
for r in rliste:
    ur=r*i
    plt.plot(i,ur)
```



Diverses intersections apparaissent : chaque valeur de r engendre un point de fonctionnement spécifique.

Proposition

Dans un premier temps, il est possible de déterminer graphiquement les diverses valeurs des grandeurs de fonctionnement, i_f et u_f, aux intersections des courbes. Ceci peut s'avérer fastidieux et peu précis au vu des graduations fournies.

Pour déterminer numériquement les grandeurs de fonctionnement, la fonction bisect de scipy peut être utilisée. Celle-ci renvoie la valeur de la racine d'une fonction f(x), c'est-à-dire la solution de l'équation $f(x) = 0$.

L'intensité de fonctionnement i_f vérifiant $u(i_f) = u_r(i_f)$, il est nécessaire de définir une fonction f telle que sa racine corresponde au point de fonctionnement, c'est-à-dire $f(i_f) = u_r(i_f) - u(i_f) = 0$.

La détermination automatisée des différentes racines s'effectue au moyen d'une boucle for. Les résultats sont stockés dans une liste, complétée au fur et à mesure par la commande append.

```
def nul(i):
    return rl[1]+rl[0]*i-r*i

lpfi=[]
for r in rliste:
    pfi=bisect(nul,0,2)
    lpfi.append(pfi)
print(lpfi)
```

qui renvoie :

```
[1.999999999998181, 1.428571428570649, 1.1111111111113132, 0.9090909090900823, 0.7692307692323084
, 0.6666666666660603, 0.5882352941171121, 0.5263157894751203, 0.4761904761890037, 0.4347826086941
4953]
```

Détermination de la puissance fournie à la charge et adaptation

La puissance fournie à la charge est enfin calculée pour les différentes valeurs de r.

```
lp=[]
for r in rliste:
    pfi=bisect(nul,0,2)
    pui=r*pfi**2
    lp.append(pui)
print(lp)
```

qui renvoie :

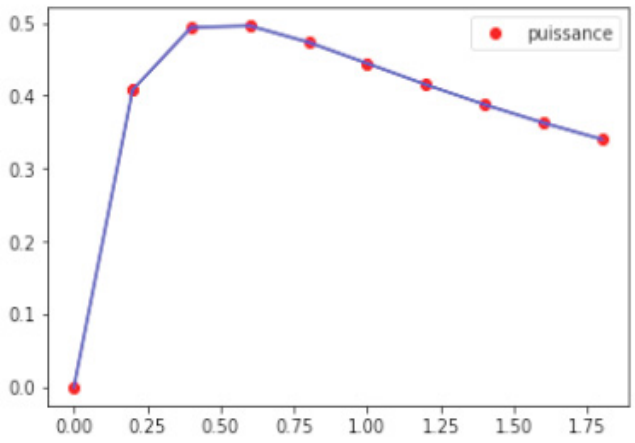
```
[0.0, 0.408163265305677, 0.4938271604940068, 0.4958677685941393, 0.4733727810669831, 0.4444444444
43636, 0.4152249134940544, 0.38781163435114674, 0.36281179138097613, 0.3402646502812019]
```

Analyse et optimisation

La liste des puissances fournies fait apparaître un maximum pour la quatrième valeur. Le tracé de la courbe de la puissance fournie en fonction de r permet de visualiser le maximum.

```
plt.plot(rliste,lp,'ro',label='puissance')
plt.plot(rliste,lp,'b')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1a13a61be0>
```



La puissance fournie est maximale pour une valeur de résistance de charge de l'ordre de $r_0 = 0{,}5\ \Omega$.

La théorie confirme ce résultat obtenu par une approche numérique. En effet, l'expression littérale de la puissance reçue est $P_r = r.\left(\frac{E0}{Rs+r}\right)^2$. Cette fonction présente un maximum pour une résistance du circuit de charge égale à la résistance de sortie de la source réelle.