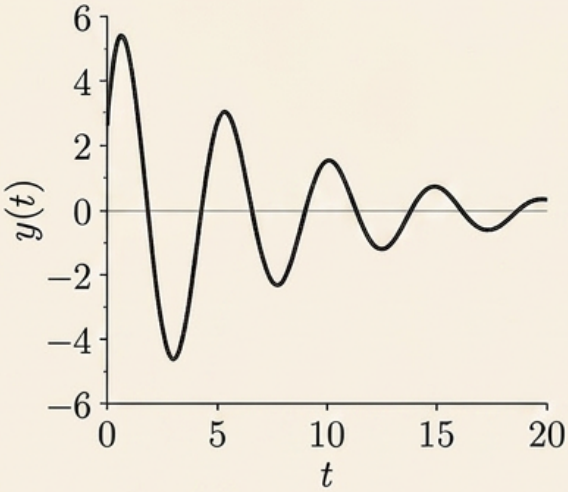
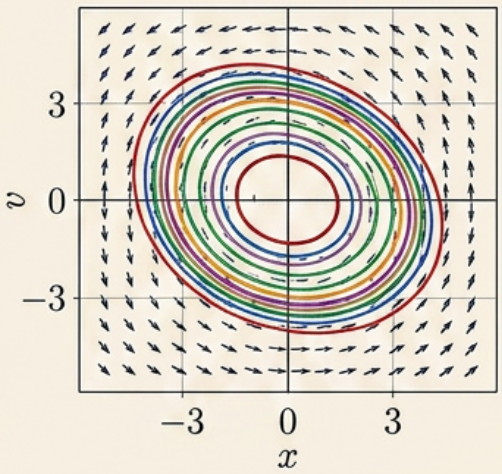
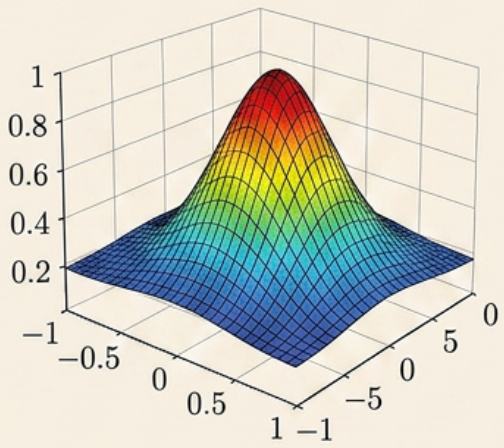
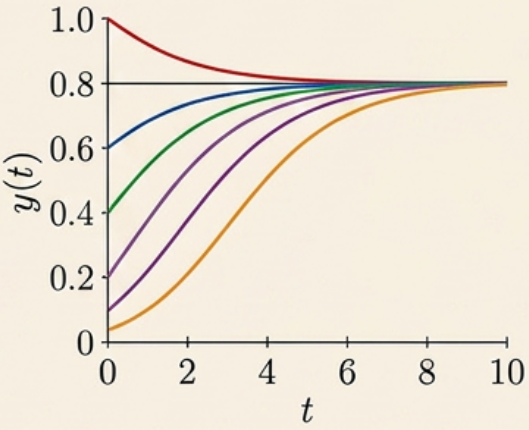
# MAXIMA
# PAR L'EXEMPLE

## VOLUME 3
## Guide des équations différentielles avec Maxima



# Guide des équations différentielles avec Maxima

## Sommaire

# 1 Introduction

Ce guide est le troisième volume de la collection "Maxima par l'exemple". Il se propose de montrer comment il est possible d'utiliser Maxima afin de résoudre des équations différentielles, soit de manière formelle, soit à l'aide de méthodes numériques.
Ce document, ainsi que les versions pdf et wxMaxima de ce Guide des équations différentielles avec Maxima sont disponibles sur le site https://maxima-french-doc.fr.

En complément des commandes standart de Maxima pour résoudre les équations différentielles, Maxima dispose aussi de packages contenant des méthodes de résolution pour certains types d'équations différentielles. Nous présenterons certains de ces packages dans ce document.

Ce guide est basé sur la version 5.49 de Maxima, et s'applique donc aussi à toutes les versions ultérieures. Je tiens à remercier le *Dr Wolfgang Lindner* dont l'ouvrage Introductory Differential Equations with Maxima a été une source d'inspiration précieuse lors de l'élaboration de ce guide. Il me faut citer aussi les manuels de *Edwin L. Woollett* intitulés Maxima by Example, en particulier le chapitre 3 intitulé Ordinary Differential Equation Tools. Enfin, le manuel wxMaxima for calculus volume 2 et en particulier le chapitre 4 contient des informations sur ce sujet très utiles.

On supposera dans cet ouvrage que les fonctions considérées sont dérivables autant de fois qu'il le faut sur les intervalles d'étude. Les chapitres de ce guide traiteront des *équations différentielles ordinaires (EDO)* qui sont caractérisées par le fait que la fonction inconnue ne dépend que d'une seule variable (souvent notée $x$ ou $t$). L'équation différentielle relie cette fonction à ses dérivées successives par rapport à cette unique variable. L'autre catégorie d'équations différentielles concerne les fonctions à plusieurs variables, dont l'équation implique des dérivées partielles, et qui seront désignées par l'acronyme *EDP (équations aux dérivées partielles)*

# 2 Définir une EDO avec Maxima

Si $y$ est une fonction de la variable $x$, ses dérivées successives $\frac{\mathrm{d}y}{\mathrm{d}x}$, $\frac{\mathrm{d}^2y}{\mathrm{d}x^2}$, ... se définissent respectivement dans Maxima par `diff(y,x)` et `diff(y,x,2)` ...
Ces dérivées permettent donc d'écrire une équation différentielle, à condition de faire précéder `diff` par **une apostrophe '** afin d'éviter que Maxima ne calcule immédiatement la dérivée.

**Commande Maxima :**

```
'diff(y,x) = -y;
'diff(y(x),x) = -y(x);
'diff(y,x) + (1/(2*x))*y = 2;
'diff(y,x,2) + y = 0;
'diff(f,x,2) + 3*'diff(f,x) = k*f;
```

*Commentaire :*

- La fonction dans l'équation différentielle peut être notée $y$ ou $y(x)$. La notation $y(x)$ est par contre indispensable s'il y a plusieurs variables.
- On peut assigner une EDO à une variable avec la commande : `eq:'diff(y,x)=-y`.

**Résultat :**

(% i1) 'diff(y,x) = -y ;

$$\frac{d}{dx}y = -y \qquad (\%\ o1)$$

(% i2) 'diff(y(x),x) = -y(x) ;

$$\frac{d}{dx}y(x) = -y(x) \qquad (\%\ o2)$$

(% i3) 'diff(y,x) + (1/(2*x))*y = 2;

$$\frac{d}{dx}y+\frac{y}{2x}=2 \qquad (\%\ o3)$$

(% i6) 'diff(y,x,2) + y = 0;

$$\frac{d^2}{dx^2}y+y=0 \qquad (\%\ o6)$$

(% i7) 'diff(f,x,2) + 3*'diff(f,x) = k*f;

$$\frac{d^2}{dx^2}f+3\left(\frac{d}{dx}f\right)=fk \qquad (\%\ o7)$$

## 3 Vérifier si une fonction est solution d'une EDO

*Principe* : on remplace la fonction inconnue de l'équation différentielle par la fonction à tester.

