# *Équations différentielles ordinaires*

*(d'ordre 1 et de degré 1)*

*© Pierre Lantagne*

*Enseignant retraité du Collège de Maisonneuve*

La première version de ce document est parue en janvier 2006. Ce document initie le lecteur à la résolution avec Maple des équations différentielles ordinaires d'ordre 1 et de degré 1. Cette initiation est faite au moyen de certaines macro-commandes de la bibliothèque `DEtools`. Les macro-commandes `DEplot` et `dfieldplot` seront employées pour obtenir des illustrations d'un champ d'éléments de contact ainsi que de certaines solutions particulières. La macro-commande `odeadvisor` de cette bibliothèque sera utilisée pour obtenir le type d'équation différentielle reconnue par l'évaluateur. La table d'environnement `infolevel` est judicieusement utilisée pour obtenir le détail des "efforts" déployés par la macro-commande `dsolve` de la bibliothèque de base de Maple.

Bonne lecture à tous !

* Ce document Maple est exécutable avec la version 2020.2

## Initialisation

```
> restart;
> with(DEtools,DEplot,dfieldplot,odeadvisor):
> with(plottools,disk):
> with(plots,display,implicitplot,odeplot,contourplot,setoptions):
  setoptions(size=[400,400],axesfont=[times,roman,12],labelfont=[times,
  roman,14],titlefont=[times,italic,15],scaling=constrained):
```

## Macro-commande dsolve

### Solution générale

Une équation différentielle est une équation qui comporte des dérivées. L'équation est dite équation différentielle ordinaire (EDO) s'il n'y a qu'une seule variable indépendante et que les dérivées sont exprimées par rapport à cette variable. L'ordre d'une EDO est celui de la dérivée de l'ordre le plus élevé apparaissant dans l'équation et le degré d'une EDO est le degré de la dérivée de l'ordre le plus élevé.

Dans cette feuille Maple, on ne traitera que des équations différentielles ordinaires d'ordre 1 et de degré 1 dont la forme normale est

$$\frac{dy}{dx} = \mathrm{f}(x, y)$$

La forme différentielle suivante d'une équation différentielle ordinaire d'ordre 1 de degré 1

$$M(x, y)\, dx + N(x, y)\, dy = 0$$

est utile pour classifier de telles équations.

La macro-commande dsolve de la bibliothèque de base peut être utile pour trouver de manière analytique la solution générale. (Rappelons que, malgré son nom, la solution générale peut ne pas exprimer toutes les solutions particulières d'une équation différentielle.)

Soit l'équation différentielle suivante à résoudre.

$$x\,dx - y\,dy = 0$$

Pour bien se faire comprendre par l'évaluateur, nous devons lui communiquer l'équation dans une écriture formulée en termes de dérivées plutôt qu'en termes de différentielles. Reformulée, l'équation différentielle à résoudre est donc $x - y\frac{dy}{dx} = 0$. De plus, il sera nécessaire d'utiliser **la syntaxe fonctionnelle** pour signifier à l'évaluateur laquelle des deux variables dans l'équation est désignée comme variable dépendante.

La formule dérivée peut être énoncée soit avec l'opérateur de dérivation `D`, soit avec la macro-commande `diff`.

Soit l'équation différentielle $x - y\frac{dy}{dx} = 0$. Donnons-lui le nom original EDO.

Notons qu'avec la macro-commande `diff`, l'assignation suivante requiert 14 caractères,

```
> EDO:=x-y(x)*diff(y(x),x)=0;
```

$$EDO := x - y(x)\left(\frac{d}{dx}\,y(x)\right) = 0 \qquad \textbf{(2.1.1)}$$

tandis qu'avec l'opérateur `D`, cela en requiert seulement 9...toute une économie!

```
> EDO:=x-y(x)*D(y)(x)=0;
```

$$EDO := x - y(x)\,D(y)(x) = 0 \qquad \textbf{(2.1.2)}$$

La macro-commande odeadvisor de la bibliothèque DEtools donne pour résultat le type d'équation différentielle **reconnue par l'évaluateur**.

```
> odeadvisor(EDO);
```

$$[\_separable] \qquad \textbf{(2.1.3)}$$

À l'aide de la macro-commande `dsolve`, résolvons cette équation différentielle à variables séparables.

```
> Sol:=dsolve(EDO,y(x));
```

$$Sol := y(x) = \sqrt{x^2 + \_C1}\,,\ y(x) = -\sqrt{x^2 + \_C1} \qquad \textbf{(2.1.4)}$$

Par défaut (sous certaines conditions), l'évaluateur s'efforcera de formuler la solution générale de manière explicite. Par contre, il est possible d'imposer ponctuellement à l'afficheur la formulation implicite de la solution générale qui a été trouvée en précisant, en option, l'attribut `implicit`.

```
> Sol:=dsolve(EDO,y(x),implicit);
```

$$Sol := -x^2 + y(x)^2 - \_C1 = 0 \qquad \textbf{(2.1.5)}$$

Utilisons la macro-commande `subs` pour avoir une formulation plus habituelle de la réponse. Reformulons Sol en termes de $y$ et de C.

```
> Sol_générale:=subs([y(x)=y,_C1=C],Sol);
```

$$Sol\_générale := -x^2 + y^2 - C = 0 \qquad \textbf{(2.1.6)}$$

Durant les "efforts" de résolution déployés par l'évaluateur, on peut être tenu au courant des différentes tentatives de `dsolve` dans la recherche d'une méthode pouvant être appliquée pour la résolution analytique de l'équation. Il suffit d'initialiser la variable `infolevel` comme suit:

```
> infolevel[dsolve]:=3;
```

$$infolevel_{dsolve} := 3 \tag{2.1.7}$$

Résolvons de nouveau l'équation EDO pour voir.

```
> Sol:=dsolve(EDO,implicit);
Methods for first order ODEs:
--- Trying classification methods ---
trying a quadrature
trying 1st order linear
trying Bernoulli
<- Bernoulli successful
```

