

TP Scilab : analyse de données expérimentales

Partie 1 – Régression linéaire

- Objectif

On cherche à déterminer la résistance R d'une résistance électrique.

Expérimentalement, pour différentes tensions U , on mesure l'intensité du courant électrique I consommé par la résistance.

A partir des points de mesures et de la loi d'Ohm ($U = R.I$), nous allons utiliser le modèle de *régression linéaire* pour estimer la résistance R .

1.1. Importation des données dans Scilab

Les mesures ont été stockées dans le fichier *data_resistance2.csv* (c'est un fichier texte au format CSV).

Copier le fichier sur le bureau.

Avec un éditeur de texte comme *Notepad++* ouvrir le fichier *data_resistance2.csv* pour en connaître la structure.

*Remarque : pendant la phase de mesures, on peut construire le fichier data_resistance2.csv avec Notepad++.
Autre solution avec un tableur : saisie des données puis exportation au format CSV.*

Dans Scilab :

Navigateur de fichiers : déplacez-vous jusqu'au bureau.

La fonction *Scilab* `csvRead()` permet d'enregistrer les données du fichier dans un tableau :

```
--> mesures = csvRead("data_resistance2.csv")
```

Résultat attendu :

```
--> mesures
mesures =
  0.    0.
  9.    0.5
 18.9   2.
 32.5   3.5
 37.9   4.2
 49.6   4.5
 56.4   5.5
 68.3   7.
 79.    7.9
 93.5   9.6
 97.7  10.2
```

La première colonne correspond au courant en mA, la seconde à la tension en V.

A partir du tableau `mesures`, créer le vecteur colonne x (courant) et le vecteur colonne y (tension).

Résultat attendu :

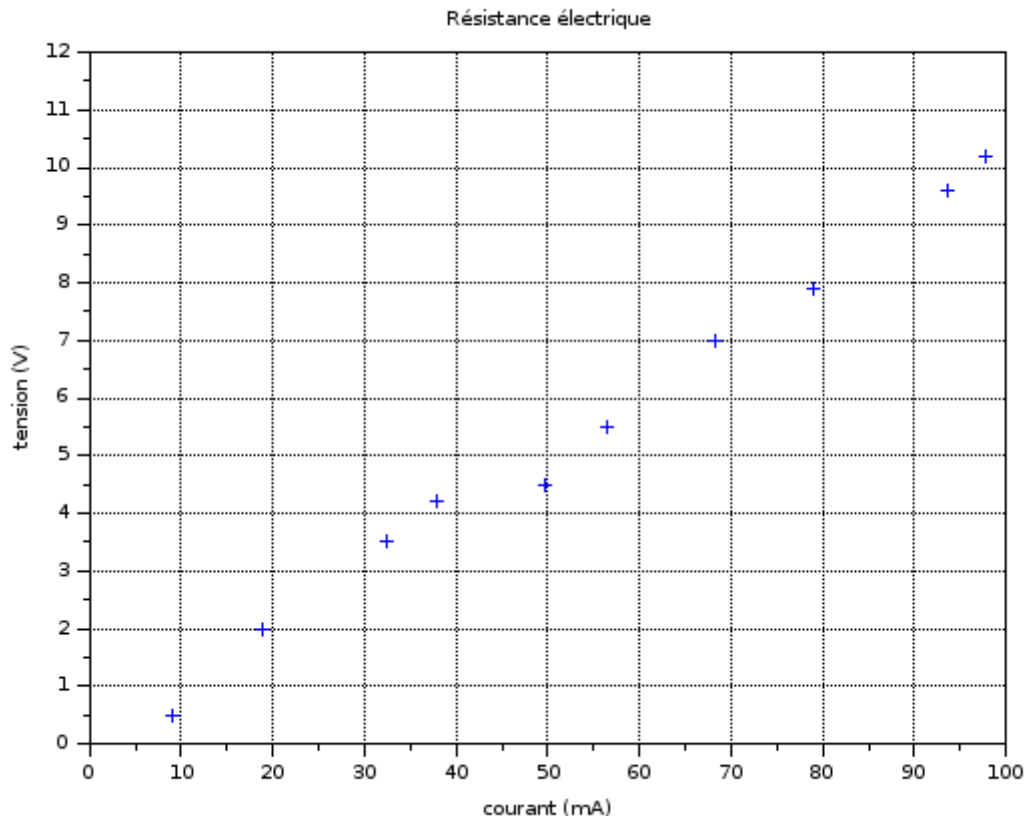
```
--> x
x =
  0.
  9.
  [...]
 97.7
```

```
--> y
y =
    0.
    0.5
    [...]
    10.2
```

1.2. Courbe U(I)

Avec la fonction *Scilab* `plot()`, tracer la courbe des données expérimentales U(I).

Résultat attendu :



1.3. Régression linéaire

1.3.1. Un peu de théorie

Le modèle de régression linéaire peut être estimé par la méthode des « moindres carrés ». Dans le cadre d'un modèle linéaire simple (comme la loi d'Ohm), on peut représenter graphiquement la relation entre x et y à travers un nuage de points. L'estimation du modèle linéaire permet de tracer la droite de régression.

La **méthode des moindres carrés** permet de comparer des données expérimentales (entachées d'erreurs de mesure), à un modèle mathématique censé décrire ces données (ici la loi d'Ohm).

On note $(M_i(x_i, y_i))$ le nuage de points que l'on cherche à ajuster par une droite $(d) : y = ax + b$.

La méthode de régression linéaire consiste à minimiser les *résidus* $y_i - ax_i - b$, c'est-à-dire la distance de chaque point M_i à la droite (d) dans la direction de l'axe des ordonnées (Cf. figure 1).