**Commande Maxima :**

```
eq: (1+x^2)*'diff(y,x) + 2*x*y = 4*x^3
sol: y = (x^4+1)/(1+x^2);
verif: ev(eq, sol, diff);
ratsimp(verif);
```

*Commentaire :* La première commande définit l'EDO, la deuxième une solution à tester. La troisième commande remplace dans l'EDO y par l'expression définie dans sol, puis calcule les dérivées. La dernière commande simplifie afin de voir si l'équation différentielle est bien vérifiée.

**Résultat :**

(% i6) eq : (1+x^ 2)*'diff(y,x) + 2*x*y = 4*x^ 3;

$$\left(x^2+1\right)\left(\frac{d}{dx}y\right)+2xy=4x^3 \qquad (\text{eq})$$

(% i7) sol : y = (x^ 4+1)/(1+x^ 2);

$$y=\frac{x^4+1}{x^2+1} \qquad (\text{sol})$$

(% i8) verif : ev(eq, sol, diff);

$$\left(x^2+1\right)\left(\frac{4x^3}{x^2+1}-\frac{2x\ \left(x^4+1\right)}{\left(x^2+1\right)^2}\right)+\frac{2x\ \left(x^4+1\right)}{x^2+1}=4x^3 \qquad (\text{verif})$$

(% i9) ratsimp(verif);

$$4x^3=4x^3 \qquad (\%\ o9)$$

**Commande Maxima :**

```
f(x):= (x^4+1)/(1+x^2);
(1+x^2)*diff(f(x),x)+2*x*f(x);
ratsimp(%);
```

*Commentaire :* On utilise une méthode classique dans cet exemple. On définit la fonction $f$ dont on veut savoir si elle est solution, et on calcule de manière explicite le premier membre de l'EDO en remplaçant $y$ par $f(x)$ et en évaluant la dérivée. Il reste à simplifier pour conclure.

**Résultat :**

(% i10) f(x) := (x^ 4+1)/(1+x^ 2);

$$f(x) := \frac{x^4+1}{1+x^2} \qquad (\%\ o10)$$

(% i11) (1+x^ 2)*diff(f(x),x)+2*x*f(x);

$$\left(x^2+1\right)\left(\frac{4x^3}{x^2+1}-\frac{2x\ \left(x^4+1\right)}{\left(x^2+1\right)^2}\right)+\frac{2x\ \left(x^4+1\right)}{x^2+1} \qquad (\%\ o11)$$

(% i12) ratsimp(%);

$$4x^3 \qquad (\%\ o12)$$

**Commande Maxima :**

```
test(fun, equation, var) := block(
    [r],
    r : subst(fun, y, equation),
    r : ev(r, diff),
    is(fullratsimp(lhs(r)-rhs(r)) = 0)
)$
eq2 : 'diff(y,x,2) + y = 0;
f(x):=k*sin(x);
test(f(x),eq2,x);
test(x*sin(x),eq2,x);
```

*Commentaire :* Pour automatiser le calcul précédent, on définit une fonction qui teste si une fonction est solution d'une EDO. Les paramètres sont la fonction à tester, l'équation différentielle, et la variable de dérivation.
Pour le calcul, on définit une nouvelle équation différentielle eq2, une fonction $f$ définie par $f(x) = k\sin(x)$ et on applique la procédure test. On fait de même avec la fonction $x \longmapsto x\sin(x)$.

**Résultat :**

(% i18) eq2 : 'diff(y,x,2) + y = 0;

$$\frac{d^2}{dx^2}y + y = 0 \qquad (eq2)$$

(% i19) f(x) :=k*sin(x);

$$f(x) := k\sin(x) \qquad (\%\ o19)$$

(% i20) test(f(x),eq2,x);

$$true \qquad (\%\ o20)$$

(% i21) test(x*sin(x),eq2,x);

$$false \qquad (\%\ o21)$$

# 4 Résoudre une EDO avec ode2, le solveur intégré de Maxima

Sans rentrer dans les cas particuliers, la théorie montre qu'une équation différentielle ordinaire admet une famille de fonctions solutions, appelée solution générale. Pour une EDO du premier ordre (équation définie par la dérivée première de $y$), la famille de fonctions est définie à une constante près. Pour une équation du deuxième ordre, elle est définie à deux constantes près. Maxima notera ces constantes par défaut %c, %k1...
**La commande intégrée de Maxima pour trouver la solution générale d'une EDO de degré 1 ou 2 est**
`ode2(équation différentielle,fonction inconnue,variable).`

**Commande Maxima :**

```
eq1 : 'diff(y,x) + 2*y = 6*x;
sol1 : ode2(eq1, y, x);
```

*Commentaire :* Application directe de la commande de résolution pour une EDO de degré 1.

**Résultat :**

```
(% i2)  eq1 : 'diff(y,x) + 2*y = 6*x;
        sol1 : ode2(eq1, y, x);
```

$$\frac{d}{dx}y+2y=6x \tag{eq1}$$

$$y=\%e^{-(2x)}\left(\frac{3\%e^{2x}\,(2x-1)}{2}+\%c\right) \tag{sol1}$$

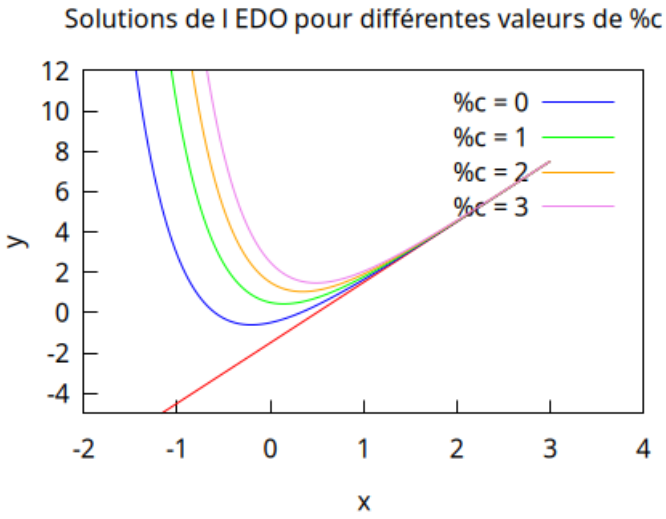
Représentons graphiquement quelques courbes de cette famille de solution :

**Commande Maxima :**

```
s1 : rhs(sol1)$
wxdraw2d(
    xrange = [-2, 4],
    yrange = [-5, 12],
    xlabel = "x",
    ylabel = "y",
    title = "Solutions de l EDO pour différentes valeurs de %c",
    color = red,
    explicit(subst(0, %c, s1), x, -3, 3),    key = "%c = 0",
    color = blue,
    explicit(subst(1, %c, s1), x, -3, 3),    key = "%c = 1",
    color = green,
    explicit(subst(2, %c, s1), x, -3, 3),    key = "%c = 2",
    color = orange,
    explicit(subst(3, %c, s1), x, -3, 3),    key = "%c = 3",
    color = violet,
    explicit(subst(4, %c, s1), x, -3, 3),    key = "%c = 4"
)$
```

*Commentaire :* La commande `rhs(sol1)` permet d'extraire le membre de droite de l'expression générale de la solution sol1, définie par $y = ....$

**Résultat :**




**Commande Maxima :**

```
eq2 : 'diff(y,x,2) - 3*'diff(y,x) + 2*y = 0;
sol2 : ode2(eq2, y, x);
```

*Commentaire :* Résolution pour une EDO de degré 2.