$$Sol := -x^2 + y(x)^2 - \_C1 = 0 \tag{2.1.8}$$

Pour rétablir la "discrétion" de l'évaluateur, il faut taper:

```
> infolevel[dsolve]:=0;
```

$$infolevel_{dsolve} := 0 \tag{2.1.9}$$

Parfois, la résolution explicite peut faire apparaître certaines fonctions analytiques inconnues du niveau collégial. Soit l'équation différentielle à variables séparables $\frac{dy}{dx} = \frac{3y}{2y^2+1}$. La solution générale est très facile à trouver en résolvant « papier-crayon ». Voyons le résultat que donnera l'évaluateur avec une formulation explicite de la solution.

```
> EDO:=Diff(y(x),x)=3*y(x)/(2*y(x)^2+1);
```

$$EDO := \frac{d}{dx} y(x) = \frac{3y(x)}{2y(x)^2+1} \tag{2.1.10}$$

```
> Sol:=dsolve(EDO,y(x));
```

$$Sol := y(x) = e^{-\frac{\text{LambertW}(2\,e^{6\_C1+6x})}{2}+3\_C1+3x} \tag{2.1.11}$$

Résolvons de nouveau mais en spécifiant l'option `implicit`.

```
> Sol:=dsolve(EDO,y(x),implicit);
```

$$Sol := x - \frac{y(x)^2}{3} - \frac{\ln(y(x))}{3} + \_C1 = 0 \tag{2.1.12}$$

Reformulons Sol en termes de $y$ et de C.

```
> Sol_générale:=subs([y(x)=y,_C1=C],Sol);
```

$$Sol\_générale := x - \frac{y^2}{3} - \frac{\ln(y)}{3} + C = 0 \tag{2.1.13}$$

## Solution avec condition initiale

La solution générale d'une équation différentielle d'ordre $n$ contient $n$ constantes arbitraires et indépendantes. Une solution particulière est une solution pour laquelle les constantes sont déterminées généralement à l'aide d'une ou de plusieurs hypothèses sur $y, y', y''$, ... Ces hypothèses sont appelées

**conditions initiales**.

Dans le cas des équations différentielles d'ordre 1 et de degré 1, une solution particulière s'obtient en spécifiant une valeur de la constante $C$. Par exemple, en posant $C = -5$ dans la solution générale $y^2 - x^2 - C = 0$ de l'équation différentielle $x - \frac{y\,dy}{dx} = 0$, on obtient la solution particulière $y^2 - x^2 + 5 = 0$.

Il est possible aussi d'isoler des solutions particulières en précisant des conditions initiales. Par exemple, soit la condition initiale précisant que $y = 2$ quand $x = 3$, c'est-à-dire $y(3) = 2$.

Rappelons-nous l'équation à résoudre.

```
> EDO:=x-y(x)*diff(y(x),x)=0;
```

$$EDO := x - y(x)\left(\frac{d}{dx}\,y(x)\right) = 0 \qquad \textbf{(2.2.1)}$$

Résolvons avec `dsolve`.

```
> Sol:=dsolve(EDO,y(x),implicit);
```

$$Sol := -x^2 + y(x)^2 - \_C1 = 0 \qquad \textbf{(2.2.2)}$$

Reformulons de manière habituelle ce résultat.

```
> Sol_générale:=subs([y(x)=y,_C1=C],Sol);
```

$$Sol\_générale := -x^2 + y^2 - C = 0 \qquad \textbf{(2.2.3)}$$

Obtenons la valeur de la constante $C$ imposée par la condition initiale $y(3) = 2$.

```
> C:=solve(subs([x=3,y=2],Sol_générale),C);
```

$$C := -5 \qquad \textbf{(2.2.4)}$$

Puisque que $C$ pointe maintenant vers la valeur 5, on obtient alors directement la solution particulière correspondante.

```
> Sol_particulière:=Sol_générale;
```

$$Sol\_particulière := -x^2 + y^2 + 5 = 0 \qquad \textbf{(2.2.5)}$$

Rendons à nouveau la variable $C$ libre.

```
> C:='C':
```

La condition initiale $y(3) = 2$ a déterminé $C = -5$. Il y a donc une correspondance entre la condition initiale $y(3) = 2$ et la valeur 5 de la constante $C$.

La macro-commande `dsolve` permet également la résolution d'une équation différentielle avec conditions initiales. Résolvons de nouveau l'équation $x - \frac{y\,dy}{dx} = 0$ mais cette fois-ci, trouvons la solution particulière satisfaisant la condition initiale $y(3) = 2$.

Le premier argument de `dsolve`, entre accolades, devra toujours spécifier l'équation différentielle à résoudre accompagnée de toutes les hypothèses sur $y$ et/ou ses dérivées. (Voir dsolve,ics)

```
> Sol_particulière:=dsolve({EDO,y(3)=2},y(x));
```

$$Sol\_particulière := y(x) = \sqrt{x^2 - 5} \qquad \textbf{(2.2.6)}$$

En résolvant « manu scriptus » cette équation, la solution particulière est plutôt formulée par $y^2 - x^2 + 5 = 0$. Or, l'évaluateur, au lieu de produire la solution particulière implicite $y^2 - x^2 + 5 = 0$, a

donné pour résultat **l'une des formulations explicites de** $y^2 - x^2 - C\_1 = 0$ **satisfaisant la condition initiale** $y(3) = 2$, c'est-à-dire $y = \sqrt{x^2 - 5}$. Les trois conditions initiales **y(-3) = 2, y(3)=-2 et y(-3) = -2** amène également l'une ou l'autre des solutions explicites obtenues de la solution générale avec la constante égale à $-5$.

```
> Sol_particulière_2:=dsolve({EDO,y(-3)=2},y(x));
  Sol_particulière_3:=dsolve({EDO,y(3)=-2},y(x));
  Sol_particulière_4:=dsolve({EDO,y(-3)=-2},y(x));
```

$$Sol\_particulière\_2 := y(x) = \sqrt{x^2 - 5}$$

$$Sol\_particulière\_3 := y(x) = -\sqrt{x^2 - 5}$$

$$Sol\_particulière\_4 := y(x) = -\sqrt{x^2 - 5} \qquad \textbf{(2.2.7)}$$

Par défaut et sous certaines conditions, la macro-commande `dsolve` résoud explicitement. C'est ce qui explique le résultat précédent. En résumé, nous devons donc être circonspect lorsqu'il s'agit d'obtenir une solution particulière avec Maple. Heureusement, que `dsolve` est "sensible" à la variable d'environnement `_EnvExplicit`. Trouvons de nouveau la solution particulière en initialisant au préalable la variable `_EnvExplicit` avec la valeur de vérité `false`.

