Equations différentielles

Solutions analytiques d'équations différentielles

Solution par intégration directe

Une équation différentielle ordinaire (ODE) est une équation qui contient des dérivées ordinaires de la variable dépendante. Une équation qui contient des dérivées partielles par rapport à deux ou plus de variables indépendantes est une équation différentielle partielle (PDE).

Un exemple simple d'ODE est :

$$\frac{dy}{dt} = t^2$$

Ici, la variable dépendante est *y* et *t* est la variable indépendante. On peut résoudre pour *y* en intégrant des deux cotés de l'équation par rapport à la variable indépendante *t*.

$$\int_0 \frac{dy}{dt} dt = \int_0 t^2 dt = \left.\frac{t^3}{3}\right|_0^t = \frac{t^3}{3}$$

La solution est $y(t) = y(0) + \frac{t^3}{3}$

Equation du second ordre

L'ordre d'une équation différentielle est l'ordre de sa plus grande dérivée. L'équation suivante est du second ordre :

$$\frac{d^2y}{dt^2} = t^3$$

Pour la résoudre, on doit intégrer deux fois :

$$\int_0 \frac{d^2y}{dt^2} dt = \frac{dy}{dt} - \dot{y}(0) = \int_0 t^3 dt = \frac{t^4}{4}$$

$$\int_0 \frac{dy}{dt} dt = y(t) - y(0) = \int_0 \left[\frac{t^4}{4} + \dot{y}(0)\right] dt = \frac{t^5}{20} + t\dot{y}(0)$$

La solution est donc $y(t) = t^5 / 20 + t\dot{y}(0) + y(0)$. On notera que deux conditions initiales doivent être fixées pour compléter la solution : $y(0)$ et $\dot{y}(0)$.

Méthode de substitution pour les équations du premier degré

On considère l'équation :

$$\tau \frac{dy}{dt} + y = f(t)$$

où τ est une constante et *f(t)* une fonction donnée. Cette équation est souvent utilisée pour décrire des processus dépendants du temps comme la température de refroidissement d'un objet. On pourrait résoudre cette équation par intégration directe mais l'algèbre nécessaire est un peu compliqué, on va donc utiliser une autre méthode.

Les équations linéaire peuvent souvent être résolues avec une solution de test de la forme $y(t) = Ae^{st}$. En substituant cette forme et sa dérivée dans l'équation précédente et en prenant *f(t)*=0, on obtient :

$$\tau \frac{dy}{dt} + y = \tau s Ae^{st} + Ae^{st} = 0$$

Pour que la solution soit générale, $Ae^{st} \neq 0$ et $\tau s + 1 = 0$. Cette équation est *l'équation caractéristique* dont *la racine caractéristique* est $s = -1/\tau$. Pour trouver *A*, on évalue la solution en *t*=0 et on trouve $y(0) = A$. La solution est donc :

$$y(t) = y(0)e^{-t/\tau}$$

Cette solution est la *réponse libre* car elle décrit le comportement ou la réponse du processus quand la *fonction de force f(t)* est nulle, c'est-à-dire quand le processus est "libéré" de l'influence de *f(t)*. La solution *y(t)* diminue avec le temps si $\tau > 0$. τ, la constante de temps, est donc une indication de la vitesse d'amortissement de *y(t)*.

Supposons maintenant que *f(t)*=0 pour *t*<0 et *f(t)*=*M* à *t*=0. Une telle fonction est une fonction marche dont la hauteur est *M*. La solution dans ce cas est $y(t) = Ae^{st} + B$ et les conditions initiales donnent $B = y(0) - A$, donc $y(t) = Ae^{st} + y(0) - A$. En substituant dans l'équation différentielle, on trouve que $\tau s + 1 = 0$ et $A = y(0) - M$. La solution de *y(t)* est donc :

$$y(t) = y(0)e^{-t/\tau} + M(1 - e^{-t/\tau})$$

La *réponse forcée* est le terme $M(1 - e^{-t/\tau})$ qui est dû à la fonction de force.

On voit donc que la *réponse totale* pour cette équation est la somme des réponses libre et forcée.