Pour cela, on cherche à minimiser la somme des carrés des résidus, définie par :

$$S = \sum_{i=1}^n (y_i - a x_i - b)^2$$

On admet que S est minimale pour :

$$a = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad \text{et :} \quad b = \bar{y} - a \bar{x}$$

où \bar{x} est la moyenne des abscisses et \bar{y} la moyenne des ordonnées :

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \quad [\text{source Wikipedia}]$$

1.3.2. Calcul des coefficients de la droite de régression

Avec des boucles for, écrire les fonctions *Scilab* :

```
m = moyenne(vecteur)
[a, b] = coefficient_droite_regression(vecteur_x, vecteur_y)
```

Résultats attendus :

```
--> mx = moyenne(x)
mx =
    49.345455
--> my = moyenne(y)
my =
    4.9909091
--> [a, b] = coefficient_droite_regression(x, y)
a =
    0.1034455
b =
   -0.1136552
```

1.3.3. Droite de régression

Avec une seule ligne de code, en manipulant directement des vecteurs, tracer la droite de régression.
`plot(x, ...)`

Résultat attendu :

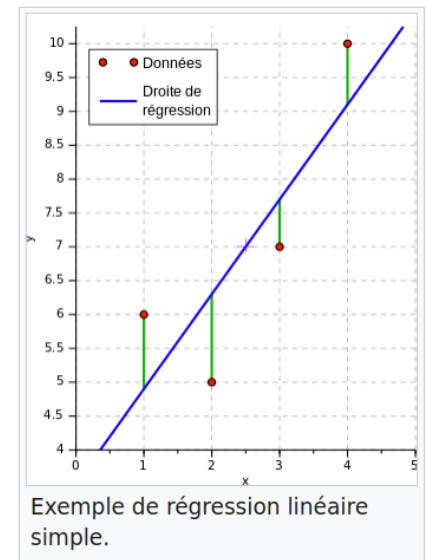
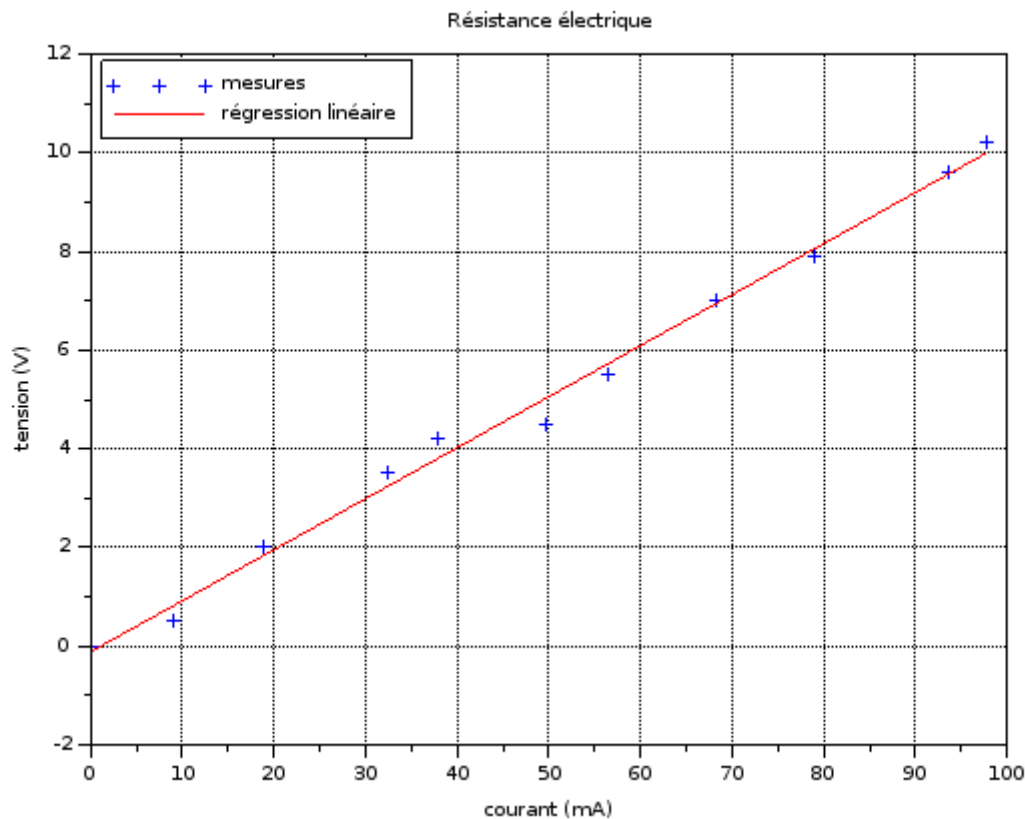


Figure 1



1.3.4. Exploitation

Proposer une valeur pour la résistance R.

1.3.5. Avec un code plus efficace

A l'aide des fonctions natives *Scilab* :

- `sum()` — somme des éléments d'un vecteur, d'un tableau
- `mean()` — moyenne des éléments d'un vecteur, d'un tableau

réécrire la fonction `coefficient_droite_regression()` sans boucle `for` (manipulation directes des vecteurs).

1.3.6. La fonction `reglin()`

Évidemment, *Scilab* propose une fonction native pour calculer les coefficients de la droite de régression.

A tester :

```
[a, b] = reglin(x',y')
```

1.4. Coefficient de corrélation linéaire

Par définition :

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

Plus le coefficient de corrélation r est proche de 1 ou -1 , et plus la somme des carrés des résidus est voisine de 0.

Le coefficient de corrélation est donc un bon indicateur de la validité de l'ajustement affine.

Écrire la fonction scilab :

```
r = coefficient_correlation(vecteur_x, vecteur_y)
```

Résultat attendu :

```
--> r=coefficient_correlation(x, y)
r =
    0.9968453
```

Commentaire ?