**Résultat :**

(% i4) eq2 : 'diff(y,x,2) - 3*'diff(y,x) + 2*y = 0;
sol2 : ode2(eq2, y, x);

$$\frac{d^2}{dx^2}y - 3\left(\frac{d}{dx}y\right) + 2y = 0 \tag{eq2}$$

$$y = \%e^{x}\%\mathrm{k2} + \%e^{2x}\%\mathrm{k1} \tag{sol2}$$

**Commande Maxima :**

```
eq3 : 'diff(y,x,2) + y = sin(x);
sol3 : ode2(eq3, y, x);
```

*Commentaire :* EDO de degré 2 avec second membre.

**Résultat :**

(% i6) eq3 : 'diff(y,x,2) + y = sin(x);
sol3 : ode2(eq3, y, x);

$$\frac{d^2}{dx^2}y + y = \sin(x) \tag{eq3}$$

$$y = \%\mathrm{k1}\sin(x) - \frac{x\cos(x)}{2} + \%\mathrm{k2}\cos(x) \tag{sol3}$$

## 5 Solution particulière d'une EDO du premier ordre

Si nous rajoutons une condition (valeur de la fonction en un point, valeur de la dérivée en un point,...), on peut alors déterminer une unique fonction parmi la famille de solutions de l'EDO du premier ordre vérifiant cette condition. Cette dernière est la solution particulière vérifiant la condition donnée.

*Principe.* On commence par résoudre l'équation différentielle afin d'obtenir sa solution générale, qui contient une constante d'intégration. On applique ensuite la condition initiale à cette solution générale pour déterminer la valeur de cette constante. On obtient ainsi la solution particulière satisfaisant la condition initiale donnée.

## 5.1 Calculer une solution particulière avec Maxima

Trouver la solution particulière $fp$ de l'EDO $\frac{dy}{dx} + 2y = x$ qui vérifie $y(0) = 1$. (On dit dans ce cas que l'on résout le problème de Cauchy)

**Commande Maxima :**

```
eq : 'diff(y,x) + 2*y = x$
sol:ode2(eq,y,x);
define(f(x),rhs(sol));
const:solve(f(0)=1,%c);
define(fp(x),subst(rhs(const[1]),%c,f(x)));
expand(fp(x));
```

*Commentaire :* on définit l'EDO, on la résout avec ode2, on définit la solution générale $f$. On calcule la constante %c en résolvant une équation prenant en compte la condition initiale, et on définit la solution particulière $fp$.

**Résultat :**

(% i2) eq : 'diff(y,x) + 2*y = x ;
sol :ode2(eq,y,x) ;

$$\frac{d}{dx}y + 2y = x \tag{eq}$$

$$y = \%e^{-(2x)}\left(\frac{\%e^{2x}\,(2x-1)}{4} + \%c\right) \tag{sol}$$

(% i3) define(f(x),rhs(sol)) ;

$$f(x) := \%e^{-(2x)}\left(\frac{\%e^{2x}\,(2x-1)}{4} + \%c\right) \tag{\% o3}$$

(% i4) const :solve(f(0)=1,%c) ;

$$\left[\%c = \frac{5}{4}\right] \tag{const}$$

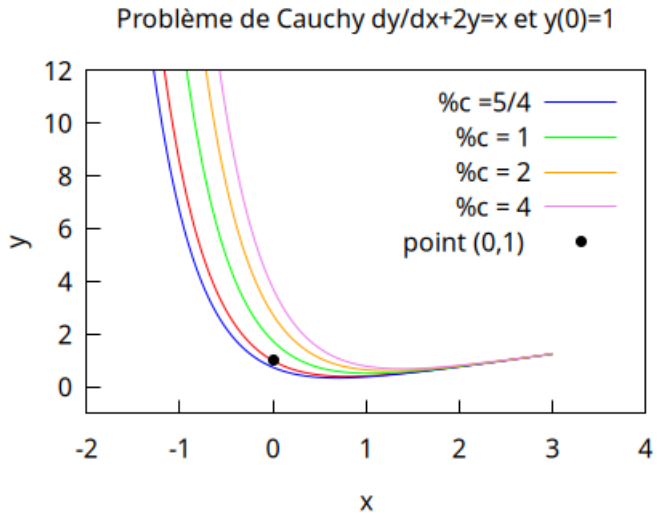
(% i5) define(fp(x),subst(rhs(const[1]),%c,f(x))) ;

$$fp(x) := \%e^{-(2x)}\left(\frac{\%e^{2x}\,(2x-1)}{4} + \frac{5}{4}\right) \tag{\% o5}$$

(% i6) expand(fp(x)) ;

$$\frac{x}{2} + \frac{5\%e^{-(2x)}}{4} - \frac{1}{4} \tag{\% o6}$$

Représentons ce problème de Cauchy. Pour cela, dessinons quelques fonctions solutions et la fonction particulière passant par le point de coordonnées $(0, 1)$.




### 5.2 Utiliser la commande intégrée ic1 de Maxima

Pour que Maxima calcule directement l'expression de la solution de l'équation différentielle du premier ordre vérifiant $y(x_0) = y_0$, il faut d'abord trouver la solution générale avec `ode2`, puis utiliser la commmande `ic1` avec la syntaxe suivante :
`ic1(solution_generale, x=x0, y=y0)`

**Commande Maxima :**

```
eq2 : 'diff(y,x) - 3*y = 6*exp(x);y(0)=2;
solgen : ode2(eq2,y,x);
solpart : ic1(solgen,x=0,y=2);
```

*Commentaire :* Recherche de la solution particulière de $\frac{dy}{dx} - 3y = 6e^x$ vérifiant $y(0) = 2$.

**Résultat :**

(% i29) eq2 : 'diff(y,x) - 3*y = 6*exp(x);y(0)=2;

$$\frac{d}{dx}y - 3y = 6\%e^x \tag{eq2}$$

$$y(0) = 2 \tag{\% o29}$$

(% i34) solgen : ode2(eq2,y,x);

$$y = \%e^{3x}\left(\%c - 3\%e^{-(2x)}\right) \tag{solgen}$$

(% i35) solpart : ic1(solgen,x=0,y=2);

$$y = 5\%e^{3x} - 3\%e^x \tag{solpart}$$

## 6 Solution particulière d'une EDO d'ordre 2

La solution générale d'une ODE d'ordre 2 est définie par deux paramètres. Si nous imposons deux conditions (valeurs de la fonction en certains points ou bien valeurs de la dérivée en certains points,...), alors on peut déterminer une unique fonction parmi la famille de solutions de l'EDO d'ordre 2 vérifiant ces deux conditions.

*Principe.* On commence par résoudre l'équation différentielle avec `ode2` afin d'obtenir sa solution générale, qui contient deux constantes d'intégration. On applique ensuite les conditions initiales à cette solution générale pour déterminer la valeur de ces constantes. On obtient ainsi la solution particulière satisfaisant au problème de Cauchy pour cette catégorie d'EDO.

## 6.1 Calculer une solution particulière avec Maxima

Cette méthode s'applique dans tous les cas où la fonction particulière est définie par deux conditions, contrairement aux commandes intégrées qui ne s'appliquent que dans certains cas.