Equations non linéaires

Les équations différentielles non linéaires se reconnaissent par le fait que la variable dépendante ou ses dérivées sont élevées à la puissance ou dans une fonction transcendante. Par exemple :

$$y\ddot{y} + 5\dot{y} + y = 0$$
$$\dot{y} + \sin y = 0$$
$$\dot{y} + \sqrt{y} = 0$$

Il n'existe pas de méthode générale de résolution pour de telles équations.

Méthode de substitution pour les équations du second degré

On considère l'équation du second ordre :

$$\ddot{y} - c^2 y = 0$$

Si on substitue $y(t) = Ae^{st}$ dans cette équation, on obtient :

$$\left(s^2 - c^2\right)Ae^{st} = 0$$

qui est satisfaite pour toutes les valeurs de *t* si $s^2 - c^2 = 0$. Donc $s = \pm c$ et la solution générale est :

$$y(t) = A_1 e^{ct} + A_2 e^{-ct}$$

On notera que la solution devient infinie quand $t \to \infty$ quelle que soit la valeur de *c*. Si la réponse libre devient infinie, l'équation est *instable*. Si la solution libre tend vers 0 , l'équation est *stable*.

On considère maintenant l'équation :

$$\ddot{y} + \omega^2 y = 0$$

Si on substitue $y(t) = Ae^{st}$ dans cette équation, on obtient la solution générale :

$$y(t) = A_1 e^{i\omega t} + A_2 e^{-i\omega t}$$

La solution est difficile à interpréter à moins d'utiliser les identités d'Euler : $e^{\pm i\omega t} = \cos\omega t \pm i \sin\omega t$. Si on substitue ces deux identités, on obtient :

$$y(t) = B_1 \sin\omega t + B_2 \cos\omega t$$

où B_1 et B_2 sont des constantes qui dépendent des conditions initiales :

$$y(t) = \frac{\dot{y}(0)}{\omega} \sin\omega t + y(0) \cos\omega t$$

La solution oscille avec une amplitude constante et une fréquence de ω radians par unité de temps. La période *P* de l'oscillation est le temps entre deux pics adjacents : $P = 2\pi / \omega$ et la fréquence en cycles par unité de temps est $f = 1/P$.

Méthodes numériques pour la résolution d'équations différentielles

Il n'est pas toujours d'obtenir une solution définie pour une équation différentielle. On utilise alors des méthodes numériques qui permettent de les résoudre. Le principe de ces méthodes est de convertir une équation différentielle en une équation de différence qui peut être programmée. En général, la précision de l'approximation augmente la complexité de la programmation.

Méthode d'Euler

La méthode d'Euler est l'algorithme le plus simple pour trouver une solution numérique à une équation différentielle. Elle donne en général les résultats les moins précis mais donne les bases de la compréhension pour des méthodes plus sophistiquées. On considère l'équation :

$$\frac{dy}{dt} = r(t)y$$

où *r(t)* est une fonction connue. D'après la définition de la dérivée, on a :

$$\frac{dy}{dt} = \lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t}$$

Si l'incrément de temps Δt est suffisamment petit, la dérivée peut être remplacée par l'expression d'approximation. On a alors :

$$\frac{y(t+\Delta t) - y(t)}{\Delta t} = r(t)y(t)$$

En supposant que la partie droite de l'équation reste constante sur l'intervalle $[t, t+\Delta t]$, on peut écrire :

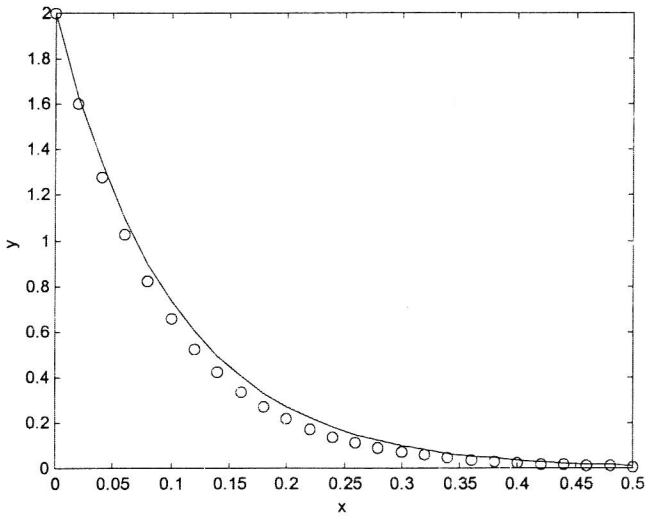
$$y(t_{k+1}) = y(t_k) + r(t_k)y(t_k)\Delta t$$

où $t_{k+1} = t_k + \Delta t$.