Initialisons de nouveau la session Maple avec `restart`.

```
> restart;
```

```
> _EnvExplicit:=false:
```

```
> EDO:=x-y(x)*D(y)(x)=0;
```

$$EDO := x - y(x)\,\mathrm{D}(y)(x) = 0 \qquad \textbf{(2.2.8)}$$

```
> Sol_particulière:=dsolve({EDO,y(3)=2},y(x));
```

$$Sol\_particulière := y(x) = RootOf\left(\_Z^2 - x^2 + 5\right) \qquad \textbf{(2.2.9)}$$

En traduisant par $y^2$ la variable $\_Z^2$ créée par Maple, on a donc la solution particulière implicite $y^2 - x^2 + 5 = 0$ formulée en terme de la macro-commande `RootOf`.
Montrons qu'avec la macro-commande `allvalues`, la formulation explicite de l'équation $y(x) = RootOf\left(\_Z^2 - x^2 + 5\right)$ correspond effectivement aux deux formulations explicites de la solution particulière $y^2 - x^2 + 5 = 0$.

```
> allvalues(Sol_particulière);
```

$$y(x) = \sqrt{x^2 - 5}, y(x) = -\sqrt{x^2 - 5} \qquad \textbf{(2.2.10)}$$

La macro-commande odetest de la bibliothèque de base permet la vérification des solutions implicites et explicites qui ont été obtenus avec `dsolve`. Si l'équation est vérifiée, `odetest` donnera comme résultat la valeur 0. Vérifions d'abord la solution particulière implicite.

```
> odetest(Sol_particulière,EDO);
```

$$0 \qquad \textbf{(2.2.11)}$$

Contrôlons maintenant les deux solutions particulières explicitent. À l'aide de la macro-commmande `map`, appliquons `odetest` sur chacune des deux solutions particulières explicites.

```
> map(odetest,[allvalues(Sol_particulière)],EDO);
```

$$[0, 0] \qquad \textbf{(2.2.12)}$$

Ce qui confirme que les deux solutions particulières explicites vérifient l'équation différentielle EDO.

# Interprétation graphique

Comme il y a eu un `restart` précédemment, il nous faut actualiser de nouveau les macro-commandes des différentes bibliothèques.

## Initialisation

```
> with(DEtools,DEplot,dfieldplot,odeadvisor):
  with(plottools,disk):
  with(plots,display,implicitplot,odeplot,contourplot,setoptions):
  setoptions(size=[400,400],axesfont=[times,roman,12],labelfont=
  [times,roman,14],titlefont=[times,italic,15],scaling=constrained):
```

La solution générale d'une équation différentielle correspond en une famille de courbes qui satisfait l'équation et chacune de ces courbes correspond à une solution particulière. Illustrons quelques solutions particulières implicites de l'équation différentielle $x - y\frac{dy}{dx} = 0$.

Afin de nous rappeler la résolution en cours, exécutons de nouveau les trois requêtes suivantes.

```
> EDO:=x-y(x)*D(y)(x)=0;
```

$$EDO := x - y(x)\,\mathrm{D}(y)(x) = 0 \qquad \textbf{(3.1)}$$

On doit résoudre EDO de façon implicite. Initialisons `_EnvExplicit` avec la valeur de vérité `false`.

```
> _EnvExplicit:=false;
```

$$_EnvExplicit := false \qquad \textbf{(3.2)}$$

```
> Sol:=dsolve(EDO);
```

$$Sol := -x^2 + y(x)^2 - \_C1 = 0 \qquad \textbf{(3.3)}$$

Reformulons Sol en termes de $y$ et de C.

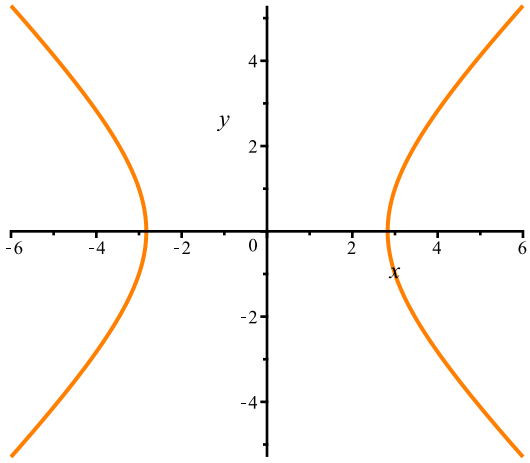
```
> Sol_générale:=subs([y(x)=y,_C1=C],Sol);
```

$$Sol\_générale := -x^2 + y^2 - C = 0 \qquad \textbf{(3.4)}$$

Maintenant, traçons quelques solutions particulières de cette famille avec $\_C1 = -8, -5, 5, 15$.
Puisque chaque solution particulière définit $y$ implicitement comme fonction de $x$, employons la macro-commande `impliciplot` de la bibliothèque `plots`.

Initialisons la première valeur de $C$ et traçons implicitement l'équation Sol_générale.

```
> C:=-8:
  Courbe1:=implicitplot(Sol_générale,x=-6..6,y=-6..6,color=coral):
  Courbe1;
```




Nous pourrions très bien créer les trois autres courbes demandées (avec $C = -5$, 5 et 15 respectivement) de la même que précédemment et réaliser ensuite dans un même graphique la superposition de ces quatre tracés. Mais, pour plus d'efficacité, automatisons cette tâche répétitive. La boucle suivante va faciliter la création des quatre structures graphiques correspondant aux quatre valeurs de $C$ retenues. L'opérateur de concaténation les deux barres verticales « || ».

