Premiers pas en Maple (Partie 3 de 5)

© Pierre Lantagne
Enseignant retraité du Collège de Maisonneuve

Pour voir le contenu des différentes sections, cliquer avec la souris sur le triangle ▶ précédant le titre. La section se déploiera ▼ et son contenu sera affiché.
(un clic gauche ou un clic droit sur le triangle permettra de rétracter la section et son contenu sera alors masqué).

Pour votre confort, vous pouvez ajuster la taille de l'affichage à l'aide de la commande *Facteur de zoom* du menu *Affichage*.

Bonne lecture à tous !

* Ce document Maple est exécutable avec la version 2020.2

Initialisation

```
> restart;
```

```
> with(plots, display, implicitplot);
  with(algcurves, plot_real_curve);
```

[*display*, *implicitplot*]

[*plot_real_curve*] **(1.1)**

Objectif

Cette troisième partie a comme objectif principal de présenter à l'étudiant la syntaxe paramétrique pour tracer le graphe d'une fonction réelle d'une variable réelle. L'élève sera initié à la macro-commande `display` de la bibliothèque `plots` afin de réaliser une superposition de tracés dans un même graphique. Ce document présente également la manière de s'y prendre pour obtenir le tracé de fonctions définies implicitement à l'aide de la macro-commande `implicitplot` de la bibliothèque `plots` et à l'aide de la macro-commande `plot_real_curve` de la bibilothèque `algcurves`.

Déroulement

Avant même d'exécuter les requêtes, l'élève doit faire d'abord une lecture attentive de la présentation des différentes transpositions en `Maple` des notions mathématiques. Pour profiter au maximum de la présentation des différents éléments fondamentaux présentés dans ce document, il ne faut surtout pas être un lecteur passif. Au contraire, au fil du déroulement de cette feuille `Maple`, l'élève devra, de plus en plus, anticiper le résultat qui sera affiché et de se convaincre, à chaque fois, que le résultat obtenu est le résultat attendu.

Syntaxe paramétrique de tracés de courbes dans le plan

Il y a trois syntaxes que l'on peut utiliser pour faire un tracé. La troisième est la plus efficace et elle est appelée syntaxe paramétrique.

- `plot(Nom de la fonction,a..b,options)` soit `plot(f,a..b,options)`
- `plot(Formule de calcul,x=a..b,options)` soit `plot(f(x),x=a..b,options)`
- **`plot([x,Formule de calcul,x=a..b],options)`** soit **`plot([x,f(x),x=a..b],options)`**

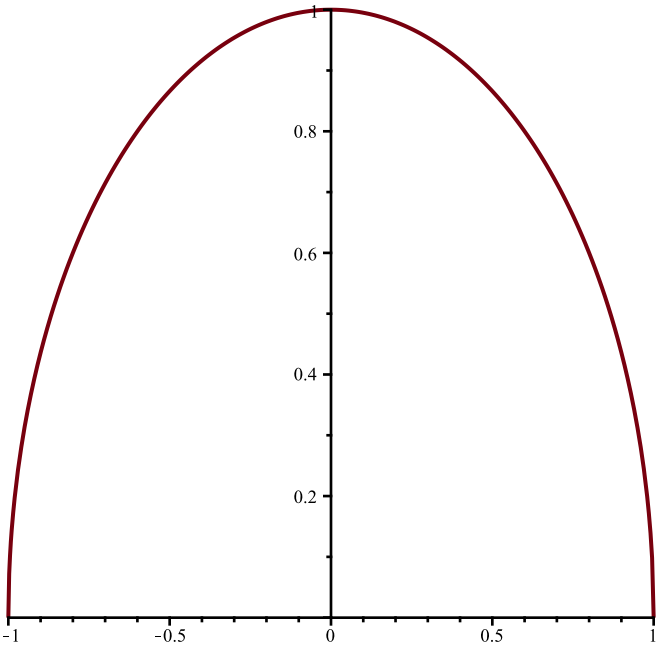
La syntaxe paramétrique est la syntaxe qui sera retenue pour les tracés de graphiques.

Voir options de la macro-commande **`plot`** pour prendre connaissance des options qui peuvent être spécifiées dans la macro-commande `plot`.

```
> f:=x->sqrt(1-x^2);
```

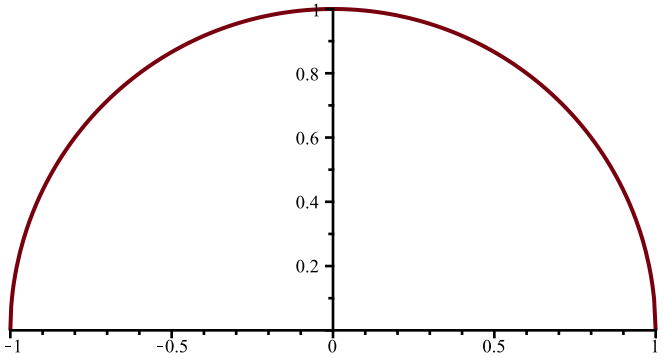
$$f := x \mapsto \sqrt{1 - x^2} \tag{4.1}$$

```
> plot([x,f(x),x=-1..1]);
```



Le résultat attendu est un demi-cercle. Forçons Maple à utiliser la même échelle pour les deux axes de coordonnées en spécifant l'option `scale=constrained`.