Trouver la solution particulière de $\frac{d^2y}{dx^2} + y = 0$ avec $y(0) = 2$ et $y'\left(\frac{\pi}{4}\right) = 1$

**Commande Maxima :**

```
eq: 'diff(y,x,2)+y = 0;y(0)=2;yprime(%pi/4)=1;
sol_gen: ode2(eq, y, x);
define(f(x),rhs(sol_gen));
define(fprime(x),diff(f(x),x));
constantes:solve([f(0)=2,fprime(%pi/4)=1],[%k1,%k2]);
subst(constantes,f(x));
```

*Commentaire :* Définir $f'$ comme fonction simplifie ensuite la résolution du système d'équations permettant de déterminer les constantes d'intégration.

**Résultat :**

**(% i20)** eq : 'diff(y, x, 2) + y = 0 ;y(0)=2 ;yprime(%pi/4)=1 ;
sol_gen : ode2(eq, y, x) ;

$$\frac{d^2}{dx^2}y + y = 0 \qquad \text{(eq)}$$

$$y(0) = 2 \qquad (\%\ o18)$$

$$yprime\left(\frac{\pi}{4}\right) = 1 \qquad (\%\ o19)$$

$$y = \%k1 \sin(x) + \%k2 \cos(x) \qquad \text{(sol\_ gen)}$$

**(% i5)** define(f(x),rhs(sol_gen)) ;

$$f(x) := \%k1 \sin(x) + \%k2 \cos(x) \qquad (\%\ o5)$$

**(% i6)** define(fprime(x),diff(f(x),x)) ;

$$fprime(x) := \%k1 \cos(x) - \%k2 \sin(x) \qquad (\%\ o6)$$

**(% i9)** constantes :solve([f(0)=2,fprime(%pi/4)=1],[%k1,%k2]) ;

$$\left[\left[\%k1 = \sqrt{2} + 2\,,\%k2 = 2\right]\right] \qquad \text{(constantes)}$$

**(% i16)** subst(constantes,f(x)) ;

$$\left(\sqrt{2} + 2\right)\sin(x) + 2\cos(x) \qquad (\%\ o16)$$

## 6.2 Utiliser la commande intégrée ic2 de Maxima

La commande `ic2` permet de déterminer la solution particulière lorsque les deux conditions portent sur les valeurs de $y(x_0) = y_0$ et de $y'(x_0) = yp_0$ qui sont connues en un même point $x_0$. Syntaxe de la commande :
`ic2(solution_generale, x=x0, y=y0, 'diff(y,x)=yp0)`

Trouver la solution particulière de $\frac{d^2y}{dx^2} + 3\frac{dy}{dx} + 2y = 0$ avec $y(0) = 1$ et $y'(0) = 0$. On parle dans ce cas de deux conditions initiales.

**Commande Maxima :**

```
eq: 'diff(y,x,2) + 3*'diff(y,x) + 2*y = 0;
sol_gen: ode2(eq,y,x);
sol_part: ic2(sol_gen, x=0, y=1, 'diff(y,x)=0);
```

**Résultat :**

(% i16) eq : 'diff(y, x, 2) + 3*'diff(y, x) + 2*y = 0 ;y(0)=1 ;yprime(0)=0 ;
sol_gen : ode2(eq, y, x) ;
sol_part : ic2(sol_gen, x=0, y=1, 'diff(y,x)=0) ;

$$\frac{d^2}{dx^2}y + 3\left(\frac{d}{dx}y\right) + 2y = 0 \qquad \text{(eq)}$$

$$y(0) = 1 \qquad (\%\ \text{o13})$$

$$yprime(0) = 0 \qquad (\%\ \text{o14})$$

$$y = \%e^{-(2x)}\%\text{k2} + \%e^{-x}\%\text{k1} \qquad \text{(sol\_ gen)}$$

$$y = 2\%e^{-x} - \%e^{-(2x)} \qquad \text{(sol\_ part)}$$

## 6.3 Utiliser la commande intégrée bc2 de Maxima

La commande `bc2` permet de résoudre le problème aux limites. Dans ce cas, les deux conditions permettant de trouver la solution particulière sont définies par les valeurs de $y$ en deux points $x_1$ et $x_2$ par $y(x_1) = y_1$ et $y(x_2) = y_2$. Syntaxe de la commande :
`bc2(solution, x=x1, y=y1, x=x2, y=y2)`

Trouver la solution particulière de $x^2\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + 2y = x^3$ avec $y(1) = 1$ et $y(2) = 4$.

**Commande Maxima :**

```
eq: x^2*'diff(y,x,2) - 2*x*'diff(y,x) + 2*y = x^3;
sol_gen: ode2(eq,y,x);
sol_bord: bc2(sol_gen, x=1, y=1, x=2, y=4);
```

**Résultat :**

(% i29) eq : x^ 2*'diff(y, x, 2) - 2*x*'diff(y, x) + 2*y = x^ 3 ;y(1)=1 ;y(2)=4 ;
sol_gen : ode2(eq, y, x) ;
sol_bord : bc2(sol_gen, x=1, y=1, x=2, y=4) ;

$$x^2\left(\frac{d^2}{dx^2}y\right)-2x\left(\frac{d}{dx}y\right)+2y=x^3 \qquad \text{(eq)}$$

$$y(1)=1 \qquad \text{(\% o26)}$$

$$y(2)=4 \qquad \text{(\% o27)}$$

$$y=\frac{x^3}{2}+\%\text{k1}x^2+\%\text{k2}\,x \qquad \text{(sol\_ gen)}$$

$$y=\frac{x^3}{2}-\frac{x^2}{2}+x \qquad \text{(sol\_ bord)}$$

# 7 Etude de quelques types d'équations différentielles

## 7.1 Equation différentielle directement intégrable : $y'=f(x)$

Résoudre l'équation différentielle $y'=x^2+\cos(x)$

**Commande Maxima :**

```
f(x):=x^2+cos(x);
eq: 'diff(y, x) = f(x);
sol_gen: ode2(eq,y,x);
y=integrate(f(x),x)+%c;
```

*Commentaire :* On définit la fonction $f$ et on résout avec `ode2`. Une autre méthode consiste à intégrer la fonction $f$ directement, sans oublier de rajouter la constante d'intégration.

**Résultat :**

(% i1) f(x) :=x^ 2+cos(x) ;

$$f(x):=x^2+\cos(x) \qquad \text{(\% o1)}$$

(% i3) eq : 'diff(y, x) = f(x) ;
sol_gen : ode2(eq, y, x) ;

$$\frac{d}{dx}y=\cos(x)+x^2 \qquad \text{(eq)}$$

$$y=\sin(x)+\frac{x^3}{3}+\%\text{c} \qquad \text{(sol\_ gen)}$$

(% i4) y=integrate(f(x),x)+%c ;

$$y=\sin(x)+\frac{x^3}{3}+\%\text{c} \qquad \text{(\% o4)}$$

## 7.2 Equation différentielle autonome $y' = f(y)$

Dans ce cas, la variable $x$ n'apparait pas explicitement dans l'équation. On résout classiquement avec `ode2` puis on isole $y$ en résolvant l'équation obtenue en $y$. Il y a souvent nécessité de transformer la réponse pour obtenir une expression simplifiée.