La méthode d'Euler pour une équation du premier ordre $\dot{y} = f(t,y)$ est : $y(t_{k+1}) = y(t_k) + \Delta t f\left[t_k, y(t_k)\right]$

où l'incrément Δt est le pas.

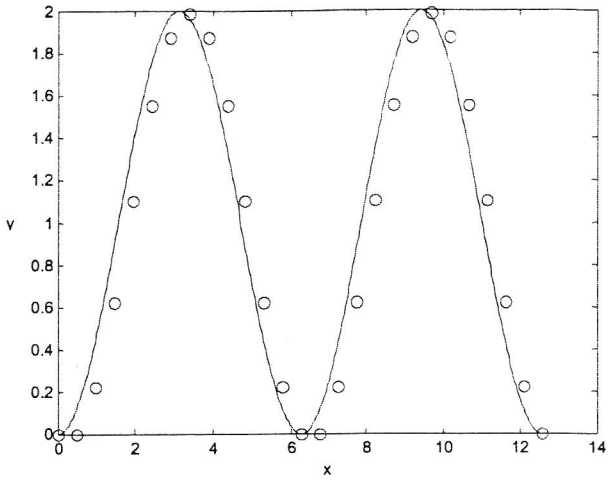
```
r=-10 ;delta=0.02 ;y(1)=2 ;
k=0 ;
for dt=delta :delta :0.5
      k=k+1 ;
   y(k+1)=y(k)+r*y(k)*delta ;
end
t=0 :delta :.5
y_vrai=2*exp(-10*t) ;
figure(1) ;
plot(t,y,'o',t,y_vrai) ;
xlabel('x') ;ylabel('y') ;
```



On voit qu'il existe une erreur notable entre la solution numérique (o) et la solution vraie. Si le pas utilisé est delta=0.005, l'erreur ne sera plus décelable.

Les méthodes numériques ont de grandes erreurs quand on tente d'obtenir des solutions qui varient rapidement. Des changements rapides peuvent être dus à de petites constantes de temps ou à des oscillations.

```
delta=2*pi/13 ;y(1)=0 ;
k=0 ;
for dt=delta :delta :4*pi
    k=k+1 ;
    y(k+1)=y(k)+sin(dt-delta)*delta ;
end
t_vrai=0 :delta/10 :4*pi ;
y_vrai=1-cos(t_vrai) ;
t=0 :delta :4*pi ;
figure(1) ;
plot(t,y,'o',t_vrai,y_vrai) ;
xlabel('x') ;ylabel('y') ;
```



Méthode d'Euler améliorée

Un façon d'améliorer la méthode d'Euler est de faire une meilleure approximation du membre droit de l'équation différentielle. L'approximation d'Euler est : $y(t_{k+1}) = y(t_k) + \Delta t f\left[t_k, y(t_k)\right]$. Supposons qu'à la place on prenne la moyenne du membre droit sur l'intervalle $[t_k, t_{k+1}]$, on a alors : $y(t_{k+1}) = y(t_k) + \frac{\Delta t}{2}(f_k + f_{k+1})$ avec $f_k = f\left[t_k, y(t_k)\right]$.

La difficulté est que f_{k+1} ne peut être évaluée que lorsque $y(t_{k+1})$ est connue, ce qui est précisément la quantité recherchée. On peut alors estimer $y(t_{k+1})$ avec la formule d'Euler, obtenir f_{k+1} que l'on utilisera dans l'équation ci-dessus pour calculer $y(t_{k+1})$. Notons $h=\Delta t$, $y_k=y(t_k)$ et x_{k+1} la valeur estimée de $y(t_{k+1})$. On obtient alors la description du processus de prédiction-correction :