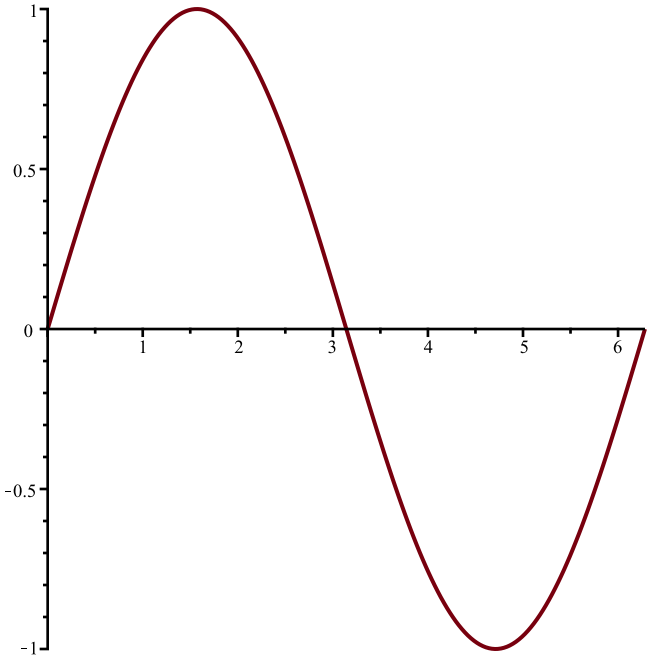
```
> plot([x,f(x),x=-1..1],scaling=constrained);
```



Voilà ! C'est beaucoup mieux ainsi mais parfois, il sera plutôt préférable de ne pas avoir la même échelle sur les deux axes pour mieux mettre en évidence les caractéristiques du graphique.

On peut, bien sûr, préciser directement la formule de calcul des images dans la macro-commande `plot`.

```
> plot([x,sin(x),x=0..2*Pi]);
```



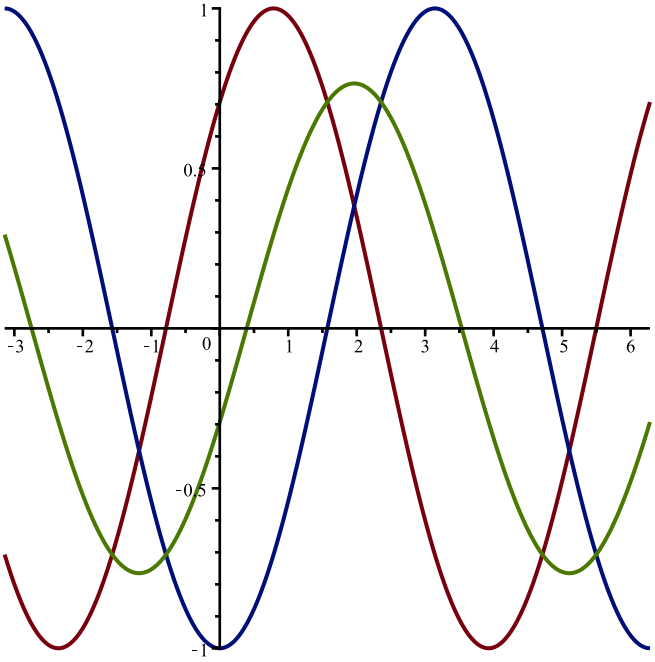
Il est possible de superposer plusieurs tracés dans un même graphique. Voyons comment faire.

```
> f:=x->sin(x+Pi/4);
  g:=x->cos(x+Pi);
```

