Chapitre 1: Calcul matriciel et représentation graphique.

Partie 1 : Calcul matriciel.

Ecriture directe de matrices et vecteurs.

MAPLE possède deux formes de constructions de vecteurs et de matrices. La première consiste en une écriture symbolique de la suite des composantes du vecteur entourée des caractères "<" et ">". Les matrices sont définies colonne par colonne en juxtaposant les "vecteurs colonnes" avec le caractère | "obtenue par Alt Gr + 6 alpha"et entourant le tout par < et >
Cette syntaxe autorise de façon simple l'augmentation en colonne(s) des matrices.
La deuxième construction utilise les fonctions Vector et Matrix. Nous verrons que ces fonctions permettent des constructions beaucoup plus élaborées.
Il faut faire attention à utiliser un V et un M majuscule pour les mots Vector et Matrix. En effet, avec un v ou un m minuscule, MAPLE construit aussi un vecteur ou une matrice mais avec une autre structure (celle utilisée dans les versions antérieures du logiciel et conservée dans cette version pour des raisons de compatibilité). MAPLE ne signalera donc aucune erreur, mais l'objet ainsi construit n'aura pas les propriétés décrites dans ce chapitre.
La commande Vector
Appel type : Vector([v[1],v[2],.....])
Il suffit de lister entre crochet les éléments du vecteur.
La commande Matrix
Appel type : Matrix([Ligne1],[Ligne2],[Ligne3],....)
Pour les matrices on donne en argument une liste de listes décrivant la matrice ligne par ligne.

```
> V1:=<1,2,3,4,5>;V2:=Vector([1,2,3,4,5]);
```

$$V1 := \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

$$V2 := \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

```
> whattype(V1);whattype(V2);
```

$$Vector[column]$$

$$Vector[column]$$

```
> M:=<<1,2,3>|<0,2,3>|<0,0,-1>>;
```

$$M := \begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 3 & 3 & -1 \end{bmatrix}$$

```
> whattype(M);
```

Matrix

```
> M:=<<0,0,6>|M>;
```

$$M := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 6 & 3 & 3 & -1 \end{bmatrix}$$

```
> whattype(M);
```

Matrix

```
> M:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
```

$$M := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> F:=Matrix(3,4);
```

$$F := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> K:=Matrix(2,3,(i,j)->((i+j)^2));
```

$$K := \begin{bmatrix} 4 & 9 & 16 \\ 9 & 16 & 25 \end{bmatrix}$$

```
>
> for i from 1 to 3 do
    for j from i to 4 do
      F[i,j]:=i*j;
  od;od;
> F;
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 4 & 6 & 8 \\ 0 & 0 & 9 & 12 \end{bmatrix}$$

Opérations élémentaires

Les opérations élémentaires sont prévues par les opérateurs suivants :
- L'addition de matrices ou de vecteurs se fait l'opérateur "+".
- Le produit par un scalaire est noté par "*".
- Le produit matriciel est noté par "."

```
> restart:
  u:=Vector([1,2,3]):v:=Vector([4,5,6]):w:=Vector([7,8,9,0]):
```

```
> u+v;
```

$$\begin{bmatrix} 5 \\ 7 \\ 9 \end{bmatrix}$$

```
> u+w;
```

Error, (in rtable/Sum) invalid input: dimensions do not match: Vector[column](1 .. 3) cannot be added to Vector[column](1 .. 4)

```
> 3*v;
```

$$\begin{bmatrix} 12 \\ 15 \\ 18 \end{bmatrix}$$

```
> M:=Matrix([[1,-1,0],[-1,2,-3],[3,1,1]]):N:=Matrix([[4,-2,3],[5,
  2,-3],[2,7,9]]):
```

```
> M+N;
```

$$\begin{bmatrix} 5 & -3 & 3 \\ 4 & 4 & -6 \\ 5 & 8 & 10 \end{bmatrix}$$

```
> 3*M;
```

$$\begin{bmatrix} 3 & -3 & 0 \\ -3 & 6 & -9 \\ 9 & 3 & 3 \end{bmatrix}$$

```
> M*N;
```

Error, (in rtable/Product) use *~ for elementwise multiplication of Vectors or Matrices; use . (dot) for Vector/Matrix multiplication

```
> M.N;
```

$$\begin{bmatrix} -1 & -4 & 6 \\ 0 & -15 & -36 \\ 19 & 3 & 15 \end{bmatrix}$$