Résoudre l'équation différentielle $y' = y(1-y)$

**Commande Maxima :**

```
eq : 'diff(y,x) = y*(1-y);
sol : ode2(eq, y, x);
sol_simp : logcontract(sol);
sol_exp  : exp(lhs(sol_simp)) = exp(rhs(sol_simp));
sol_y : solve(sol_exp, y);
```

*Commentaire :* Il faut guider Maxima dans les transformations à effectuer. Pour cela, on applique manuellement l'exponentielle à chaque membre de la solution obtenue puis on résout par la commande `solve`.

**Résultat :**

```
(% i44) eq : 'diff(y,x) = y*(1-y);
        sol : ode2(eq, y, x);
        sol_simp : logcontract(sol);
        sol_exp  : exp(lhs(sol_simp)) = exp(rhs(sol_simp));
        sol_y : solve(sol_exp, y);
```

$$\frac{d}{dx}y = (1-y)\ y \qquad \text{(eq)}$$

$$\log(y) - \log(y-1) = x + \%c \qquad \text{(sol)}$$

$$\log\left(\frac{y}{y-1}\right) = x + \%c \qquad \text{(sol\_ simp)}$$

$$\frac{y}{y-1} = \%e^{x+\%c} \qquad \text{(sol\_ exp)}$$

$$\left[y = \frac{\%e^{x+\%c}}{\%e^{x+\%c}-1}\right] \qquad \text{(sol\_ y)}$$

Il est possible de transformer ce résultat en utilisant les propriétés de l'exponentielle et de remplacer $\%e^{x+\%c}$ par $K\%e^x$, où $K$ est une constante arbitraire. Il faut manuellement guider Maxima pour arriver à ce résultat, par exemple avec la commande `subst(K*%e^x,%e^(x+%c),sol_y);`

```
(% i10) subst(K*%e^ x,%e^ (x+%c),sol_y);
```

$$\left[y = \frac{\%e^x K}{\%e^x K - 1}\right] \qquad \text{(\% o10)}$$

## 7.3 Equation différentielle à variables séparables : $y' = f(y)g(x)$

Pour ce type d'équation différentielle, on peut utiliser deux méthodes :

1. *Méthode 1* : on résout avec `ode2` puis on exprime $y$ en fonction de $x$ lorsque cela est possible.
2. *Méthode 2* : on sépare les variables ($y$ dans un membre, $x$ dans l'autre), on intègre chaque membre puis on cherche à exprimer $y$ en fonction de $x$. Dans ce cas, il faut souvent aider Maxima car les commandes classiques ne sont pas adaptées à la situation.

Lors de la résolution de ce type d'équation différentielle, on arrive souvent à obtenir l'égalité entre une expression en $y$ avec une expression en $x$, mais il est souvent impossible d'isoler simplement $y$. On pourra cependant utiliser des méthodes numériques pour étudier la solution obtenue.

**Exemple 1** : Résoudre l'équation différentielle $y' = yx$.

**Commande Maxima :**

```
eq: 'diff(y,x) = y*x;
sol: ode2(eq, y, x);
```

*Commentaire :* On utilise ici la commande `ode2`

**Résultat :**

(% i2) eq : 'diff(y,x) = y*x ;
sol : ode2(eq, y, x) ;

$$\frac{d}{dx}y = x\,y \tag{eq}$$

$$y = \%e^{\frac{x^2}{2}}\,\%c \tag{sol}$$

**Commande Maxima :**

```
eq/y;
sol1:integrate(1/y,y)=integrate(x,x)+K;
solve(sol1,y);
```

*Commentaire :* La résolution se fait par séparation des variables. La commande `eq/y:` permet d'identifier les membres de droite et de gauche à intégrer. On retrouve le résultat moyennant une écriture différente pour la constante d'intégration.

**Résultat :**

(% i3) eq/y ;

$$\frac{\frac{d}{dx}y}{y} = x \tag{\% o3}$$

(% i4) sol1 :integrate(1/y,y)=integrate(x,x)+K ;

$$\log(y) = \frac{x^2}{2} + K \tag{sol1}$$

(% i5) solve(sol1,y) ;

$$\left[y = \%e^{\frac{x^2}{2}+K}\right] \tag{\% o5}$$

**Exemple 2** : Résoudre l'équation différentielle $y' = \dfrac{x}{y^2+1}$

**Commande Maxima :**

```
eq: 'diff(y,x) = x/(1 + y^2);
sol: ode2(eq, y, x);
sol2:solve(sol,y)$sol2[3];
```

*Commentaire :* On obtient une équation cubique en $y^3$ comme résultat. La commande `solve` donne trois racines, dont deux complexes. On isole la racine réelle qui est la troisième obtenue.

**Résultat :**

(% i7) eq : 'diff(y,x) = x/(1 + y^ 2);
sol : ode2(eq, y, x);

$$\frac{d}{dx}y = \frac{x}{y^2+1} \tag{eq}$$

$$\frac{y^3+3y}{3} = \frac{x^2}{2} + \%c \tag{sol}$$

(% i9) sol2 :solve(sol,y)$sol2[3];

$$y = \left(\frac{\sqrt{9x^4+36\%cx^2+36\%c^2+16}}{4} + \frac{6\%c+3x^2}{4}\right)^{\frac{1}{3}} - \frac{1}{\left(\frac{\sqrt{9x^4+36\%cx^2+36\%c^2+16}}{4} + \frac{6\%c+3x^2}{4}\right)^{\frac{1}{3}}} \tag{\% o9}$$

**Commande Maxima :**

```
eq*(y^2+1);
sol3:integrate(y^2+1,y)=integrate(x,x)+K;
```

*Commentaire :* Pour la séparation des variables, on transforme l'équation différentielle et on intègre chaque membre manuellement.

**Résultat :**

(% i10) eq*(y^ 2+1);

$$\left(y^2+1\right)\left(\frac{d}{dx}y\right) = x \tag{\% o10}$$

(% i11) sol3 :integrate(y^ 2+1,y)=integrate(x,x)+K;

$$\frac{y^3}{3} + y = \frac{x^2}{2} + K \tag{sol3}$$

## 7.4 Equation différentielle linéaire $y' = f(x)y + g(x)$

Résoudre l'équation différentielle $y' = xy + x - 1$

**Commande Maxima :**

```
eq: 'diff(y,x) = x*y +x-1;
ode2(eq,y,x);
```

*Commentaire :* La solution s'exprime en fonction de la fonction d'erreur, qui est définie mathématiquement par $\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\,dt$

**Résultat :**

(% i19) eq : 'diff(y,x) = x*y +x-1 ;
ode2(eq,y,x);

$$\frac{d}{dx}y = xy + x - 1 \tag{eq}$$

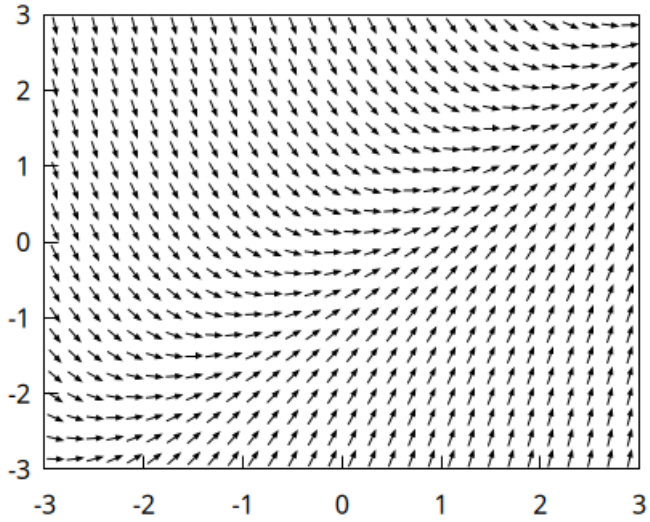
$$y = \%e^{\frac{x^2}{2}}\left(-\left(\frac{\sqrt{\pi}\,erf\left(\frac{x}{\sqrt{2}}\right)}{\sqrt{2}}\right) + \%c - \%e^{-\left(\frac{x^2}{2}\right)}\right) \tag{\% o19}$$