$$f := x \mapsto \sin\left(x + \frac{\pi}{4}\right)$$

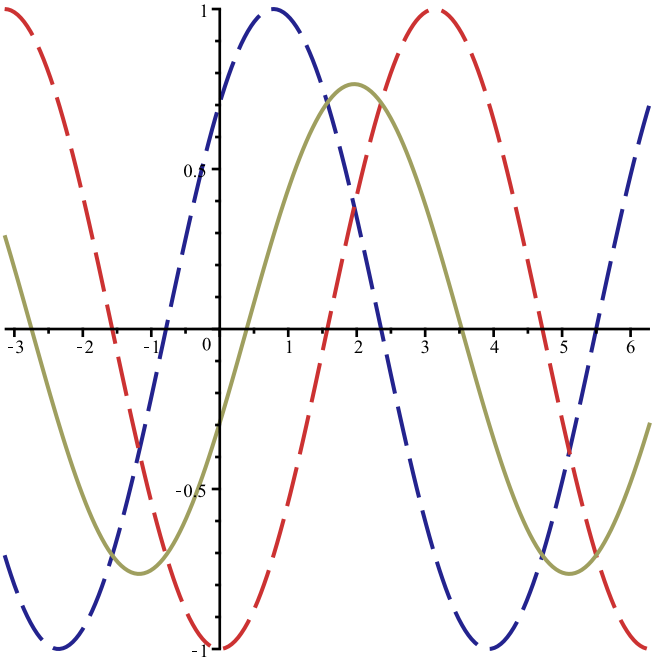
$$g := x \mapsto \cos(x + \pi) \qquad \textbf{(4.2)}$$

```
> plot([ [x,f(x),x=-Pi..2*Pi],
       [x,g(x),x=-Pi..2*Pi],
       [x,(f+g)(x),x=-Pi..2*Pi] ]);
```



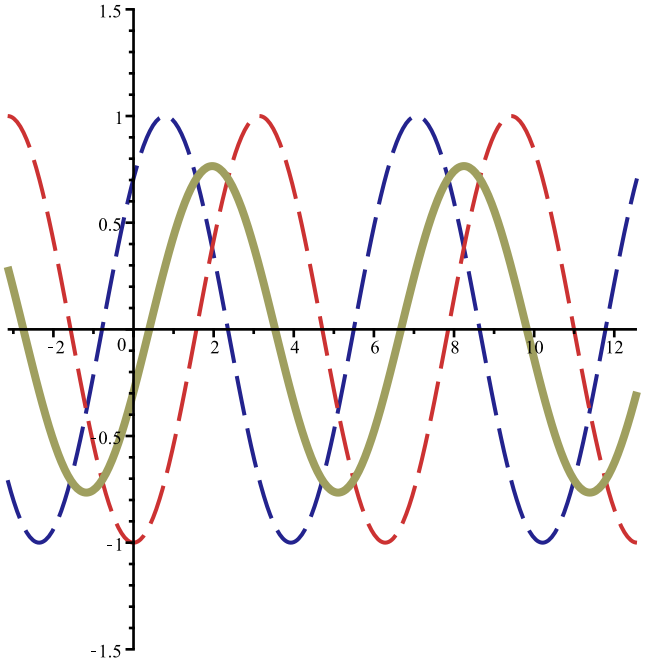
Contrôlons la couleur et le type de trait de chacun des tracés.

```
> plot([ [x,f(x),x=-Pi..2*Pi],
       [x,g(x),x=-Pi..2*Pi],
       [x,(f+g)(x),x=-Pi..2*Pi] ],
       color=[navy,orange,khaki],
       linestyle=[3,3,1]);
```




```
> plot([ [x,f(x),x=-Pi..4*Pi],
       [x,g(x),x=-Pi..4*Pi],
       [x,(f+g)(x),x=-Pi..4*Pi] ],
```

```
color=[navy,orange,khaki],
linestyle=[3,3,1],
thickness=[1,1,3],
view=[-Pi..4*Pi,-1.5..1.5]);
```




```
>
```

L'extension plots

L'extension plots de Maple 2020.2 est une bibliothèque de macro-commandes spécialisées pour la création et la superposition de structures graphiques.

```
> with(plots);
```

[*animate*, *animate3d*, *animatecurve*, *arrow*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *conformal3d*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *densityplot*, *display*, *dualaxisplot*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *interactive*, *interactiveparams*, *intersectplot*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *multiple*, *odeplot*, *pareto*, *plotcompare*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedra_supported*, *polyhedraplot*, *rootlocus*, *semilogplot*, *setcolors*, *setoptions*, *setoptions3d*, *shadebetween*, *spacecurve*, *sparsematrixplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*] **(5.1)**

La macro-commande `with` donne comme résultat la liste de toutes les macro-commandes de l'extension. Obtenons le nombre d'éléments de cette liste avec la macro-commande nops.

```
> nops((5.1));
```

57 **(5.2)**

La macro-commande `display` de cette bibliothèque est l'une de ces 57 macro-commandes utiles pour la superposition de structure graphiques déjà créées. À l'usage, on verra que cette macro-commande permet de

mieux contrôler la superposition de structures graphiques.

Si on prévoie n'utiliser que certaines macro-commandes d'une extension, il suffit de préciser dans la macro-commande `with` la liste de ces macro-commandes après le nom de la bibliothèque. C'est ce qui a été fait au début de ce document à la section Initialisation:

```
> restart;
> with(plots,display);
```

Créeons quelques structures graphiques que nous allons superposer.