```
> M^2=M.M;
```

$$\begin{bmatrix} 2 & -3 & 3 \\ -12 & 2 & -9 \\ 5 & 0 & -2 \end{bmatrix} = \begin{bmatrix} 2 & -3 & 3 \\ -12 & 2 & -9 \\ 5 & 0 & -2 \end{bmatrix}$$

```
> M^(-1);
```

$$\begin{bmatrix} \frac{5}{13} & \frac{1}{13} & \frac{3}{13} \\ -\frac{8}{13} & \frac{1}{13} & \frac{3}{13} \\ -\frac{7}{13} & -\frac{4}{13} & \frac{1}{13} \end{bmatrix}$$

```
> %.M;
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> M^(-2);(M^2)^(-1);
```

$$\begin{bmatrix} -\frac{4}{169} & -\frac{6}{169} & \frac{21}{169} \\ -\frac{69}{169} & -\frac{19}{169} & -\frac{18}{169} \\ -\frac{10}{169} & -\frac{15}{169} & -\frac{32}{169} \end{bmatrix}$$

$$\begin{bmatrix} -\frac{4}{169} & -\frac{6}{169} & \frac{21}{169} \\ -\frac{69}{169} & -\frac{19}{169} & -\frac{18}{169} \\ -\frac{10}{169} & -\frac{15}{169} & -\frac{32}{169} \end{bmatrix}$$

$$\begin{bmatrix} -\frac{4}{169} & -\frac{6}{169} & \frac{21}{169} \\ -\frac{69}{169} & -\frac{19}{169} & -\frac{18}{169} \\ -\frac{10}{169} & -\frac{15}{169} & -\frac{32}{169} \end{bmatrix}$$

Affectation d'éléments d'une matrice ou d'un vecteur

On peut remplacer une suite de composantes, dont les indices sont consécutifs, par un Vecteur de même nombres de composantes

```
> v:=Vector([1,2,3,4,5]);
```

$$v := \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

```
> v[3..5]:=<sqrt(3),sqrt(4),sqrt(5)>: v;
```

$$\begin{bmatrix} 1 \\ 2 \\ \sqrt{3} \\ 2 \\ \sqrt{5} \end{bmatrix}$$

```
> v[2..3]:=<1,2,3>;
Error, number of indices does not match rank
```

Si toutes les composantes à remplacer doivent prendre la même valeur, la syntaxe peut se simplifier

```
> v:=Vector([1,2,3,4,5]):
  v[2..4]:=0: v;
```

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 5 \end{bmatrix}$$

Il est possible de nommer les éléments d'un vecteur ou d'une matrice à partir de la droite, pour celà les indices sont des entiers négatifs.

```
> v:=Vector([1,2,3,4,5]):
  v[-1];v[-2];
```

$$5$$

$$4$$

```
> v[2..-2]:=0:v;
```

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 5 \end{bmatrix}$$

```
> M:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
  M[-3..-2,2..-1];
```

$$M := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$$

Bibliothèque Linear Algebra

```
> with(LinearAlgebra):
> Determinant(M);
```

$$13$$

```
> with(linalg):det(M);
```

$$13$$

SubMatrix

La commande SubMatrix permet d'extraire une matrice à partir d'une autre matrice.
L'appel type est SubMatrix(A,r,s)
Où A : La matrice source
r: est soit un intervalle d'entiers [n..m] soit un ensemble d'entiers. [r1,r2....rm]
s: est soit un intervalle d'entiers [p..q] soit un ensemble d'entiers. [s1,s2....sm]
La commande à pour conséquence l'extraction des lignes ayant pour indices r□,r□....r_{m} et ensuite l'extraction parmis ces lignes des colonnes ayant pour indices [s□,s□....s_{m}]

```
> M;
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> SubMatrix(M,[1,3],[1..2]);
```

$$\begin{bmatrix} 1 & 2 \\ 7 & 8 \end{bmatrix}$$

```
> whattype(%);
```

$$Matrix$$

ColumnDimension & RowDimension

Appel type : ColumnDimension(Matrice), RowDimension(Matrice)
La commande ColumnDimension permet de donner le nombre de colonnes d'une matrice A.
La commande RowDimension permet de donner le nombre de lignes d'une matrice A.

```
> ColumnDimension(Matrix([[1,-1,0],[-1,2,-3],[1,-1,0]]));
```

$$3$$

```
> RowDimension(Matrix([[1,-1,0],[-1,2,-3],[1,-1,0]]));
```

$$3$$

Option shape

L'option shape permet de caractériser le "profil" d'une matrice. L'option identity crée une matrice identité.