# 8 Champ de direction d'une EDO d'ordre 1

Pour une équation différentielle $y' = f(x, y)$, la dérivée $y'$ représente au point $(x, y)$ la pente de la tangente à la courbe solution passant par ce point. A chaque point $(x, y)$ du plan, on dessine un petit segment de pente $f(x, y)$. Ces segments indiquent la direction que doit suivre toute solution passant par ce point. La représentation graphique de ces segments est le champ de direction de l'EDO. Maxima dispose d'une commande intégrée pour tracer ce champ de direction :
`load(drawdf)$` : chargement du package dédié
`wxdrawdf(f(x,y),[x,-10,10],[y,-10,10]);` représente la champ entre -10 et 10 sur chaque axe.

**Commande Maxima :**
```
load(drawdf)$
wxdrawdf(x - y, [x,-3,3], [y,-3,3])$
```
*Commentaire :* Représentation du champ de direction de $y' = x - y$

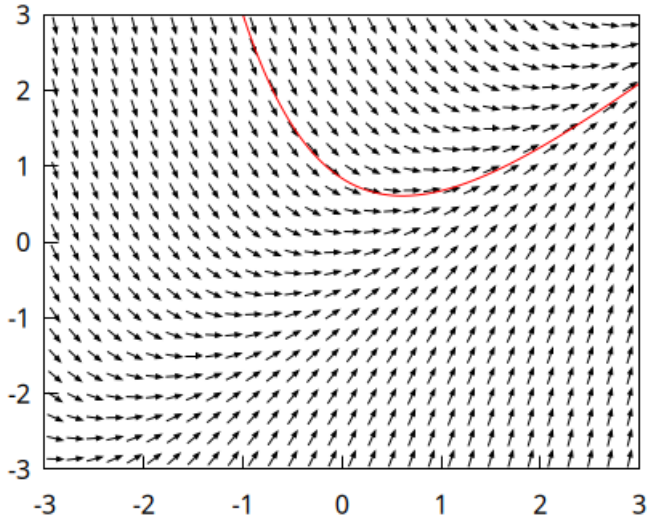
**Résultat :**




On peut facilement rajouter sur ce champ de direction une solution particulière avec l'option `[trajectory_at,a,b]` qui trace la solution vérifiant $y(a) = b$.

**Commande Maxima :**
```
load(drawdf)$
wxdrawdf(x-y,[x,-3,3],[y,-3,3],[trajectory_at,-1,3])$
```

**Résultat :**



# 9 Approximation numérique d'une solution particulière d'une EDO

## 9.1 Méthode d'Euler pour une EDO d'ordre 1

*Principe* : pour une équation différentielle d'ordre 1 $y' = f(x, y)$, on peut obtenir une approximation numérique de la solution définie par la condition initiale $(x_0, y_0)$.

$y'_0 = f(x_0, y_0)$ est la pente de cette solution particulière au point d'abscisse $x_0$. Connaissant le point de départ et la pente, on peut donc approximer la courbe par un segment de droite sur l'intervalle $[x_0, x_0 + \Delta x]$, où $\Delta x$ est une petite valeur. On obtient donc

$$x_1 = x_0 + \Delta x \qquad y_1 = y_0 + y'_0 \times \Delta x$$

On recommence ensuite ce procédé en remplaçant $(x_0, y_0)$ par $(x_1, y_1)$ et ainsi de suite. On construit donc un algorithme qui permet d'obtenir la valeur de $y$ en un point donné $c$. En pratique, on divise l'intervalle $[x_0, c]$ en $n$ sous intervalles d'amplitude $\Delta x$. La précision de l'approximation est liée directement au nombre $n$ choisi et augmente avec lui.

## 9.2 Approximation d'une valeur de la solution particulière

On considère la solution particulière de l'équation différentielle $y' = x - y$ vérifiant $y(0) = 1$. Trouvons une approximation de $y(1)$ en prenant un pas de $\frac{1}{100}$.

**Commande Maxima :**

```
/* Méthode d'Euler pour y' = x - y, y(0) = 1 */
/* Paramètres */
x0: 0$
y0: 1$
h: 1/100$
nsteps: 100$
/* Fonction f(x,y) = x - y */
f(x, y) := x - y$
/* Initialisation */
x: x0$
y: y0$
/* Boucle d'Euler */
for i: 1 thru nsteps do (
    y: y + h * f(x, y),
    x: x + h
)$
/* Résultat : approximation de y(1) */
print("Approximation de y(1) par Euler :", float(y))$
```

**Résultat :**

"Approximation de y(1) par Euler :"

$$0.732064682546459$$

**Commande Maxima :**

```
kill(x, y)$
eq: 'diff(y,x) = x - y;
sol_exacte: ode2(eq, y, x);
sol_ci: ic1(sol_exacte, x=0, y=1);
y_exact: rhs(solve(sol_ci, y)[1]);
print("Valeur exacte y(1) :", ev(y_exact, x=1, numer))$
```

*Commentaire :* Nous comparons avec la solution exacte. Pour cela, on résout avec `ode2`, on trouve la solution particulière avec `ic1` et on calcule la valeur de cette fonction en 1.

**Résultat :**

```
(% i135) /* Comparaison avec la solution exacte */
         kill(x, y)$
         eq : 'diff(y,x) = x - y;
         sol_exacte : ode2(eq, y, x);
         sol_ci : ic1(sol_exacte, x=0, y=1);
         y_exact : rhs(solve(sol_ci, y)[1]);
         print("Valeur exacte y(1) :", ev(y_exact, x=1, numer))$
```

$$\frac{d}{dx}y = x - y \qquad \text{(eq)}$$

$$y = \%e^{-x}\left(\%e^{x}\left(x-1\right)+\%c\right) \qquad \text{(sol\_ exacte)}$$

$$y = \%e^{-x}\left(\%e^{x}x-\%e^{x}+2\right) \qquad \text{(sol\_ ci)}$$

$$\%e^{-x}\left(\%e^{x}x-\%e^{x}+2\right) \qquad \text{(y\_ exact)}$$

"Valeur exacte y(1) :"

$$0.7357588823428847$$

### 9.3 Tracer une courbe approximant une solution particulière

*Principe* : pour tracer sur un intervalle donné une courbe approximant une solution particulière d'une EDO d'ordre 1 par la méthode d'Euler, on subdivise l'intervalle en $n$ parties, et on calcule par la méthode précédente une valeur approchée de la valeur de $y$ solution particulière en chaque point de ces sous-intervalles. Il reste à placer ces points sur un graphique et à les relier.

Tracer une courbe approximant la solution particulière sur $[0, 2\pi]$ de l'équation différentielle

$$y' = \cos(x) + y \quad \text{vérifiant } y(0) = 1$$

On découpera l'intervalle en 100 sous-intervalles.