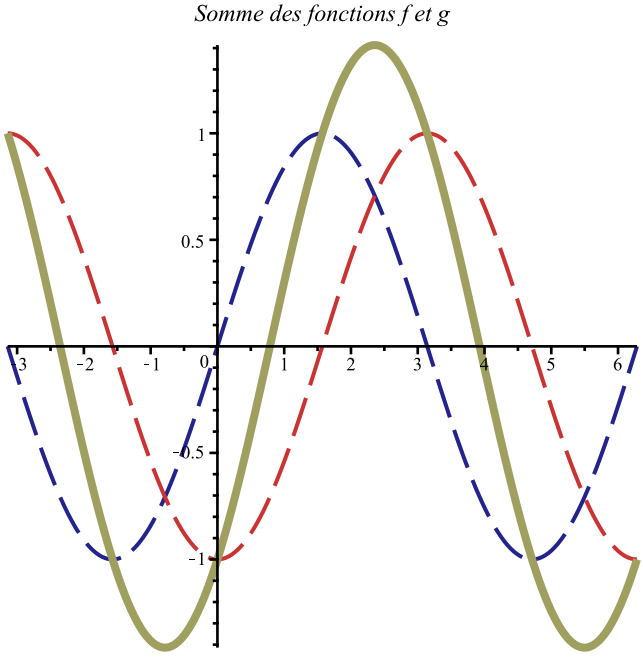
```
> f:=x->sin(x);
  g:=x->cos(x+Pi);
```

$$f := x \mapsto \sin(x)$$

$$g := x \mapsto \cos(x + \pi) \quad \textbf{(5.3)}$$

```
> Courbe_1:=plot([x,f(x),x=-Pi..2*Pi],color=navy,linestyle=3):
  Courbe_2:=plot([x,g(x),x=-Pi..2*Pi],color=orange,linestyle=3):
  Somme:=plot([x,(f+g)(x),x=-Pi..2*Pi],color=khaki,linestyle=1,
  thickness=3):
```

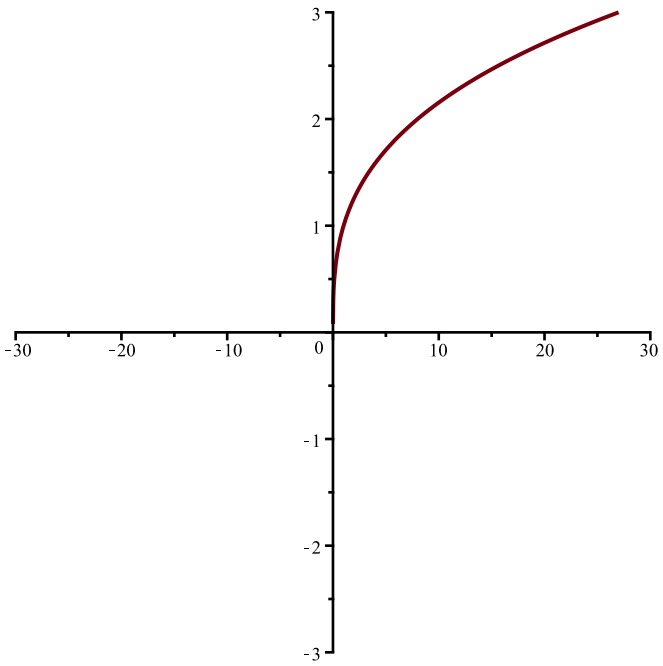
```
> display({Courbe_1,Courbe_2,Somme},
          title="Somme des fonctions f et g",
          titlefont=[TIMES,ITALIC,14]);
```



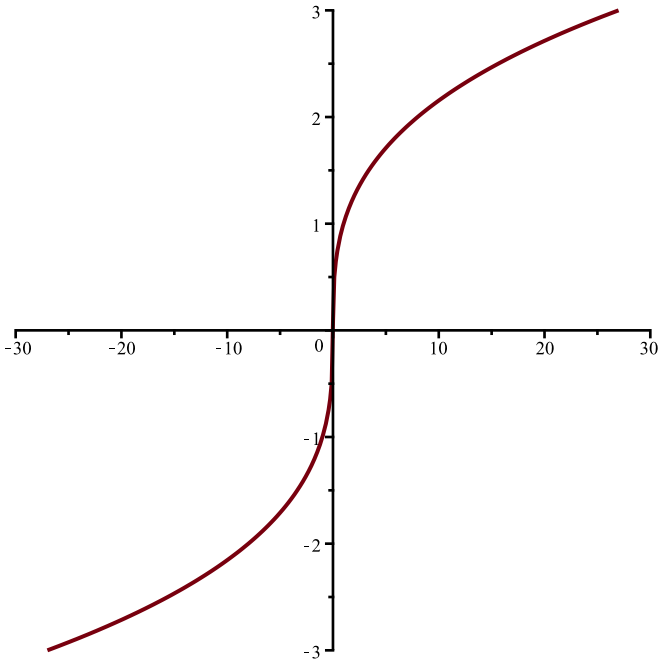
`root` et `surd` revisitée avec la macro-commande plot

Voyons graphiquement la différence dans le tracé de la fonction racine cubique d'un nombre lorsqu'on transpose en `Maple` la racine cubique avec `root` et `surd`.

```
> plot([x,root(x,3),x=-27..27],view=[-30..30,-3..3]);
```




```
> plot([x,surd(x,3),x=-27..27],view=[-30..30,-3..3]);
```



Expliquer la différence qu'on observe entre l'utilisation de `root` et de `surd` dans le tracé des deux derniers graphiques.