Remarque : la fonction native *Scilab* `correl()` donne directement le coefficient de corrélation :

```
--> r=correl(x,y)
r =
    0.9968453
```

Partie 2 – Détermination de la constante de temps d'un système du 1^{er} ordre

- Objectif

On s'intéresse à la phase de préchauffage d'un four, initialement à 40 °C, avec une consigne à 240 °C.

Le four se comporte comme un système du 1^{er} ordre.

On cherche à mesurer la constante de temps τ .

2.1. Importation des données dans Scilab

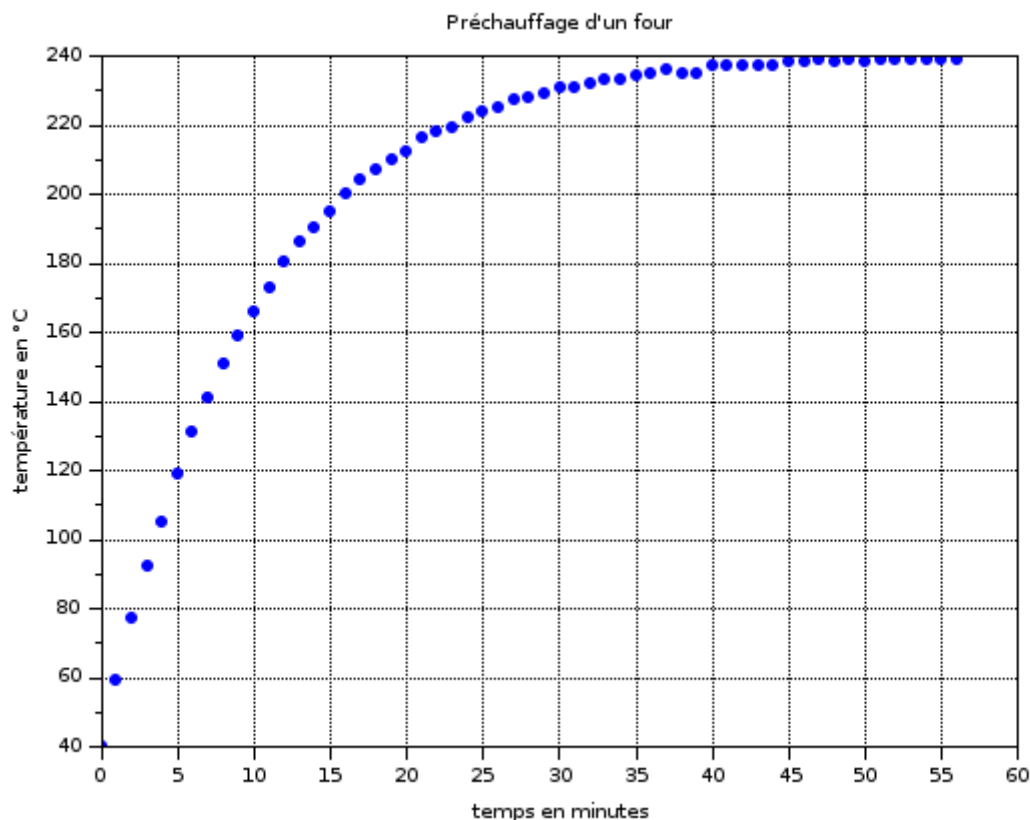
Les mesures ont été stockées dans le fichier *mesures_four.csv*

Importer les mesures dans un tableau *mesures*.

Créer le vecteur colonne *t* (temps en minutes) et le vecteur colonne *T* (température en °C).

Tracer la courbe des données expérimentales *T(t)*.

Résultat attendu :



A partir de vos connaissances, mesurer graphiquement la constante de temps.

2.2. Droite de régression

La réponse en température du four (considéré comme un système du 1^{er} ordre) est donnée par la relation :

$$T(t) = T_f + (T_0 - T_f)e^{-t/\tau}$$

avec

- t le temps (en minutes)
- T la température (en °C)
- T_0 la température initiale (40 °C à $t=0$)
- T_f la température finale (240 °C)
- τ la constante de temps (en minutes)

La relation ci-dessus n'est pas linéaire.

Mais par un changement de variable judicieux, on peut se ramener à une relation linéaire :