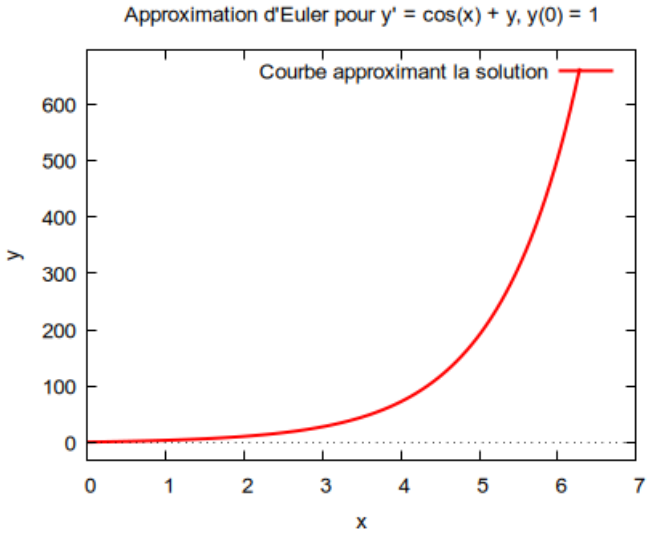
**Commande Maxima :**

```
x0: 0.0$y0: 1.0$x_final: 2.0*%pi$nsteps: 100$
h: float((x_final - x0) / nsteps)$
f(x, y) := float(cos(x) + y)$
x: x0$y: y0$
/* Listes pour stocker les points */
X_list: [x]$
Y_list: [y]$
/* Boucle d'Euler */
for i: 1 thru nsteps do (
    y: y + h * f(x, y),    x: x + h,
    X_list: endcons(x, X_list),
    Y_list: endcons(y, Y_list)
)$
wxplot2d([discrete, X_list, Y_list],
    [style, [lines, 2, red]],
    [xlabel, "x"],
```

```
    [ylabel, "y"],
    [title, "Approximation d'Euler pour y' = cos(x) + y, y(0) = 1"],
    [legend, "Courbe approximant la solution"]
)$
```

*Commentaire :* Afin d'alléger la charge de calcul pour Maxima, on utilise des valeurs approchées des points calculés avec la commande `float`, car le calcul avec les valeurs exactes est très long.

**Résultat :**




## 9.4 Approximation d'une solution avec la commande intégrée rk

Pour une EDO du premier ordre, la commande `rk` intégrée à Maxima applique la méthode de Runge-Kutta classique d'ordre 4 pour calculer numériquement des approximations successives de la solution sur un intervalle donné. Ce dernier est découpé en $n$ sous-intervalles, sur lesquels 4 calculs de la pente sont effectués. Cette méthode est donc plus précise que celle d'Euler présentée précédemment. La syntaxe de cette commande : `rk(equation, fonction, valeur initiale, [variable, début, fin, pas]);`

Trouver une approximation de la solution de $y' = x - y$ avec $y(0) = 1$ sur $[0, 1]$ avec un pas de $0,01$.

**Commande Maxima :**

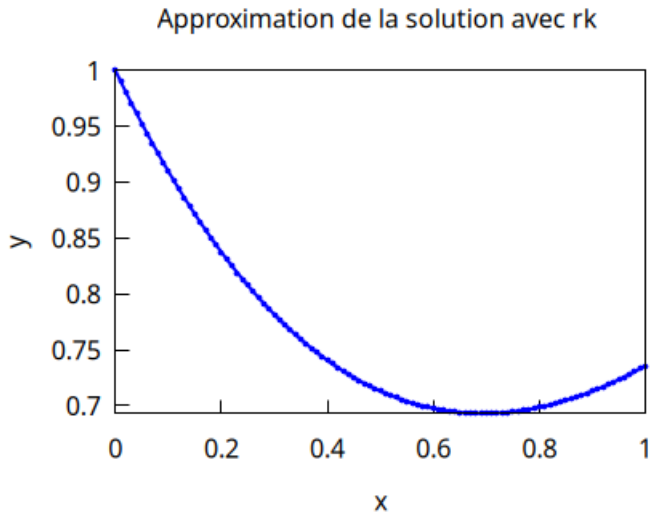
```
rk( x - y,  y, 1,[x, 0, 1, 0.4]);
listepoints:rk( x - y,  y, 1,[x, 0, 1, 0.01])$
wxdraw2d(
    title = "Approximation de la solution avec rk ",
    xlabel = "x",
    ylabel = "y",
    color = blue,
    point_type = filled_circle,
    point_size = 0.3,
    line_width = 2,
    points_joined = true,
    points(listepoints)
)$
```

*Commentaire :* `rk` renvoie une liste de points. La première commande sert à illustrer la liste obtenue dans le cas où le pas est de 0.25. On génère ensuite la liste de points pour le découpage attendu et on trace les points que l'on relie pour obtenir la courbe approximant la solution particulière de l'équation différentielle.

**Résultat :**

(% i19) rk( x - y, y, 1,[x, 0, 1, 0.4]);

$$[[0.0\,,1.0]\,,[0.4\,,0.7408]\,,[0.8\,,0.69887232]\,,[1.0\,,0.7359367307946667]] \qquad (\%\ o19)$$




## 10 La commande intégrée desolve

Il existe une autre commande native de Maxima pour résoudre une équation différentielle. Il s'agit de `desolve` qui utilise la transformation de Laplace pour cette résolution. Cette commande est plus générale que `ode2` car elle permet de résoudre des systèmes d'équations différentielles. Au niveau de la syntaxe de cette commande, les fonctions inconnues doivent être écrites sous forme fonctionnelle ($y(x)$ à la place de y).

```
desolve([eq1, eq2, ...], [y(x), z(x), ...]);
```

**Exemple 1** : Résoudre $y' + y = e^{-x}$ avec $y(0) = 2$.

**Commande Maxima :**

```
de: 'diff(y(x), x) + y(x) = exp(-x);
sol: desolve(de, y(x));
sol_ci: ev(sol, y(0) = 2);
```

*Commentaire :* La solution générale est exprimée en fonction de la valeur initiale en 0. Si la solution particulière est définie par cette valeur initiale en 0, on peut l'obtenir directement en remplaçant $y(0)$ par sa valeur. Sinon, on considère $y(0)$ comme une constante que l'on détermine avec la condition qui est donnée, comme au chapitre 5.

**Résultat :**

(% i11) de : 'diff(y(x), x) + y(x) = exp(-x);
sol : desolve(de, y(x));
sol_ci : ev(sol, y(0) = 2);

$$\frac{d}{dx}\,y(x) + y(x) = \%e^{-x} \qquad (\text{de})$$

$$y(x) = \%e^{-x}x + \%e^{-x}\,y(0) \qquad (\text{sol})$$

$$y(x) = \%e^{-x}x + 2\%e^{-x} \qquad (\text{sol\_ ci})$$

**Exemple 2** : Résoudre $y'' - y = e^x$ avec $y(0) = 1$ et $y'(0) = 0$.
On utilise ici la commande `atvalue` qui permet de définir les valeurs initiales qui seront utilisées par la commande `desolve` pour trouver la solution particulière.
`atvalue(y(x), x=0, 1);` et `atvalue('diff(y(x),x), x=0, 0);`

**Commande Maxima :**

```
atvalue(y(x), x = 0, 1)$
atvalue('diff(y(x), x), x = 0, 0)$
de2: 'diff(y(x), x, 2) - y(x) = exp(x);
sol2: desolve(de, y(x));
```

*Commentaire :* On obtient directement la solution particulière.