Votre réponse:

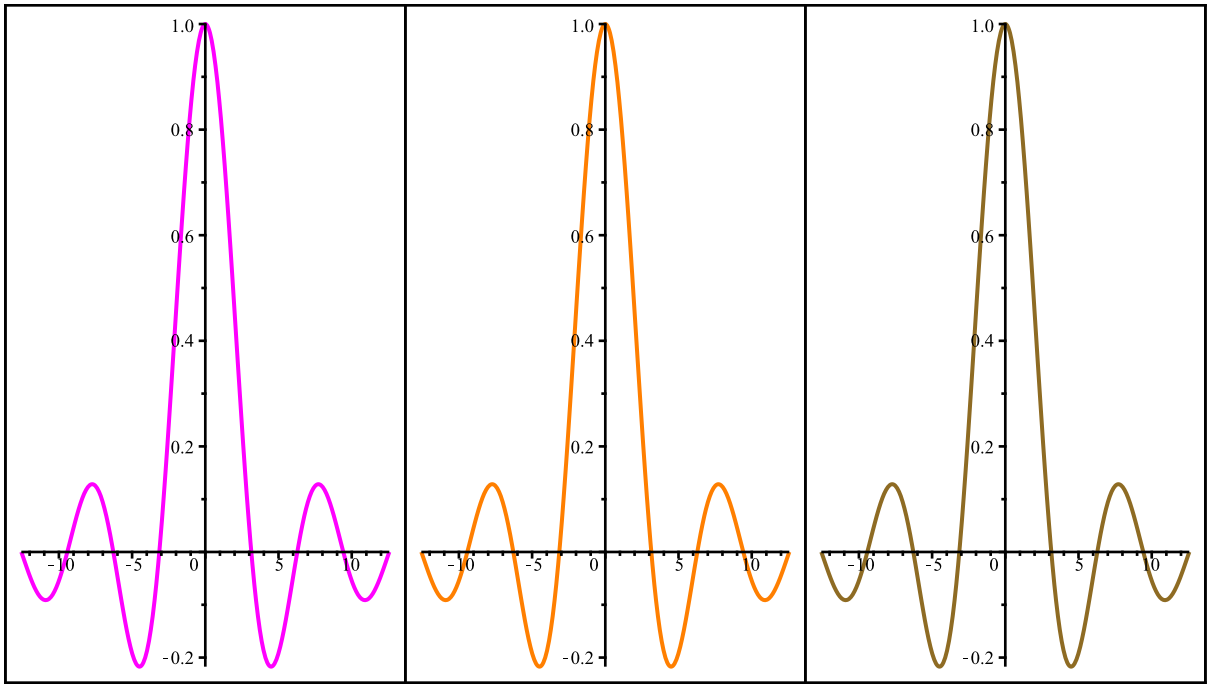
Complément sur la superposition de tracés de courbes dans le plan

Voyons comment il possible de contrôler l'ordre avec lequel s'effectue une superposition de tracés dans un même graphique. Cela nécesssite l'utilisation de crochets au lieu d'accolades.

```
> A:=plot([x,sin(x)/x,x=-4*Pi..4*Pi],color=magenta):
  B:=plot([x,sin(x)/x,x=-4*Pi..4*Pi],color=coral):
  C:=plot([x,sin(x)/x,x=-4*Pi..4*Pi],color=sienna):
```

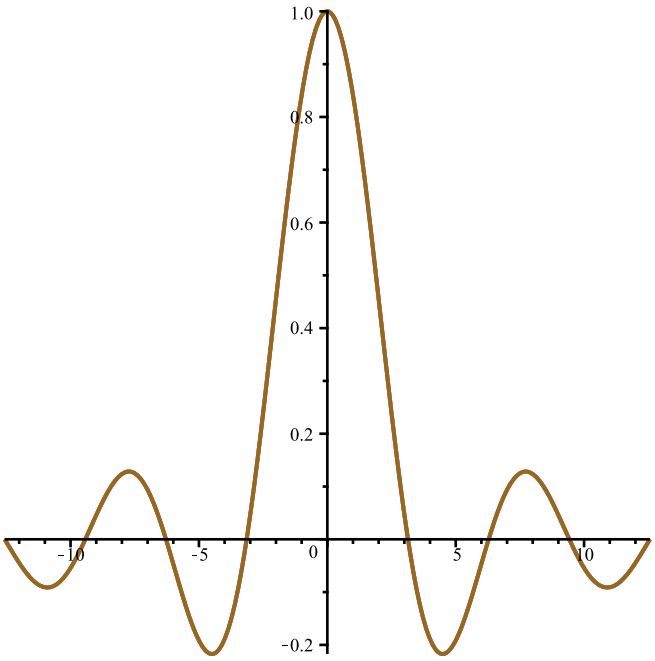
Voici d'abord, séparément, les tracés des graphes A, B et C.

```
> display(Matrix(1,3,[A,B,C]));
```