prédicteur d'Euler : $x_{k+1} = y_k + hf(t_k, y_k)$

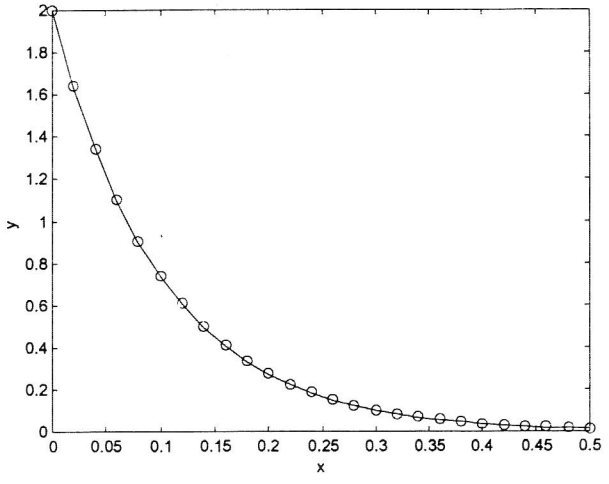
correcteur : $y_{k+1} = y_k + \frac{h}{2}\left[f(t_k, y_k) + f(t_{k+1}, x_{k+1})\right]$

que l'on peut encore exprimer sous la forme :

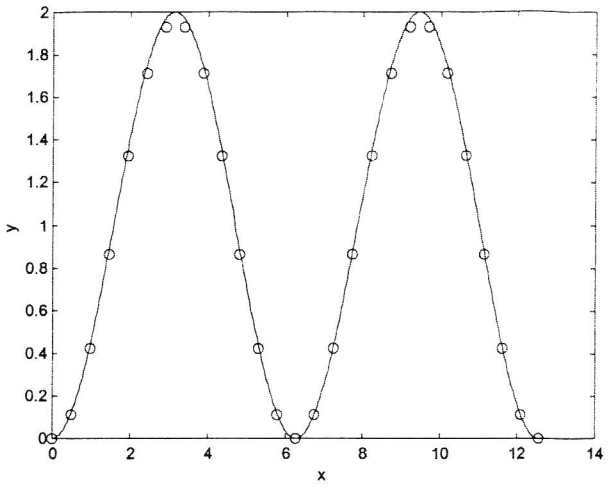
$$g_1 = hf(t_k, y_k)$$
$$g_2 = hf(t_k + h, y_k + g1)$$
$$y_{k+1} = y_k + \frac{1}{2}(g_1 + g_2)$$

Si on applique cette méthode aux exemples précédents :

```
r=-10 ;delta=0.02 ;y(1)=2 ;
k=0 ;
for dt=delta :delta :0.5
    k=k+1 ;
    x(k+1)=y(k)+delta*r*y(k) ;
    y(k+1)=y(k)+(delta/2)*(r*y(k)+r*x(k+1))
end
t=0 :delta :.5
y_vrai=2*exp(-10*t) ;
figure(1) ;
plot(t,y,'o',t,y_vrai) ;
xlabel('x') ;ylabel('y') ;
```



```
delta=2*pi/13 ;y(1)=0 ;
k=0 ;
for dt=delta :delta :4*pi
    k=k+1 ;
    y(k+1)=y(k)+(delta/2)*...
           (sin(dt-delta)+sin(dt)) ;
end
t_vrai=0 :delta/10 :4*pi ;
y_vrai=1-cos(t_vrai) ;
t=0 :delta :4*pi ;
figure(1) ;
plot(t,y,'o',t_vrai,y_vrai) ;
xlabel('x') ;ylabel('y') ;
```



On voit que les erreurs sont moindres mais encore existantes au voisinage des sommets dans le deuxième exemple.

Méthodes de Runge-Kutta

La *série de Taylor* forme la base de nombreuses méthodes de résolution des équations différentielles et notamment des *méthodes de Runge-Kutta*. Elle peut servir à représenter la solution *y(t+h)* en termes de *y(t)* et de ses dérivées :

$$y(t+h) = y(t) + h\dot{y}(t) + \frac{1}{2}h^2\ddot{y}(t) + \ldots$$

Le nombre de termes conservés dans la série défini la précision. Les dérivées sont calculées à partir de l'équation différentielle. Si ces dérivées peuvent être trouvées, la série de Taylor précédente peut être utilisée pour représenter dans le temps. En pratique, les dérivées d'ordre supérieur sont difficiles à calculer et la série est tronquée à partir de certains termes.