Créons d'abord la liste des valeurs que la constante $C$ prendra.

```
> Valeurs_de_C:=[-8,-5,5,15];
```

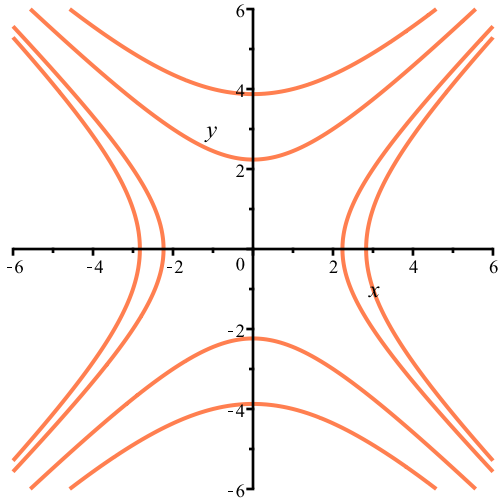
$$Valeurs\_de\_C := [-8, -5, 5, 15] \quad \textbf{(3.5)}$$

Ensuite, avec une boucle, créons les quatre tracés.

```
> for i from 1 to nops(Valeurs_de_C) do
   C:=Valeurs_de_C[i];
   Courbe||i:=implicitplot(Sol_générale,x=-6..6,y=-6..6,grid=[60,60],
  color="HTML 16");
  od:
  C:='C':    # Pour rendre C libre
  i:='i':    # Pour rendre i libre
```

À l'aide de la macro-commande `display` de la bibliothèque `plots`, affichons la superposition de ces quatres tracés.

```
> Famille:=display(Courbe||(1..nops(Valeurs_de_C))):
  Famille;
```

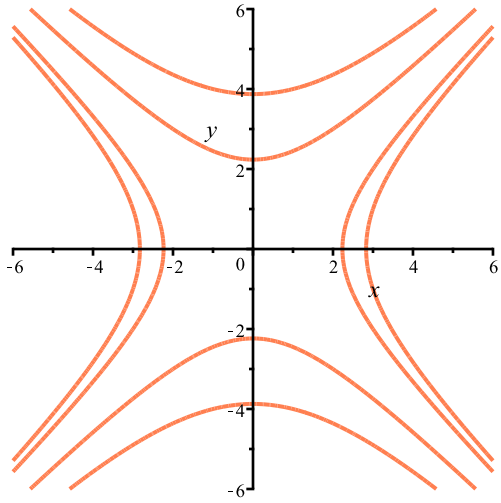



Nous pouvons tout aussi bien employer la macro-commande `contourplot` de la bibliothèque `plots` pour réaliser plus directement la superposition de ces quatres tracés. En effet, la solution générale peut, d'un certain point de vue, être considérée comme une fonction F de deux variables définit par $F(x,y) = y^2 - x^2$.

```
> Formule:=solve(Sol_générale,C);
```

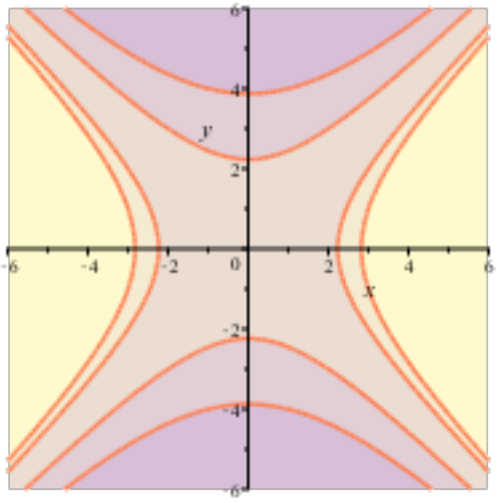
$$Formule := -x^2 + y^2 \qquad \textbf{(3.6)}$$

```
> Famille:=contourplot(Formule,x=-6..6,y=-6..6,grid=[70,70],
                       contours=Valeurs_de_C,color="HTML 16",
  thickness=1):
> display(Famille);
```




Si vous êtes un tantinet patient, vous pouvez même mettre un peu de couleurs à l'aide des options `coloring` et `filled`.

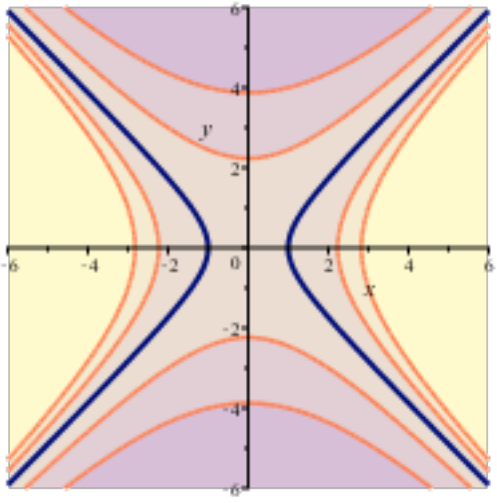
```
> Famille:=contourplot(Formule,x=-6..6,y=-6..6,grid=[60,60],
                       contours=Valeurs_de_C,color="HTML 16",coloring=
  ["HTML 63","HTML 133"],filled=true):
> display(Famille);
```




Pour mettre en évidence une solution particulière, par exemple correspondant à $C = -1$, il suffit de tracer séparément cette solution particulière puis de superposer son tracé avec celui de la famille.

```
> C:=-1:
  Sol_particulière:=implicitplot(Sol_générale,x=-6..6,y=-6..6,
                    color="Niagara 2",thickness=2,grid=[60,60]):
  C:='C':   # Pour rendre C libre
```

```
> display([Sol_particulière,Famille]);
```




# Champ d'éléments de contact

Une équation différentielle ordinaire du premier ordre et du premier degré est une équation de la forme $\frac{dy}{dx} = \mathrm{f}(x, y)$. En se rappelant l'interprétation graphique de $\frac{dy}{dx}$, f(x,y) est donc une formule donnant la pente de la tangente à la courbe solution passant par le point $(x, y)$. Ainsi, l'équation différentielle $x - y\frac{dy}{dx} = 0$ (de manière équivalente