Superposons-les maintenant avec la macro-commande `display`.

```
> display([A,B,C]);
```

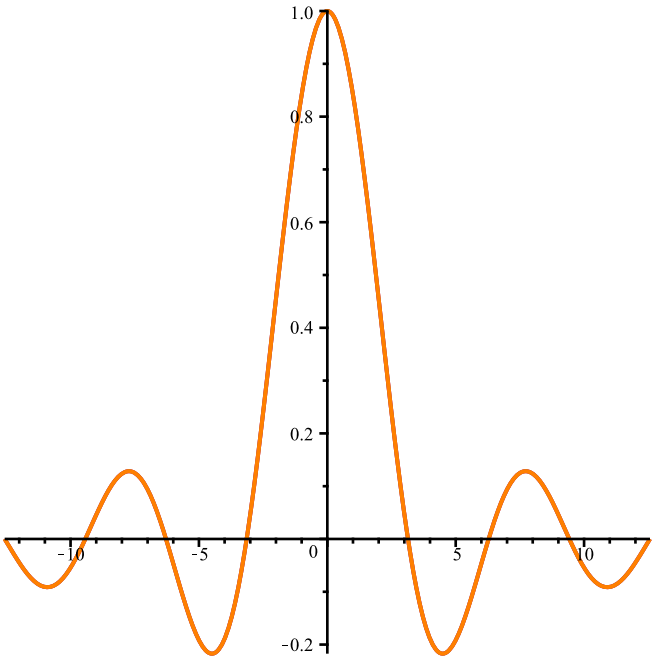


On ne voit que le tracé C. Pouvez-vous en donner la raison ?

Votre réponse:

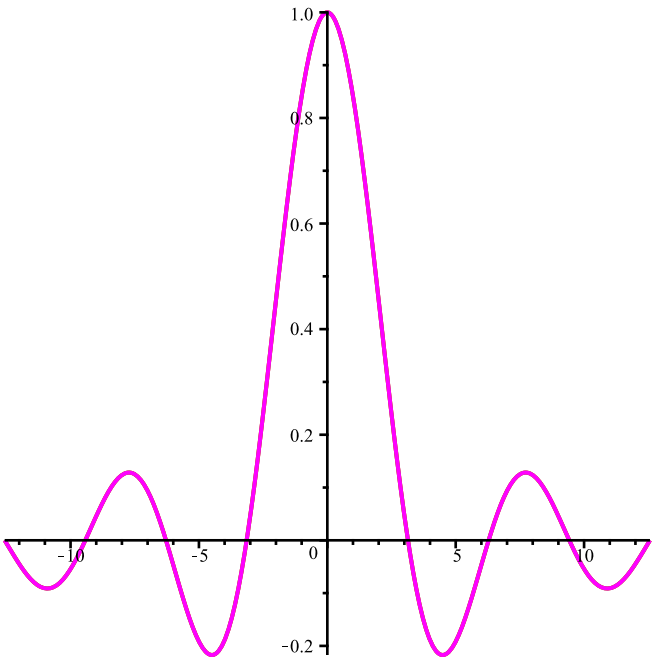
Noter la différence avec la superposition suivante.

```
> display([A,C,B]);
```