**Résultat :**

```
(% i50) atvalue(y(x), x = 0, 1)$
        atvalue('diff(y(x), x), x = 0, 0)$
        de2 : 'diff(y(x), x, 2) - y(x) = exp(x);
        sol2 : desolve(de, y(x));
```

$$\frac{d^2}{dx^2}\,y(x) - y(x) = \%e^x \tag{de2}$$

$$y(x) = \frac{\%e^x x}{2} + \frac{\%e^x}{4} + \frac{3\%e^{-x}}{4} \tag{sol2}$$

**Exemple 3** :
Résoudre le système d'équations différentielles :

$$\begin{cases} \dfrac{dx}{dt} = x + y \\ \dfrac{dy}{dt} = 4x - 2y \end{cases}$$

avec les conditions initiales $x(0) = 1$ et $y(0) = 0$.

**Commande Maxima :**

```
atvalue(x(t), t=0, 1)$
atvalue(y(t), t=0, 0)$
eq1: 'diff(x(t),t) = x(t) + y(t);
eq2: 'diff(y(t),t) = 4*x(t) - 2*y(t);
sol3: desolve([eq1,eq2], [x(t),y(t)]);
```

*Commentaire :* dans cet exemple, $x$ et $y$ désignent des fonctions, tandis que la variable est $t$.

**Résultat :**

```
(% i79) atvalue(x(t), t = 0, 1)$
        atvalue(y(t), t = 0, 0)$
        eq1 : 'diff(x(t), t) = x(t) + y(t);
        eq2 : 'diff(y(t), t) = 4*x(t) - 2*y(t);
        sol3 : desolve([eq1, eq2], [x(t), y(t)]);
```

$$\frac{d}{dt}\,x(t) = y(t) + x(t) \tag{eq1}$$

$$\frac{d}{dt}\,y(t) = 4\,x(t) - 2\,y(t) \tag{eq2}$$

$$\left[x(t) = \frac{4\%e^{2t}}{5} + \frac{\%e^{-(3t)}}{5}, y(t) = \frac{4\%e^{2t}}{5} - \frac{4\%e^{-(3t)}}{5}\right] \tag{sol3}$$

# 11 Pour aller plus loin : le package contrib_ode

Ce package étend les possibilités de `ode2` pour certaines EDO que `ode2` ne sait pas résoudre, notamment des équations du premier ordre non linéaires et certaines équations linéaires homogènes du second ordre à coefficients variables, qui peuvent avoir plusieurs familles de solutions. Pour les EDO du premier ordre, `contrib_ode` commence par appeler `ode2`, puis essaie d'autres méthodes (factorisation, équations de Clairaut, équations de Lagrange, équations de Riccati, équations d'Abel...). Les commandes essentielles de ce package :
`load(contrib_ode);` chargement du package
`contrib_ode(equation, fonction, variable);` cherche à résoudre l'équation différentielle
`ode_check(equation,solution à tester);` teste si la fonction "solution à tester" vérifie ou non l'équation différentielle.

**Exemple 1** : Résoudre $xy'' - (1+xy)y' + y = 0$

**Commande Maxima :**

```
load(contrib_ode)$
eq: x*'diff(y,x)^2 - (1+x*y)*'diff(y,x) + y = 0;
ode2(eq,y,x);
contrib_ode(eq,y,x);
ode_check(eq,[y=log(x)]);
```

*Commentaire :* On teste d'abord `ode2` qui n'arrive pas à résoudre l'équation différentielle, puis la commande `contrib_ode` pour comparer. La dernière commande teste si la fonction log est bien une solution, ce qui est le cas si la réponse est 0.

**Résultat :**

```
(% i29)  eq : x*'diff(y,x)^ 2 - (1 + x*y)*'diff(y,x) + y = 0;
         ode2(eq,y,x);
         contrib_ode(eq,y,x);
         ode_check(eq,[y=log(x)]);
```

$$x\left(\frac{d}{dx}y\right)^2 - (xy+1)\left(\frac{d}{dx}y\right) + y = 0 \qquad \text{(eq)}$$

$$x\left(\frac{d}{dx}y\right)^2 - (xy+1)\left(\frac{d}{dx}y\right) + y = 0 \qquad \text{(\% t27)}$$

"first order equation not linear in y'"

$$false \qquad \text{(\% o27)}$$

$$x\left(\frac{d}{dx}y\right)^2 - (xy+1)\left(\frac{d}{dx}y\right) + y = 0 \qquad \text{(\% t28)}$$

"first order equation not linear in y'"

$$[y = \log(x) + \%c\,,\ y = \%e^{x}\%c] \qquad \text{(\% o28)}$$

$$0 \qquad \text{(\% o29)}$$

**Exemple 2** : Résoudre $y' = y^2 - 2xy + x^2 + 1$

**Commande Maxima :**

```
eq2: 'diff(y,x) = y^2 - 2*x*y + x^2 + 1;
contrib_ode(eq2, y, x);
```

*Commentaire :* Il s'agit d'une équation de Riccati.

**Résultat :**

(% i34) eq2 : 'diff(y,x) = y^ 2 - 2*x*y + x^ 2 + 1 ;
contrib_ode(eq2, y, x) ;

$$\frac{d}{dx}y = y^2 - 2xy + x^2 + 1 \qquad \text{(eq2)}$$

$$\left[\left[x = \frac{1}{\sqrt{\%t-1}} + \%c\,,\, y = x - \sqrt{\%t-1}\right],\left[x = \%c - \frac{1}{\sqrt{\%t-1}}\,,\, y = x + \sqrt{\%t-1}\right]\right] \qquad \text{(\% o34)}$$

Ce package comprend aussi la commande `odelin`, qui permet de résoudre certaines équations linéaires homogènes du second ordre à coefficients variables. Pour la syntaxe :
`odelin(equation, fonction, variable);`

Exemple 3 : Résoudre $(x^2+1)y'' + xy' - y = 0$

**Commande Maxima :**

```
eq3: (x^2 + 1)*'diff(y,x,2) + x*'diff(y,x) - y = 0;
sol3: odelin(eq3, y, x);
```

*Commentaire :* `odelin` renvoie une liste de fonctions formant une base de l'espace des solutions. On génère donc la solution générale par combinaison linéaire des fonctions retournées `odelin`.

**Résultat :**

(% i9) eq3 : (x^ 2 + 1)*'diff(y,x,2) + x*'diff(y,x) - y = 0 ;
sol3 : odelin(eq3, y, x) ;

$$(x^2+1)\left(\frac{d^2}{dx^2}y\right) + x\left(\frac{d}{dx}y\right) - y = 0 \qquad \text{(eq3)}$$

$$\left\{x\,,\,\sqrt{x^2+1}\right\} \qquad \text{(sol3)}$$

**Commande Maxima :**

```
sol4:listify(sol3);
y=%k1*sol4[1]+%k2*sol4[2];
```

*Commentaire :* On transforme le résultat qui est un ensemble en une liste par la commande `listify`, puis on génère la solution générale de l'équation différentielle.

**Résultat :**

(% i13) sol4 :listify(sol3) ;

$$\left[x\,,\,\sqrt{x^2+1}\right] \qquad \text{(sol4)}$$

(% i14) y=%k1*sol4[1]+%k2*sol4[2] ;

$$y = \%k2\sqrt{x^2+1} + \%k1x \qquad \text{(\% o14)}$$

# Index