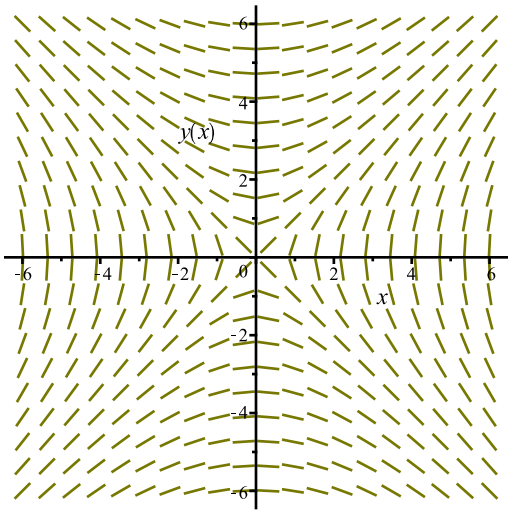
$\frac{dy}{dx} = \frac{x}{y}$ ) peut être visualisée par un graphique appelé ***champ d'éléments de contact***.

Chaque élément de ce champ est un petit segment de droite centré au point $(x_0, y_0)$ d'orientation f $(x_0, y)$.

Ainsi, pour l'équation différentielle $\frac{dy}{dx} = \frac{x}{y}$, au point $(2, 1)$, la pente de la tangente à la courbe de la solution particulière passant par ce point est $\frac{dy}{dx} = \frac{2}{1}$. Au point $(2, 2)$ la pente de la tangente à la courbe de la solution particulière passant par ce point est $1$. Dans le premier cas, on illustre le résultat par un petit segment de droite centré en $(2, 1)$ de pente $2$ et, dans le second cas, par un autre segment de droite centré en $(2, 2)$ de pente $1$. En répétant ce processus avec un certain quadrillage de couples $(x, y)$ régulièrement espacé, on obtient ce qu'on appelle un *champ d'éléments de contact*, parfois traduit littéralement de l'anglais par champ de pentes.

La macro-commande dfieldplot de la bibliothèque `DEtools` permet le tracé d'un champ d'éléments de contact. La bibliothèque `DEtools` impose la formulation fonctionnelle de la variable dépendante. Dans le cas d'une fonction de deux variables $x$ et $y$, si la variable $y$ est désignée comme dépendante, on doit donc l'énoncer dans les requêtes avec la syntaxe fonctionnelle $y(x)$.

```
> Champ:=dfieldplot(EDO,[y(x)],x=-6..6,y=-6..6,
        arrows=line,dirgrid=[20,20],color="Niagara 16"):
> Champ;
```




On peut également représenter dans un champ d'éléments de contact, des courbes appelées **isoclines** ( du grec *iso* qui signifie même et du latin *clinare* qui signifie pencher). Les isoclines sont des courbes le long desquelles les éléments de contact ont une direction ( une inclinaison ) donnée.

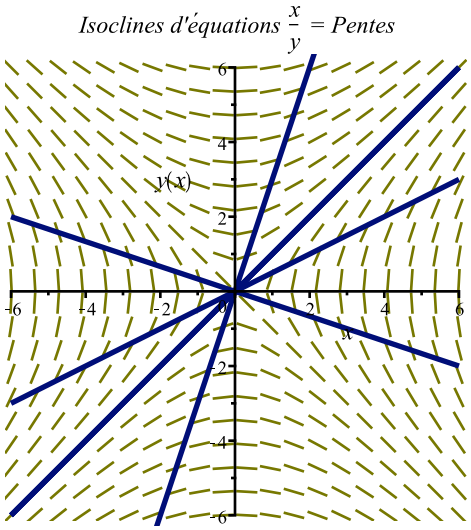
Superposons, dans le champ d'éléments de contact précédent, les isoclines de pente $-3$, $\frac{1}{3}$ et $2$.

Pour obtenir de meilleurs tracés, traçons plutôt les isoclines avec la macro-commande `plot` en traçant $y = \frac{x}{Pentes}$ au lieu de tracer des équations de la forme $Pentes = \frac{x}{y}$ avec la macro-commande `implicitplot` car $y$ peut être explicitée de manière unique en termes de $x$.

```
> Pentes:=[-3,1/3,2];
```

$$Pentes := \left[-3, \frac{1}{3}, 2\right] \quad \textbf{(4.1)}$$

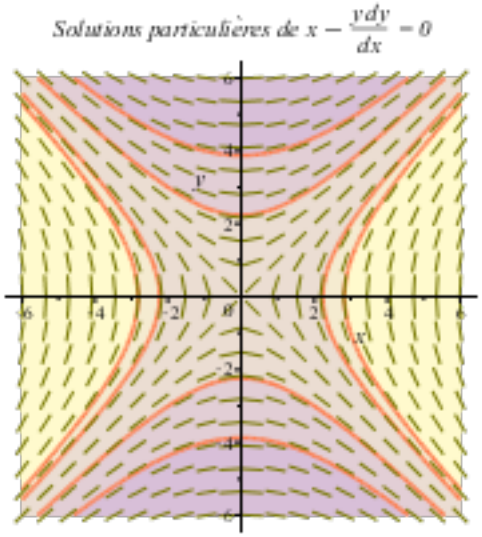
```
> for i from 1 to nops(Pentes) do
   isocline||i:=plot([x,x/Pentes[i]],x=-6..6,thickness=2,color="Niagara
  2"):
  od:
  i:='i':   # Pour rendre i libre
> display({isocline||(1..nops(Pentes)),Champ},view=[-6..6,-6..6],title=
  typeset("Isoclines d'équations ", x/y, " = Pentes"));
```




L'intérêt d'un champ d'éléments de contact apparaît clairement avec l'utilisation de l'ordinateur. Que l'équation différentielle possède ou non une solution analytique, ce type de tracé permet de visualiser les trajectoires des courbes solutions d'une équation différentielle.