$$T(t) = T_f + (T_0 - T_f)e^{-t/\tau}$$

$$e^{-t/\tau} = \frac{T(t) - T_f}{T_0 - T_f}$$

$$-t/\tau = \ln\left(\frac{T(t) - T_f}{T_0 - T_f}\right)$$

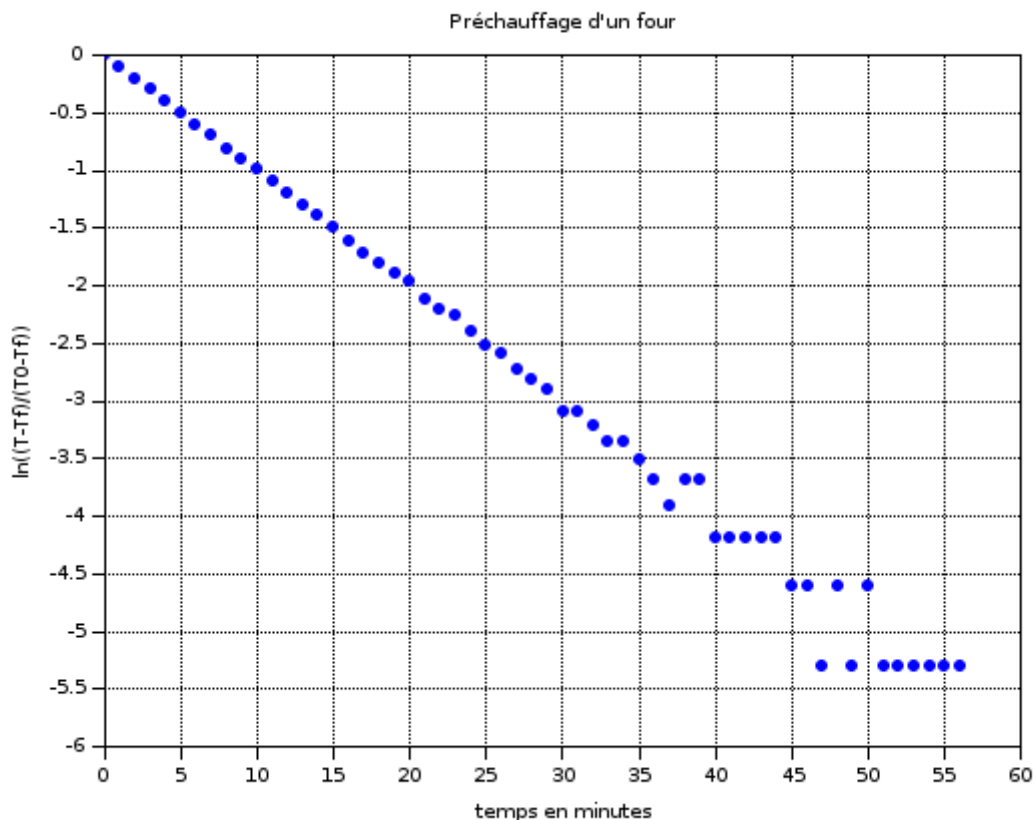
On posant : $Y(t) = \ln\left(\frac{T(t) - T_f}{T_0 - T_f}\right)$ on se ramène à une droite : $Y(t) = -t/\tau$

Construire le vecteur Y à partir du vecteur T .

Remarque : La fonction *Scilab* `log()` donne le logarithme népérien.

Tracer la courbe $Y(t)$.

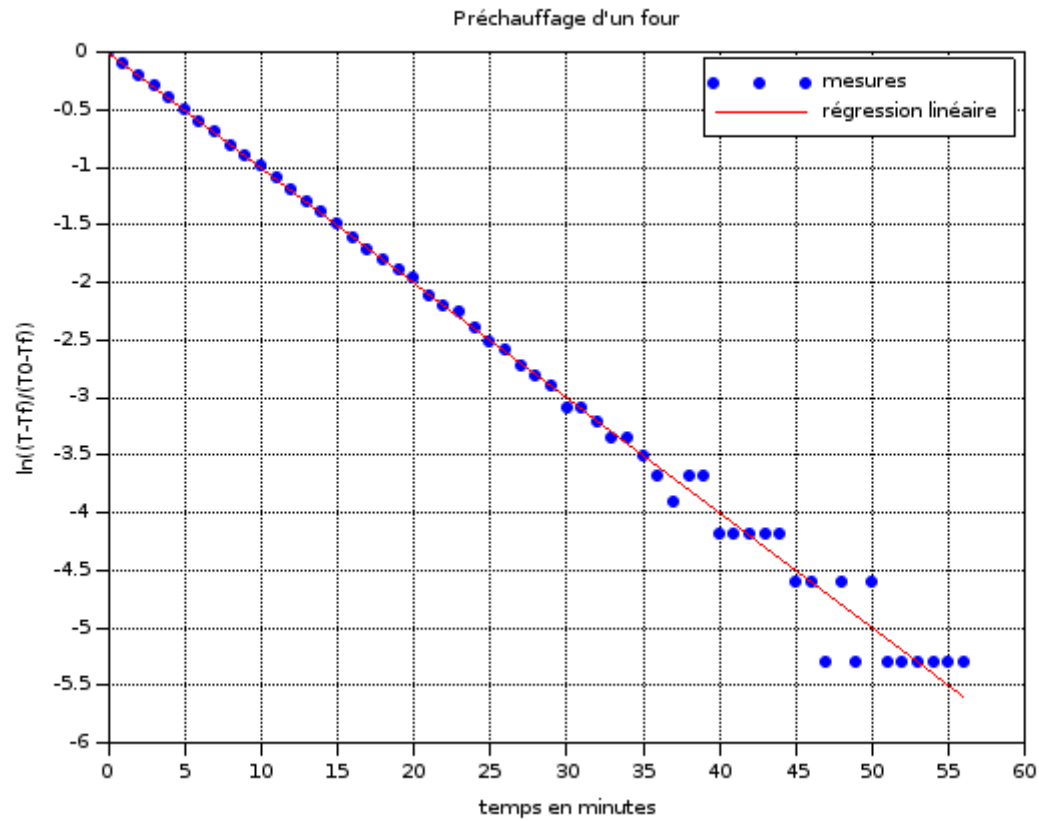
Résultat attendu :



On peut maintenant faire une régression linéaire (Cf. partie 1) :
calculer les coefficients de la droite de régression.

Tracer la droite de régression.