Les méthodes de Runge-Kutta ont été développées à cause de cette difficulté à calculer les dérivées d'ordre supérieur. Elles utilisent plusieurs évaluations de la fonction *f(t,y)* qui approximent la série de Taylor. Le nombre de termes dans la série développée détermine l'ordre de la méthode de Runge-Kutta. Ainsi, l'algorithme de Runge-Kutta à l'ordre 4 développe la série de Taylor jusqu'au terme en h^4.

La méthode de Runge-Kutta à l'ordre 2 exprime y_{k+1} tel que :

$$y_{k+1} = y_k + w_1 g_1 + w_2 g_2$$

où w_1 et w_2 sont des facteurs constants et

$$g_1 = hf(t_k, y_k)$$
$$g_2 = hf(t_k + \alpha h, y_k + \beta h f_k)$$

La famille des algorithmes de Runge-Kutta à l'ordre 2 est caractérisée par les paramètres α, β, w_1 et w_2. Pour développer la série de Taylor jusqu'au terme en h^2, ces coefficients doivent satisfaire les relations :

$$w_1 + w_2 = 1$$
$$w_1\alpha = 1/2$$
$$w_2\beta = 1/2$$

Donc, un des paramètres peut être choisi indépendamment.

La famille des algorithmes de Runge-Kutta à l'ordre 4 exprime y_{k+1} tel que :

$$y_{k+1} = y_k + w_1 g_1 + w_2 g_2 + w_3 g_3 + w_4 g_4$$

$$g_1 = hf(t_k, y_k)$$
$$g_2 = hf(t_k + \alpha_1 h, y_k + \alpha_1 g_1)$$
$$g_3 = hf\left[t_k + \alpha_2 h, y_k + \beta_2 g_2 + (\alpha_2 - \beta_2) g_1\right]$$
$$g_4 = hf\left[t_k + \alpha_3 h, y_k + \beta_3 g_2 + \gamma_3 g_3 + (\alpha_3 - \beta_3 - \gamma_3) g_1\right]$$

La comparaison avec la série de Taylor conduit à 8 équations et 10 paramètres. Deux peuvent donc être choisi indépendamment, on choisit en général :

$w_1 = w_4 = 1/6$
$w_2 = w_3 = 1/3$
$\alpha_1 = \alpha_2 = 1/2$
$\beta_2 = 1/2$
$\gamma_3 = \alpha_3 = 1$
$\beta_3 = 0$

Fonctions Matlab ode23 et ode45

Les fonctions Matlab ode23 et ode45 implémentent les méthode de Runge-Kutta avec un pas variable. ode23 est une combinaison des méthodes à l'ordre 2 et 3, ode45une combinaison des méthodes à l'ordre 4 et 5. En général, ode45 est plus rapide et plus précise que ode23 mais utilise un pas plus grand qui peut produire une solution graphique peu lissée.

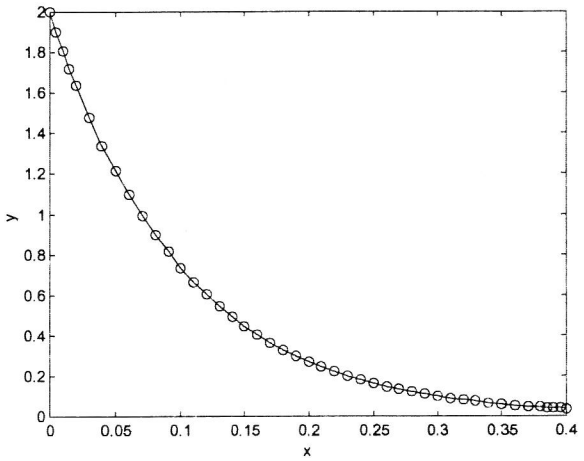
La syntaxe est `[t,y]=ode23('equadif',tspan,y0)`.

`equadif` est le nom du fichier contenant la fonction dont les arguments d'entrée sont *t* et *y* et dont l'argument de sortie doit être un vecteur colonne représentant *dy/dt*, c'est-à-dire *f(t,y)*. Le nombre de ligne dans ce vecteur colonne doit être égal à l'ordre de l'équation.

Le vecteur ligne `tspan` contient les valeurs de départ et de fin de la variable indépendante *t* et occasionnellement toute valeur intermédiaire pour laquelle on recherche une solution.

Le paramètre `y0` est la condition initiale $y(t_0)$.