Superposons au champ d'éléments de contact précédent, les solutions particulières correspondant aux conditions initiales donnant C = −8, −5, 5 et 15.

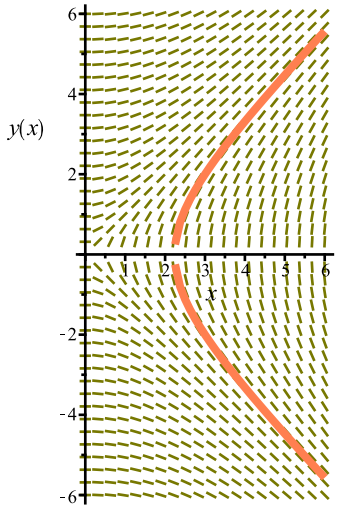
```
> display([Famille,Champ],title=typeset("Solutions particulières de ",
  x-y*dy/dx = 0));
```



La macro-commande DEplot de la bibliothèque `DEtools` permet automatiquement la superposition du tracé d'un champ d'éléments de contact avec les tracés des solutions particulières.
**Attention**: `Deplot` résoud explicitement même si la variable d'environnement `_EnvExplicit` est initialisée à `false`. Dans le cas où la solution générale explicite n'est pas unique, le traçage des solutions particulières doit être limité aux quadrants respectifs spécifiés par les conditions initiales. Ainsi, pour la solution particulière correspondant à la condition initiale $y(3) = 2$ et $y(3) = -2$, nous devons faire la superposition de deux tracés.

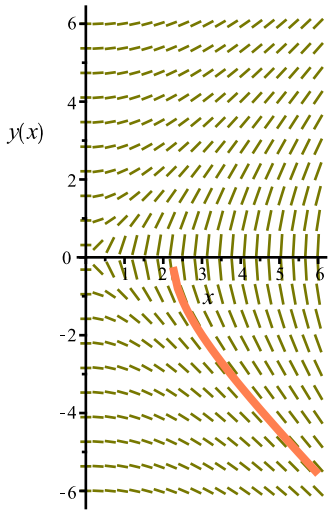
```
> C1:=DEplot(EDO,y(x),x=0..6,[[y(3)=2]],y=0..6,linecolor="HTML 16",
  color="Niagara 16",arrows=line,dirgrid=[20,20]):
  C2:=DEplot(EDO,y(x),x=0..6,[[y(3)=-2]],y=-6..0,linecolor="HTML 16",
  color="Niagara 16",arrows=line,dirgrid=[20,20]):
Warning, plot may be incomplete, the following errors(s) were issued:
   cannot evaluate the solution further left of 2.2360679, probably a singularity
Warning, plot may be incomplete, the following errors(s) were issued:
   cannot evaluate the solution further left of 2.2360679, probably a singularity
```

```
> display([C1,C2]);
```




Si on essait de le faire d'un seul jet, c'est-à-dire sans se limiter au seul quadrant correspondant à la condition initiale, voici ce que cela donne.

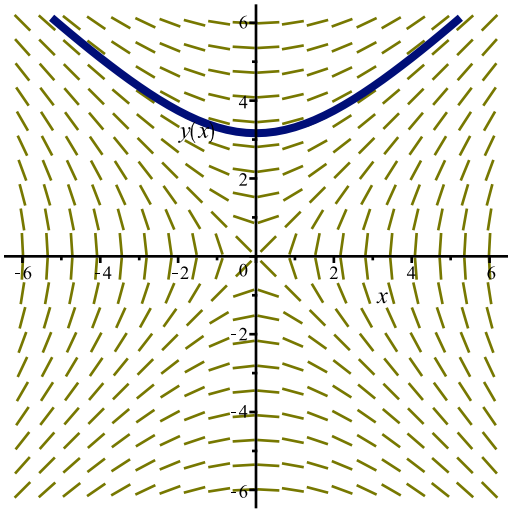
```
> DEplot(EDO,y(x),x=0..6,[[y(3)=-2]],y=-6..6,linecolor="HTML 16",
  color="Niagara 16",arrows=line,dirgrid=[20,20]);
Warning, plot may be incomplete, the following errors(s) were issued:
   cannot evaluate the solution further left of 2.2360679, probably a singularity
```




Horrible!

De plus, dans un tel cas, lorsque la condition initiale est à l'origine, `Deplot` ne pourra représenter correctement la solution particulière dans un champ d'éléments de contact que d'une partie évidemment de la solution générale.

```
> DEplot(EDO,y(x),x=-6..6,[[y(0)=sqrt(10)]],y=-6..6,linecolor="Niagara
  2",color="Niagara 16",arrows=line,dirgrid=[20,20]);
```




En résumé, pour illustrer dans un champ d'éléments de contact des solutions particulières lorsque la solution générale explicite n'est pas unique, il vaut mieux créer séparément les objets *champ* et *tracés* des solutions particulières puis superposer le tout sans utiliser `DEplot`.

Dans le cas où la solution générale explicite correspond à une seule solution, la macro-commande `DEplot` permet de tracer correctement les solutions particulières, que les conditions initiales soient à l'origine ou non.

Soit l'équation différentielle $y' = -2x - y$.

```
> EDO:=D(y)(x)=-2*x-y(x);
  Sol:=dsolve(EDO);
```

$$EDO := \mathrm{D}(y)(x) = -2x - y(x)$$

$$Sol := y(x) = -2x + 2 + e^{-x}\_C1 \quad \textbf{(4.2)}$$

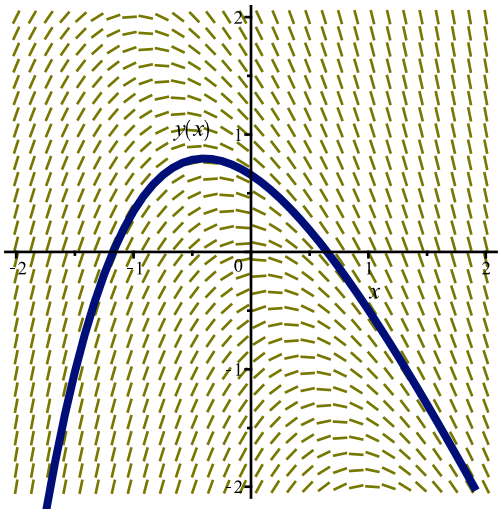
La solution générale explicite de cette équation est donc unique.

```
> Sol_générale:=subs([y(x)=y,_C1=C],Sol);
```