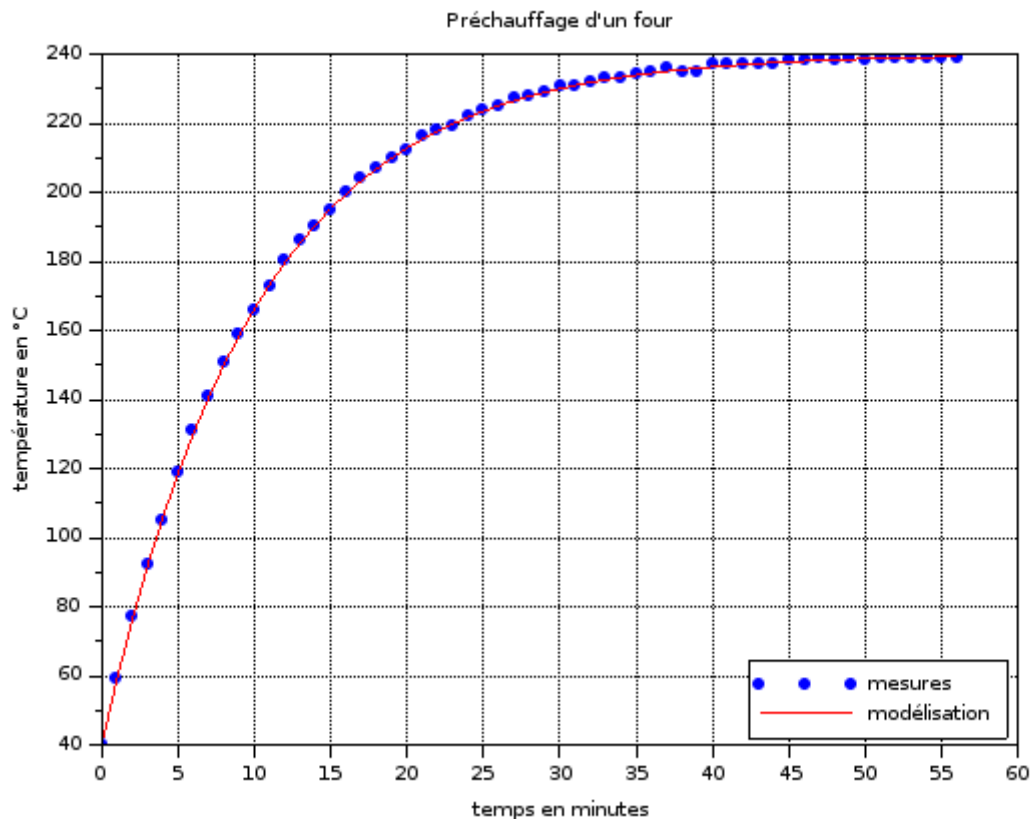
Résultat attendu :



En déduire la constante de temps τ .

Comparer la courbe expérimental $T(t)$ et la courbe du modèle théorique $T(t)=T_f+(T_0-T_f)e^{-t/\tau}$
obtenue avec $\tau = 10,02$ minutes.

Résultat attendu :



2.3. Autre technique plus générale

Il n'est pas toujours possible de se ramener à une relation linéaire.

On peut alors utiliser une méthode générale qui consiste à minimiser l'erreur :

$$e = \sum_{i=1}^n (T_{i,\text{exp}} - T_{i,\text{theo}})^2$$

Écrire une fonction $e = \text{erreur}(\tau)$ prenant pour argument un nombre τ (constante de temps)

et renvoyant la quantité $e = \sum_{i=1}^n (T_{i,\text{exp}} - T_{i,\text{theo}})^2$ donnant la somme du carré des écarts entre les températures expérimentales $T_{i,\text{exp}}$ et les températures $T_{i,\text{theo}}$ vérifiant la loi recherchée.

Dans la suite, on cherche à déterminer précisément la valeur du paramètre τ qui permet de minimiser e .

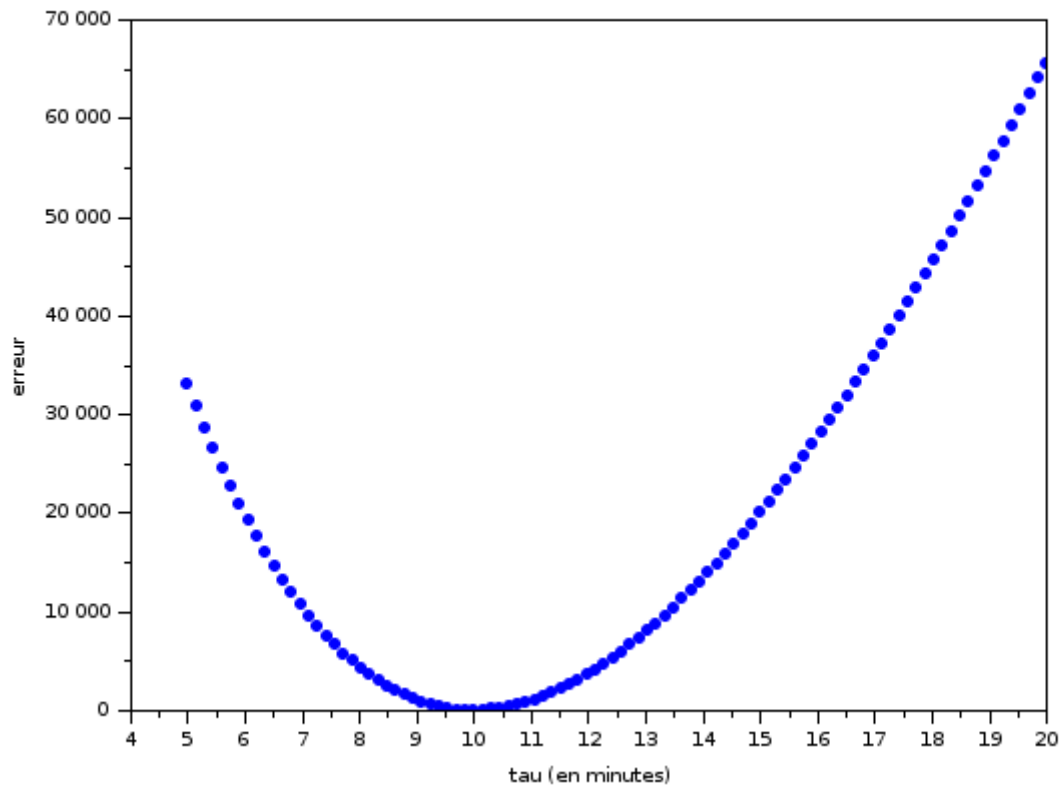
Dans notre cas, on estime manuellement que $(5 \leq \tau \leq 20) \text{ minutes}$.