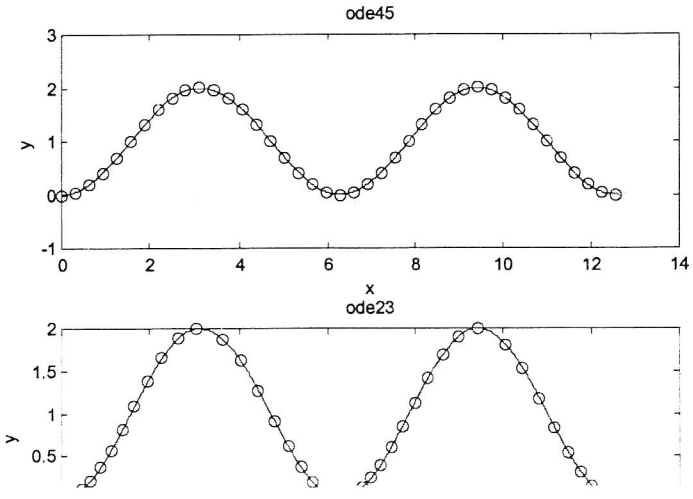
```
function dy=rccirc(t,y)
% modèle de circuit RC sans tension appliquée
% fichier rccirc.m
dy=-10*y ;
%programme principal
[t,y]=ode45('rccirc',[0 0.4],2) ;
y_vrai=2*exp(-10*t) ;
figure(1) ;
plot(t,y,'o',t,y_vrai) ;
xlabel('x') ;ylabel('y') ;
```



Taille du pas

Le pas utilisé par ode23 est plus petit que celui utilisé par ode45. En effet, cette dernière a une erreur de troncation plus petite et peut donc utiliser un pas plus grand. Quand la solution change rapidement, comme pour une solution oscillante, les points de la solution numérique ne donnent pas toujours une courbe lissée. Il est donc parfois plus utile d'utiliser ode23 pour tracer une solution.

```
function dy=sinfc(t,y)
% fonction sinus
% fichier sinfc.m
dy=sin(t) ;
%programme principal
t_vrai=0 :0.01 :4*pi ;
figure(1) ;
subplot(2,1,1) ;
[t,y]=ode45('sinfn',[0 4*pi],0) ;
y_vrai=1-cos(t_vrai) ;
plot(t,y,'o',t_vrai,y_vrai) ;
xlabel('x');ylabel('y');title('ode45')
subplot(2,1,2) ;
```



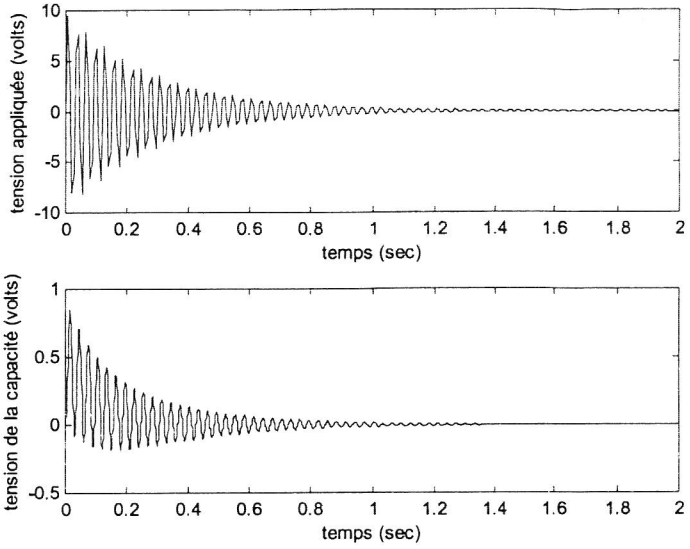
```
[t,y]=ode23('sinfn',[0 4*pi],0) ;
y_vrai=1-cos(t_vrai) ;
plot(t,y,'o',t_vrai,y_vrai) ;
xlabel('x') ;ylabel('y') ;
title('ode23') ;
```