$$Sol\_générale := y = -2x + 2 + e^{-x} C \quad \textbf{(4.3)}$$

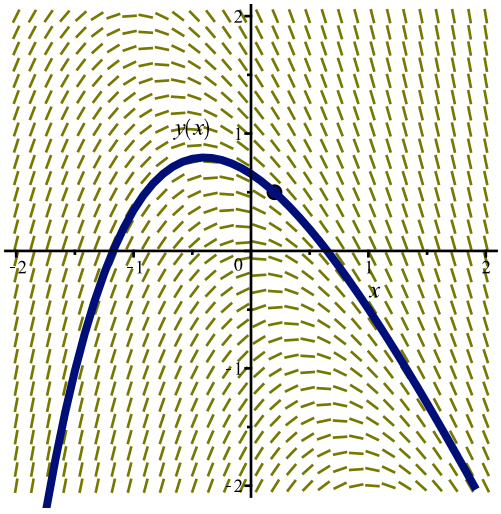
Traçons, avec `DEplot`, la solution particulière correspondant à la condition initiale $y\left(\frac{1}{5}\right) = \frac{1}{2}$.

```
> Graphique:=DEplot(EDO,y(x),x=-2..2,[[y(1/5)=1/2]],y=-2..2,
  linecolor="Niagara 2",color="Niagara 16",arrows=line,dirgrid=[30,30])
  :
  Graphique;
```




Superposons dans ce graphique le tracé du point $\left(\frac{1}{5}, \frac{1}{2}\right)$. Utilisons la macro-commande `disk` de la bibliothèque `plottools`.

```
> P:=disk([1/5,1/2],0.05,color="Niagara 2"):
  display(Graphique,P);
```

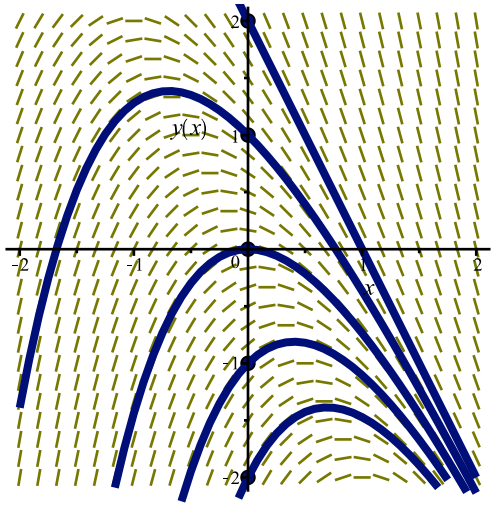


Bien sûr, les tracés de solutions particulières correspondant à des solutions initiales à l'origine seront correctement rendus.

```
> P1:=disk([0,-2],0.05,color="Niagara 2"):
  P2:=disk([0,-1],0.05,color="Niagara 2"):
  P3:=disk([0,0],0.05,color="Niagara 2"):
  P4:=disk([0,1],0.05,color="Niagara 2"):
  P5:=disk([0,2],0.05,color="Niagara 2"):
```

```
> Graphique:=DEplot(EDO,y(x),x=-2..2,[[y(0)=-2],[y(0)=-1],[y(0)=0],[y
  (0)=1],[y(0)=2]],y=-2..2,linecolor="Niagara 2",color="Niagara 16",
  arrows=line,dirgrid=[25,25]):
```

```
> display([Graphique,P||(1..5)]);
```




Mais le tracé manuel des solutions particulières avec la macro-commande `plot` donne des tracés de meilleures qualités. En effet, puisque la solution générale explicite est unique, on peut utiliser efficacement la macro-commande `plot`.

```
> Champ:=dfieldplot(EDO,[y(x)],
  x=-2..2,y=-2..2,arrows=line,dirgrid=[25,25],color="Niagara 16",
  dirgrid=[30,30]):
```

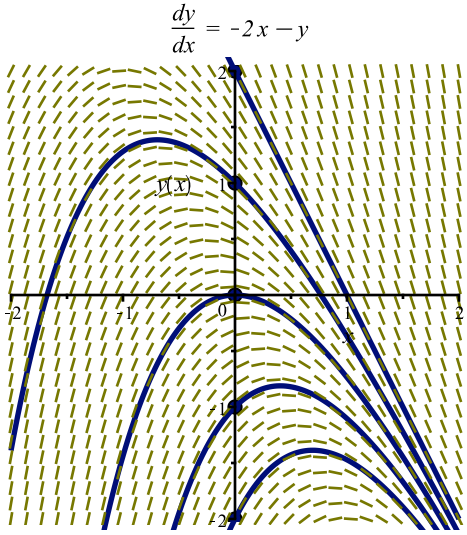
Superposons au champ d'éléments de contact précédent, les solutions particulières correspondant aux conditions initiales $y(0) = C$ pour C = $-2,\ -1,\ 0,\ 1$ et 2.

```
> Valeurs:=[-4,-3,-2,-1,0];
```

$$Valeurs := [-4, -3, -2, -1, 0] \qquad \textbf{(4.4)}$$

```
> for i from 1 to nops(Valeurs) do
   C:=Valeurs[i];
   Courbe||i:=plot([x,rhs(Sol_générale),x=-2..2],color="Niagara 2",
  thickness=2):
  od:
  C:='C':  # Pour rendre C libre
  i:='i':  # Pour rendre i libre
```

```
> display([Courbe||(1..nops(Valeurs)),Champ,P||(1..5)],view=[-2..2,-2.
  .2],title=typeset(dy/dx = -2*x-y));
```




# Résolution numérique

Soit l'équation différentielle $\frac{dy}{dx} = \cos(xy)$.

```
> EDO:=diff(y(x),x)=cos(x*y(x));
```

$$EDO := \frac{d}{dx} y(x) = \cos(x y(x)) \qquad \textbf{(5.1)}$$

Utilisons la macro-commande `dsolve` pour résoudre cette équation.

```
> dsolve(EDO,y(x));
```

Aucun résultat n'est apparu. Résolvons de nouveau cette équation mais en suivant l'évaluateur dans sa recherche d'un méthode de résolution.