Proposer un algorithme utilisant une boucle `for` et permettant d'obtenir un tableau, noté `tab_e`, contenant les valeurs de e calculées pour 100 valeurs de τ comprises dans l'intervalle 5 et 20.

On pourra utiliser `linspace(x1, x2, n)` qui renvoie un vecteur de n éléments régulièrement espacés entre $x1$ et $x2$.

Tracer la courbe `erreur(tau)`.

Résultat attendu :



En déduire la constante de temps.

- Complément : fonction native *Scilab* `fminsearch()` de recherche du minimum d'une fonction à une variable

A tester :

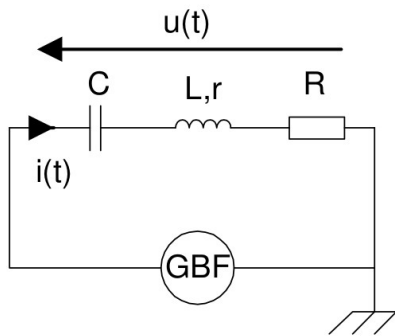
```
--> [tau, erreurmin] = fminsearch(erreur,x0=5)
tau =
    9.9841309
erreurmin =
    13.103485
```

Partie 3 – Système à deux grandeurs

Dans les parties 1 et 2, nous avons une seule grandeur à déterminer.
Nous allons voir ici un exemple avec deux grandeurs inconnues.

- Objectif

On s'intéresse à un circuit électrique RLC (résistance / bobine / condensateur) alimenté par une tension sinusoïdale de fréquence réglable :



La source de tension u (GBF) est réglée à $5 V_{\text{eff}}$

$R = 100 \Omega$; $C = 1 \mu\text{F}$

On cherche le modèle série (L , r) de la bobine (L est l'inductance en henry, et r la résistance interne).

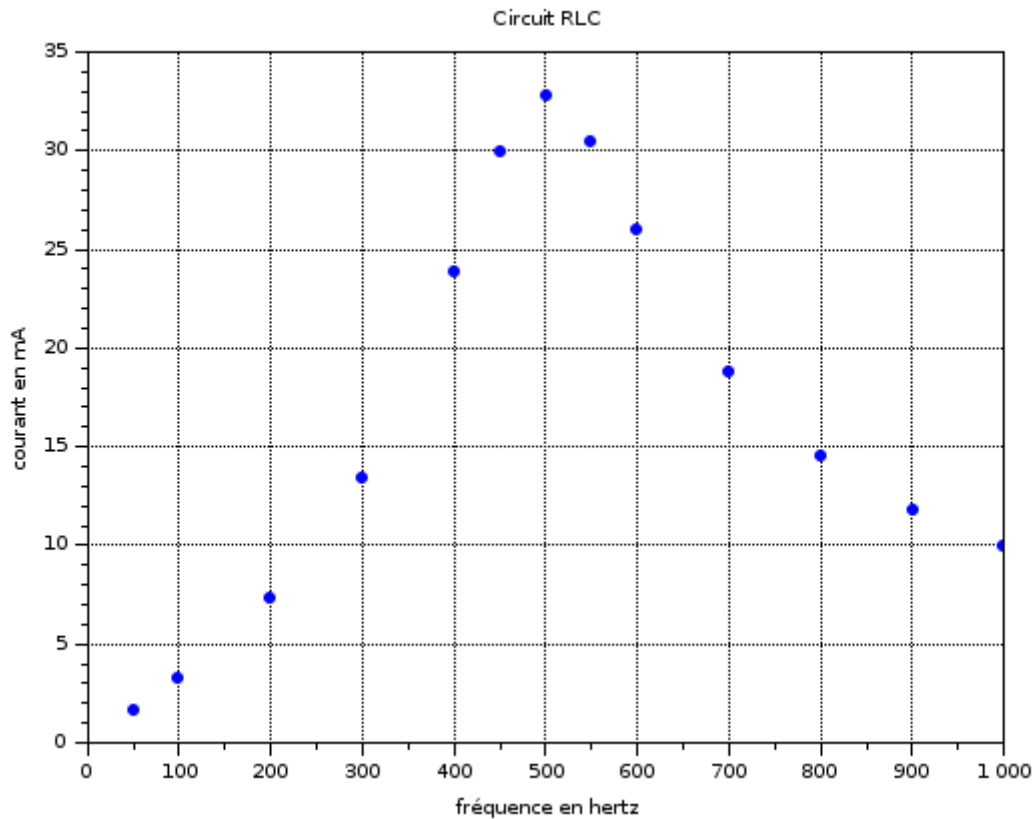
Pour cela, on mesure le courant efficace pour différentes fréquences.

3.1. Importation des données dans Scilab

Les mesures ont été stockées dans le fichier *mesures_RLC.csv*
Importer les mesures dans un tableau *mesures*.

Créer le vecteur colonne f (fréquences en hertz) et le vecteur colonne I (intensité du courant efficace en mA).

Tracer la courbe des données expérimentales $I(f)$.
Résultat attendu :



3.2. Modèle théorique

Les lois de l'électricité permettent d'écrire que :

$$I(f) = \frac{U}{\sqrt{(R+r)^2 + \left(2\pi Lf - \frac{1}{2\pi Cf}\right)^2}}$$

Dans notre expérimentation, U, R et C sont des constantes connues.

Pour une fréquence donnée, le courant efficace I est donc une fonction des deux variables L et r.