Utilisation de paramètres globaux

La commande `global x y z` permet à toutes les fonctions et tous les fichiers utilisant cette commande de partager les valeurs des variables `x`, `y` et `z`. L'utilisation de `global` permet notamment d'éviter la nécessité de changer la valeur de certains paramètres dans chaque fichier où ils sont utilisés.

```
function dy=circuit(t,y)
% modèle circuit RC avec amortissement sinusoïdal
% fichier circuit.m
global tau_1 P
v=10*exp(-t/tau_1)*sin((2*pi/P)*t) ;
dy=10*(v-y) ;

%programme principal
global tau_1 P tf
tau_1=0.3 ;P=0.03 ;tf=2 ;
[t,y]=ode23('circuit',[0 tf],0) ;
tp=0 :tf/300 :tf ;
v=10*exp(-tp/tau_1)*sin((2*pi/P)*tp);
figure(1) ;
subplot(2,1,1) ;
plot(tp,v) ;
xlabel('temps (sec)');
ylabel('tension appliquée (volts)');
subplot(2,1,2) ;
plot(t,y) ;
xlabel('temps (sec)') ;
ylabel('tension de la capacité (volts)')
```



Extension aux équations d'ordre supérieur

Pour résoudre une équation différentielle d'ordre supérieur àl'ordre 2, on doit d'abord la réécrire comme un système d'équation du premier ordre.

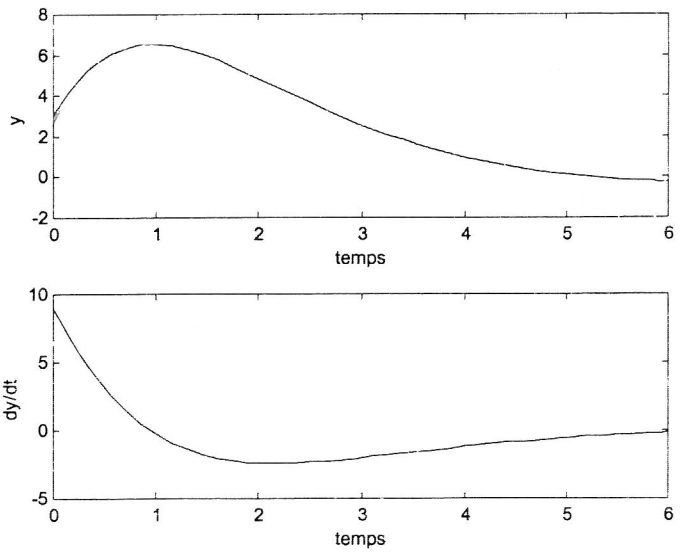
Considérons l'équation du second ordre : $5\ddot{y}+7\dot{y}+4y=f(t)$ que l'on peut écrire sous la forme $\ddot{y}=\frac{1}{5}f(t)-\frac{7}{5}\dot{y}-\frac{4}{5}y$.

En posant $\begin{matrix} x_1 = y \\ x_2 = \dot{y} \end{matrix}$, on alors le système : $\begin{matrix} \dot{x}_1 = x_2 \\ \dot{x}_2 = \frac{1}{5}f(t)-\frac{4}{5}x_1-\frac{7}{5}x_2 \end{matrix}$

On peut maintenant écrire une fonction Matlab qui caculera les valeurs de $\dot{x}_1$ et $\dot{x}_2$ les stockera dans un vecteur colonne dx

```
function dx=exemple(t,x)
% fichier exemple.m
dx(1)=x(2) ;
dx(2)=(1/5)*(sin(t)-4*x(1)-7*x(2));
dx=[dx(1) ;dx(2)] ;

%programme principal
[t,x]=ode45('exemple',[0 6],[3 9]);
figure(1) ;
subplot(2,1,1) ;
plot(t,x(:,1)) ;
xlabel('temps');
ylabel('y');
subplot(2,1,2) ;
plot(t,x(:,2)) ;
xlabel('temps') ;
ylabel('dy/dt') ;
```



On notera que dx(1) et dx(2) représentent $\dot{x}_1$ et $\dot{x}_2$; et que x(1) et x(2) représentent x_1 et x_2.

Dans la fonction exemple.m, les trois lignes de programmation pourraient être remplacées par une seule :

```
dx=[x(2) ; (1/5)*(sin(t)-4*x(1)-7*x(2))];
```

On notera enfin que les solutions x_1 (=y) et x_2 (=dy/dt) sont retournées sous la forme d'une matrice x de deux colonnes, la première contenant les valeurs de x_1 (x(:,1)), la seconde les valeurs de x_2 (x(:,2)).