Ici également.

```
> display([B,C,A]);
```

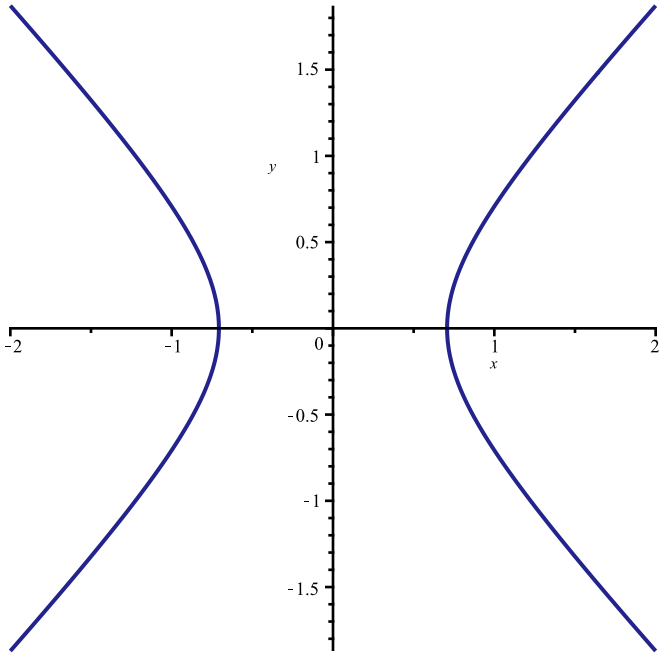



```
> unassign('A','B','C');
```

Tracés de courbes définies implicitement

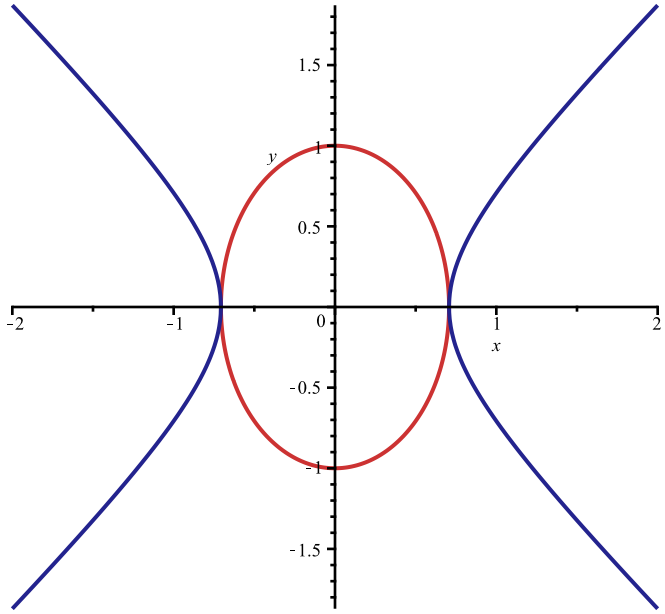
La macro-commande `implicitplot` de l'extension `plots` permet de tracer des courbes définies implicitement. Par exemple, traçons l'hyperbole d'équation $x^2 - y^2 = \frac{1}{2}$. Avec `implicitplot`, le premier argument doit être une égalité définissant implicitement y comme fonction de x et ensuite, il faut préciser les intervalles de traçage pour les variables x et y.

```
> C1:=implicitplot(x^2-y^2=1/2,x=-2..2,y=-2..2,color=navy):
  C1;
```



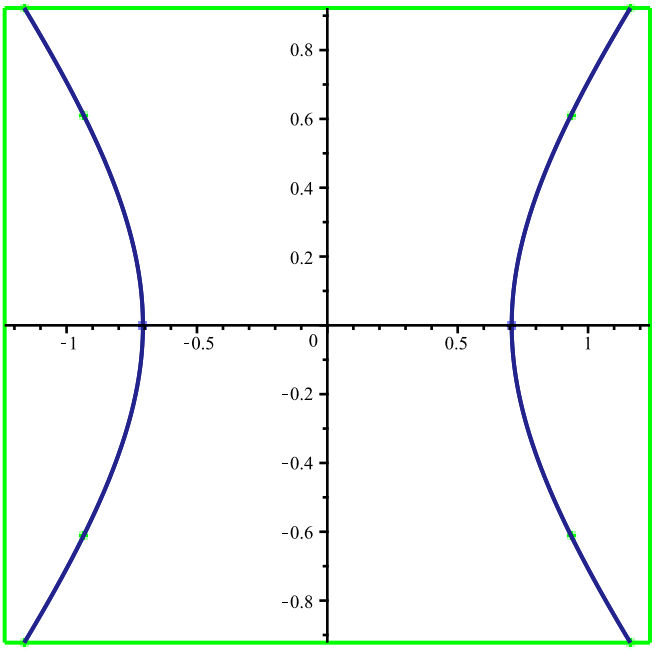
Superposons finalement ce tracé avec celui de l'ellipse d'équation $2x^2 + y^2 = 1$.

```
> C2:=implicitplot(2*x^2+y^2=1,x=-1..1,y=-1..1,color=orange):
> display({C1,C2},scaling=constrained);
```