3.3. Erreur entre modèle théorique et données expérimentales

3.3.1. Fonction d'erreur

On reprend la définition vue dans la partie 2.3 :

$$e = \sum_{i=1}^n (I_{i,\text{exp}} - I_{i,\text{theo}})^2$$

Compléter le code :

```
U = 5000 // mV
R = 100 // ohms
C = 1e-6 // farad
function e=erreur(L, r)
    // vecteur I théorique
    Itheo = [ . . . ]
    e = sum( [ . . . ] )
endfunction
```

Résultat attendu :

```
--> erreur(L=0.05, r=100)
ans =
    852.56187
--> erreur(L=0.1, r=50)
ans =
```

3.3.2. Tracé de la courbe en 3 dimensions *erreur(L, r)*

Le fabricant de la bobine donne les grandeurs nominales suivantes : 0,1 H ; 50 Ω

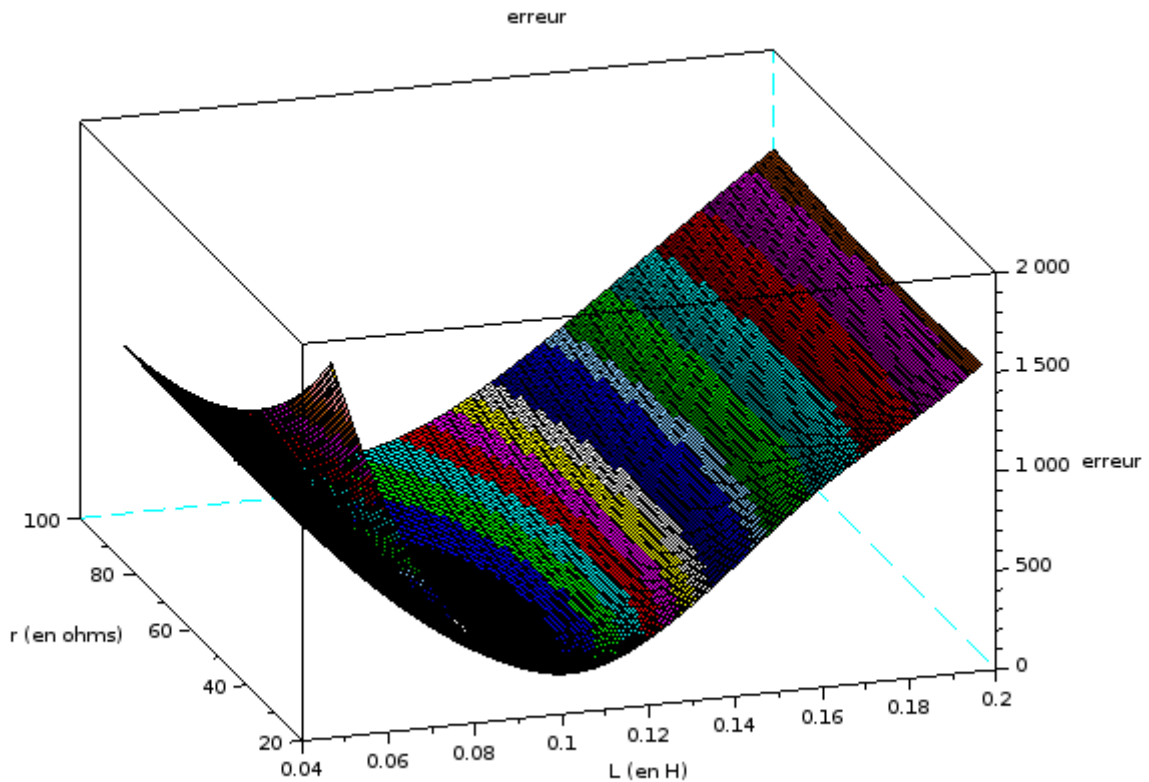
Nous allons évaluer la fonction *erreur(L, r)* dans les intervalles :

- $L = 0,05$ à $0,2$ H (sur 100 points)
- $r = 25$ à 100 Ω (sur 100 points)

Compléter le code :

```
tab_L = linspace( . . . ) // axe x 100 points
tab_r = linspace( . . . ) // axe y 100 points
tab_e = feval(tab_L,tab_r,erreur)' // tableau 100x100 = 10000 points
scf()
surf(tab_L,tab_r,tab_e)
xlabel("L (en H)")
ylabel("r (en ohms)")
zlabel("erreur")
title("erreur")
```

Résultat attendu (rotation avec clic droit) :



Évaluer les valeurs de r et L qui minimisent l'erreur.

3.3.3. Tracé des courbes de niveaux *erreur(L, r)*

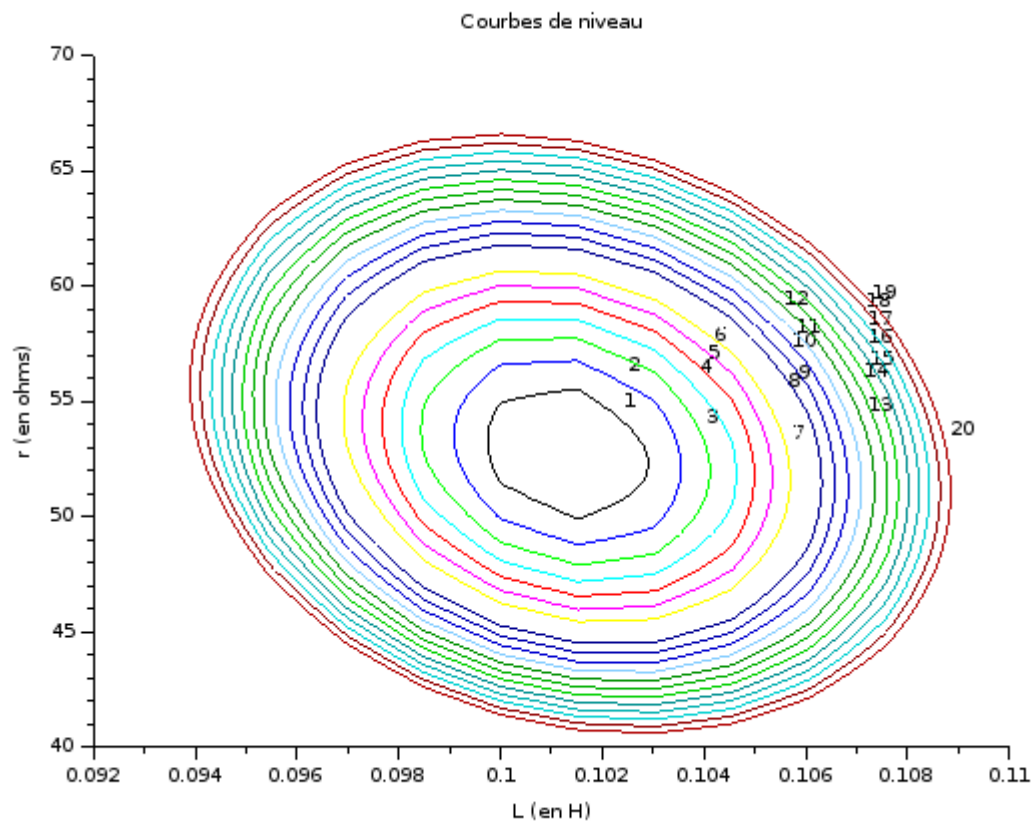
La représentation en courbes de niveaux est plus adaptée à la recherche du minimum.