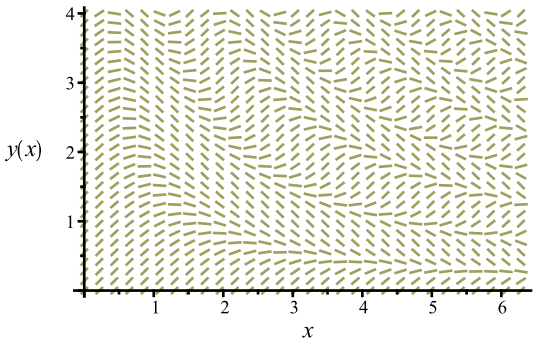
```
> infolevel[dsolve]:=3:
> dsolve(EDO,y(x));
Methods for first order ODEs:
--- Trying classification methods ---
trying a quadrature
trying 1st order linear
trying Bernoulli
trying separable
trying inverse linear
trying homogeneous types:
trying Chini
differential order: 1; looking for linear symmetries
trying exact
Looking for potential symmetries
trying inverse_Riccati
trying an equivalence to an Abel ODE
differential order: 1; trying a linearization to 2nd order
--- trying a change of variables {x -> y(x), y(x) -> x}
differential order: 1; trying a linearization to 2nd order
trying 1st order ODE linearizable_by_differentiation
--- Trying Lie symmetry methods, 1st order ---
 -> Computing symmetries using: way = 3
 -> Computing symmetries using: way = 4
 -> Computing symmetries using: way = 5
```

```
trying symmetry patterns for 1st order ODEs
-> trying a symmetry pattern of the form [F(x)*G(y), 0]
-> trying a symmetry pattern of the form [0, F(x)*G(y)]
-> trying symmetry patterns of the forms [F(x),G(y)] and [G(y),F(x)]
-> trying a symmetry pattern of the form [F(x),G(x)]
-> trying a symmetry pattern of the form [F(y),G(y)]
-> trying a symmetry pattern of the form [F(x)+G(y), 0]
-> trying a symmetry pattern of the form [0, F(x)+G(y)]
-> trying a symmetry pattern of the form [F(x),G(x)*y+H(x)]
-> trying a symmetry pattern of conformal type
```

Comme on le voit, l'évaluateur a essayé plusieurs méthodes analytiques de résolution. Ces méthodes se sont avérées vaines. L'évaluateur connaît, bien sûr, d'autres techniques plus avancées de résolution qui passent par des dévelopements en séries, par des transformations de Laplace. L'aide de `dsolve` donne de l'information sur ce sujet mais, en ce qui nous concerne, nous n'allons pas élaborer là-dessus.

Quoique nous n'avons pu obtenir une solution générale analytique, cela ne veut pas dire pour autant qu'on ne peut pas obtenir de solutions particulières. En effet, visualisons les trajectoires de ces solutions particulières avec `dfieldplot`.

```
> Champ:=dfieldplot(EDO,[y(x)],x=0..2*Pi,y=0..4,color=khaki,arrows=
  LINE,dirgrid=[30,30]):
  Champ;
```




L'évaluateur a la capacité d'employer certaines méthodes numériques de résolution pouvant donner de bonnes approximations des solutions avec conditions initiales. La feuille Maple intitulée « Équations différentielles et méthodes numériques », que vous pouvez retrouver sur mon site Internet, élabore sur quelques méthodes numériques élémentaires. Sans donner de détails ici, voyons comment, avec dsolve, obtenir numériquement une courbe solution particulière.

En précisant, en option, l'attribut « **numeric** », le résultat sera une procédure (une fonction) de calcul pour $y$ dans la courbe solution de l'équation différentielle correspondant à la valeur de $x$. Donnons le nom *Points_particuliers* à cette procédure.

```
> infolevel[dsolve]:=0;
```

$$infolevel_{dsolve} := 0 \qquad \textbf{(5.2)}$$

```
> Points_particuliers:=dsolve({EDO,y(1)=2},y(x),numeric);
```

$$Points\_particuliers := \textbf{proc}(x\_rkf45) \ \ldots \ \textbf{end proc} \qquad \textbf{(5.3)}$$

Ainsi, Points_particuliers (t) donnera comme résultat le point dont les coordonnées sont $(t, y(t))$ de la solution particulière isolée par la condition initiale $y(1) = 2$.

```
> seq(lprint('Points_particuliers'(k)=Points_particuliers(k)),k=0..6);
Points_particuliers(0) = [x = 0., y(x) = HFloat(1.54862082936198409)]
Points_particuliers(1) = [x = HFloat(1.), y(x) = HFloat(2.)]
Points_particuliers(2) = [x = 2., y(x) = HFloat(1.25701489993524107)]
Points_particuliers(3) = [x = 3., y(x) = HFloat(.649599018600757971)]
Points_particuliers(4) = [x = 4., y(x) = HFloat(.425973821191194424)]
Points_particuliers(5) = [x = 5., y(x) = HFloat(.328715526879530151)]
Points_particuliers(6) = [x = 6., y(x) = HFloat(.269788295416576285)]
```

Reste donc à générer un certain nombre de points pour le tracé de cette solution particulière que nous superposerons ensuite dans un champ d'éléments de contact pour cette équation différentielle. Au lieu de le faire manuellement, la macro-commande `odeplot` de la bibliothèque `plots` est plus efficace dans ce cas.

```
> Sol_Particulière:=odeplot(Points_particuliers,[x,y(x)],0..2*Pi,color=
  "Niagara 16",grid=[30,30],thickness=2):
> display([Champ,Sol_Particulière]);
```