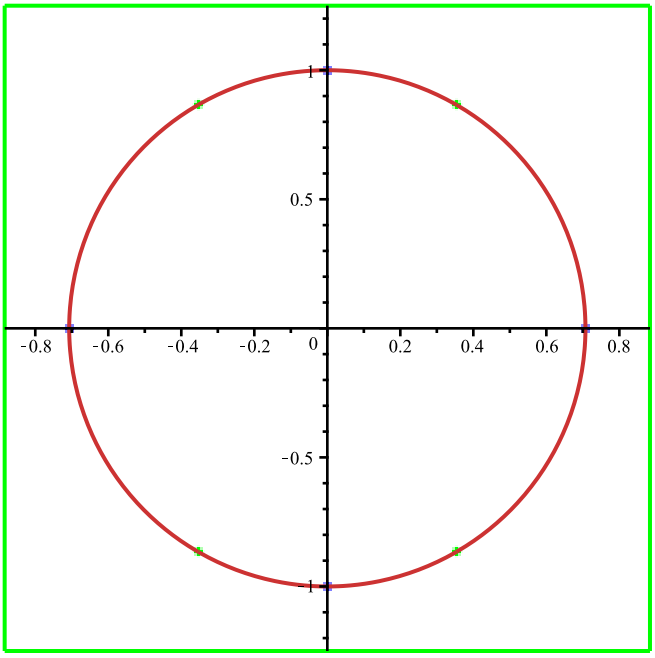


Traçons de nouveau cette ellipse et cette hyperbole mais, cette fois, en utilisant la macro-commande `plot_real_curve` de l'extension `algcurves`. Dans ce cas-ci, il n'y a pas d'intervalles de traçage à préciser.

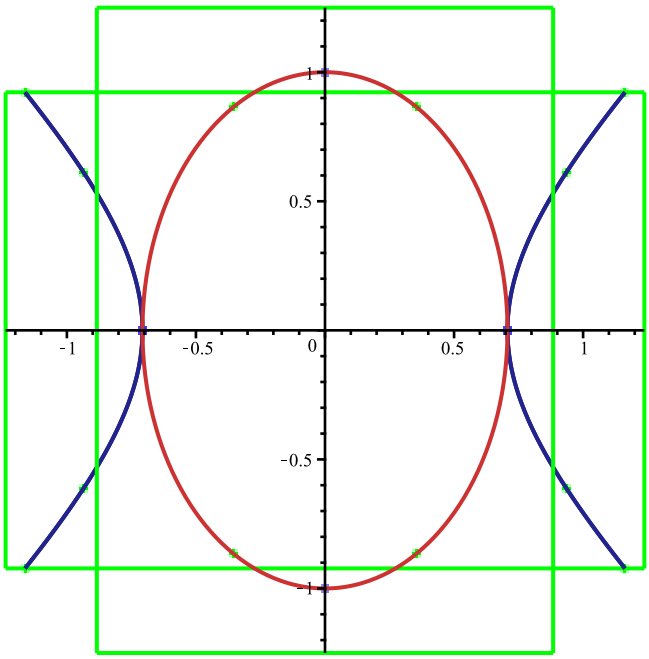
```
> C3:=plot_real_curve(x^2-y^2-1/2,x,y,colorOfCurve=navy):
  C3;
```




```
> C4:=plot_real_curve(2*x^2+y^2-1,x,y,colorOfCurve=orange):
  C4;
```

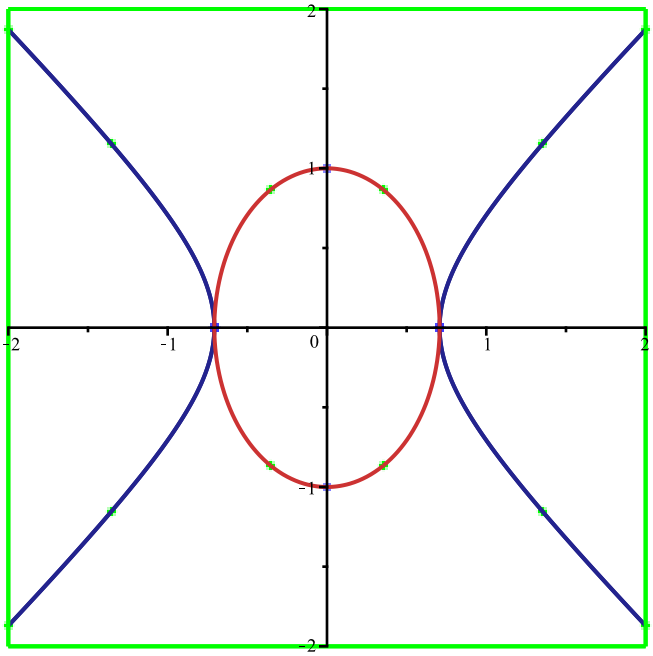



```
> display({C3,C4},scaling=constrained);
```



Nous devons utiliser l'option `view` pour contrôler individuellement l'intervalle d'affichage de chaque tracé pour mieux effectuer ensuite une superposition.

```
> C3:=plot_real_curve(x^2-y^2-1/2,x,y,colorOfCurve=navy,view=[-2..2,-2.
  .2]):
  C4:=plot_real_curve(2*x^2+y^2-1,x,y,colorOfCurve=orange,view=[-2..2,
  -2..2]):
  display({C3,C4},scaling=constrained);
```



Ce qui a été présenté ici n'est qu'un aperçu des possibilités de traçage de lieux géométriques dans le plan.