Tester le code :

```
scf()
contour(tab_L,tab_r,tab_e',0:1:20)
xlabel("L (en H)")
```

```
ylabel("r (en ohms)")
title("Courbes de niveau")
```

Résultat attendu :

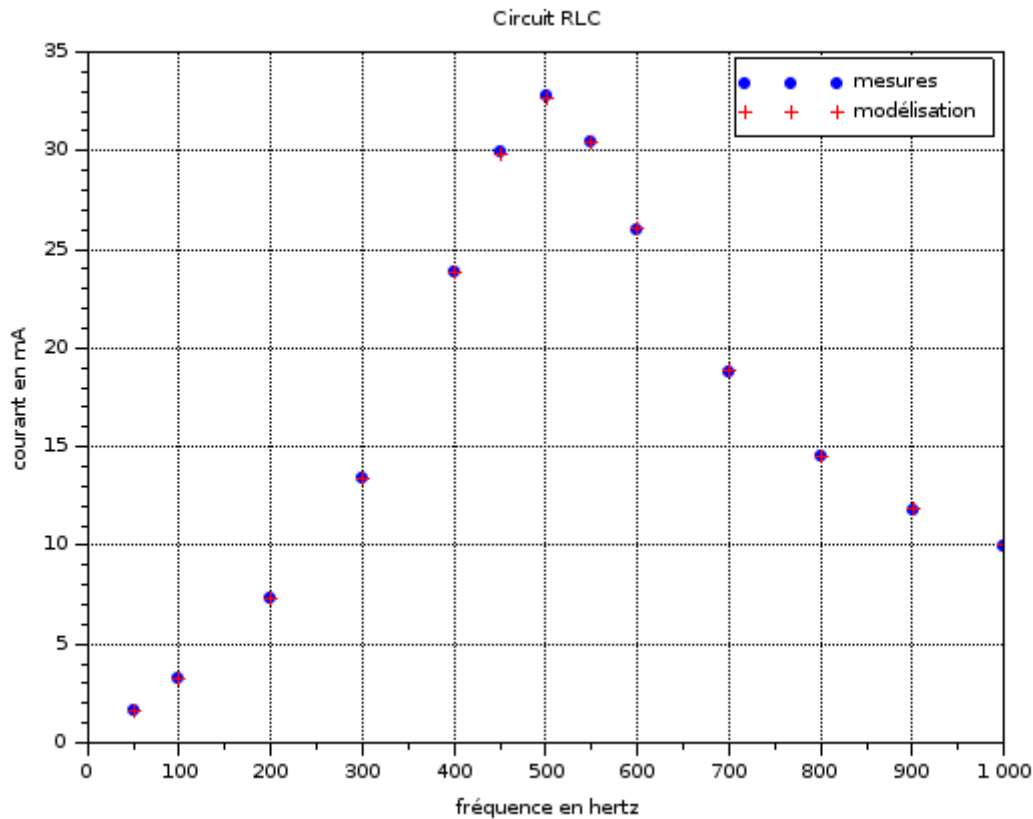


Évaluer les valeurs de r et L qui minimisent l'erreur.

3.3.4.

Comparer la courbe expérimentale $I(f)$ et la courbe du modèle théorique pour les valeurs (L, r) obtenues ci-dessus.

Résultat attendu (avec 0,101 H et 53 Ω) :



3.3.5. Complément : fonction native *Scilab* `fminsearch()` de recherche du minimum d'une fonction à deux variables

A tester :

```
// redéfinition de la fonction erreur() avec un vecteur [L,r] en argument
function e=erreur2(Lr)
    L = Lr(1)
    r = Lr(2)
    // vecteur
    Itheo = U./sqrt((R+r)^2+(2*pi*f*L-1./(2*pi*f*C)).^2) // mA
    e = sum((I-Itheo).^2)
endfunction

--> erreur2([0.1 50])
ans =
    1.8842927

--> [Lr, erreurmin] = fminsearch(erreur2,x0=[0.1 50])
Lr =
    0.1012989    52.73462
erreurmin =
    0.0001126
```

- Licence

Cette œuvre est mise à disposition selon les termes de la **Licence Creative Commons** :

- Attribution
- Pas d'utilisation commerciale
- Partage dans les mêmes conditions
- version 3.0 France

<https://creativecommons.org/licenses/by-nc-sa/3.0/fr/>

© **Fabrice Sincère**

- Page d'accueil : <http://fabrice.sincere.pagesperso-orange.fr/>
<http://fabrice.sincere.free.fr/>
- Mail : fabrice.sincere@wanadoo.fr
fabrice.sincere@ac-grenoble.fr