L'instruction plot(t,x) tracera en même temps x1 et x2 en fonction du temps.

Solutions d'équations non linéaires

On a vu précédemment que lorsqu'on résout une équation non linéaire, il est possible de vérifier le résultat numérique en utilisant une approximation qui réduit l'équation à une équation linéaire. Prenons l'exemple d'un pendule non linéaire dont l'équation de mouvement est :

$$\ddot{\theta}+\frac{g}{L}\sin\theta=0$$

Si l'on utilise une approximation aux petits angles sin ≈ θ, l'équation devient : $\ddot{\theta}+\frac{g}{L}\theta=0$ qui est linéaire et a comme solution $\theta(t)=\theta(0)\cos\sqrt{\frac{g}{L}}t$.

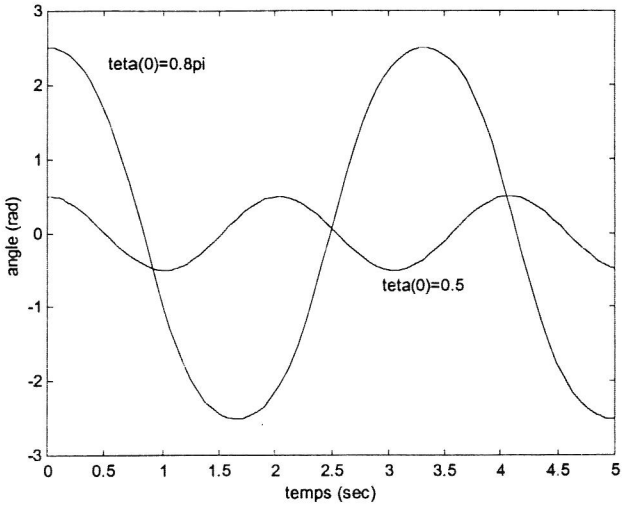
Donc, pour résoudre l'équation différentielle du second ordre, on la réécrit, avec $x_1=\theta$ et $x_2=\theta'$, sous le système :

$$\dot{x}_1=\dot{\theta}=x_2$$

$$\dot{x}_2=\ddot{\theta}=-\frac{g}{L}\sin x_1$$

```
function dx=pendule(t,x)
% fichier pendule.m
global g L
dx=[x(2) ;-(g/L)*sin(x(1))] ;

%programme principal
global g L
g=9.81 ;L=1 ;
[ta,xa]=ode45('pendule',[0 5],[.5 0]);
[tb,xb]=ode45('pendule',[0 5],[.8*pi 0]);
figure(1) ;
plot(ta,xa(:,1),tb,xb(:,1));
xlabel('temps (sec)');
ylabel('angle (rad)');
gtext('teta(0)=0.5') ;
gtext('teta(0)=0.8pi') ;
```



Méthodes matricielles

On peut utiliser les opérations matricielles pour diminuer le nombre de lignes à taper dans la fonction de l'équation différentielle. Prenons l'exemple d'une masse m reliée à un ressort de raideur k et subissant une force de friction visqueuse de module c et une force $f(t)$. L'équation de son mouvement est alors :

$$m\ddot{y}+c\dot{y}+ky=f(t)$$

En posant $\begin{matrix}\dot{x}_1=y\\ x_2=\dot{y}\end{matrix}$, on a alors le système : $\begin{matrix}\dot{x}_1=x_2\\ \dot{x}_2=\frac{1}{m}f(t)-\frac{k}{m}x_1-\frac{c}{m}x_2\end{matrix}$ que l'on peut écrire sous la forme matricielle :

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} f(t)$$ soit sous forme compacte $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}f(t)$

```
function dx=msd(t,x)
% fichier msd.m pour la masse avec un ressort et une friction.
% la position est la première variable, la vitesse la seconde.
global c f k m
A=[0 1 ;-k/m -c/m] ;
B=[0 ;1/m] ;
dx=A*x+B*f ; ;

%programme principal
global c f k m
c=2 ;f=10 ;k=5 ;m=1 ;
[t,x]=ode23('msd',[0 5],[0 0]);
figure(1) ;
plot(t,x);
xlabel('temps (sec)');
gtext('déplacement') ;
gtext('vitesse') ;
```