```
> Id:=Matrix(3,3,shape=identity);
```

$$Id := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

On ne peut pas modifier une matrice crée avec l'option shape.

```
> Id[1,1]:=2;
```

```
Error, invalid assignment to identity diagonal
```

La matrice identitée crée n'est pas forcément carrée.

```
> Id2:=Matrix(2,3,shape=identity);
```

$$Id2 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

L'option shape=scalar[n] définit une matrice diagonale, non nécessairement carrée, dont les éléments diagonaux sont égaux à n

```
> M:=Matrix(3,3,shape=scalar[3]);
```

$$M := \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

L'option shape=diagonal définit une matrice diagonale, non nécessairement carrée, dont les éléments diagonaux sont ceux d'un vecteur source.

```
> Diag2:=Matrix(3,3,Vector([1,2,3]),shape=diagonal);
```

$$Diag2 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Dans une matrice crée par l'option diagonal seuls les éléments hors diagonale ne peuvent pas être modifiés

```
> Diag2[1,2]:=1;
```

Error, attempt to assign non-zero to off-diagonal entry of a diagonal Matrix

```
> Diag2[2,2]:=alpha;
```

$$Diag2_{2,2} := \alpha$$

```
> Diag2;
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

L'option shape=symmetric permet de créer des matrices symétriques.

```
> A:=Matrix(3,3,shape=symmetric);
```

$$A := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> A[1,2]:=alpha;A[2,3]:=beta;A[1,3]:=gamma;
```

$$A_{1,2} := \alpha$$

$$A_{2,3} := \beta$$

$$A_{1,3} := \gamma$$

```
> A;
```

$$\begin{bmatrix} 0 & \alpha & \gamma \\ \alpha & 0 & \beta \\ \gamma & \beta & 0 \end{bmatrix}$$

L'affectation à un élément A[i,j] d'une valeur change automatiquement l'élement A[j,i] pour la même valeur.

```
> A[2,1]:=teta;
```

$$A_{2,1} := teta$$

```
> A;
```

$$\begin{bmatrix} 0 & teta & \gamma \\ teta & 0 & \beta \\ \gamma & \beta & 0 \end{bmatrix}$$

En plus des options précédentes les étudiants sont invités à consulter le Help pour les options **fill** et **scan**

Affichage de matrices de grande taille.

Par défaut seul l'affichage de matrice de taille inférieure à 10 lignes et 10 colonnes est autorisés. Cette valeur peut être changée par la commande interface.
Pour les matrices de grandes tailles, un double clic ouvre une fenêtre qui permet de naviguer dans les éléments de la matrice.

```
> restart:
> M11:=Matrix(11,11,5);
```

$$M11 := \begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

```
> interface(rtablesize=15);
```

$$10$$

Cette dernière commande permet de changer le maximum des lignes et colonnes autorisées de 10 à 15. Le résultat affiché est la dernière valeur autorisée.

```
> M:=Matrix(50,50):
  for i from 1 to 50 do
   for j from 1 to 50 do
  if i>=j then
  M[i,j]:=1:
  fi:od:od:
> M;
```

$$\begin{bmatrix} \textit{50 x 50 Matrix} \\ \textit{Data Type: anything} \\ \textit{Storage: rectangular} \\ \textit{Order: Fortran\_order} \end{bmatrix}$$

La remarque Fortran_order signifie que le stockage de la matrice est effectué selon la norme Fortran (colonne par colonne) et C_order signifie que le stockage se fait selon la norme C (ligne par ligne).

Stockage de matrices de grandes tailles.

Les matrices de grandes tailles proviennent souvent de modélisation de phénomènes réels et leurs

stockage est une tache complexe à gérer pour les logiciels, lors de la modélisation de phénomènes réels les matrices ont des propriétés (symétrique, diagonale, tridiagonale...etc.)

```
> T:=Matrix(4,4,storage=triangular[upper]);
```

$$T := \begin{bmatrix} 0 & 0 & 0 & 0 \\ \_ & 0 & 0 & 0 \\ \_ & \_ & 0 & 0 \\ \_ & \_ & \_ & 0 \end{bmatrix}$$

```
> T[2,1];
```

Error, unable to lookup index (2, 1) -- not in storage

```
> T[1,2];
```

$$0$$

```
> T[1,1]:=alpha;
```

$$T_{1,1} := \alpha$$

```
> T;
```

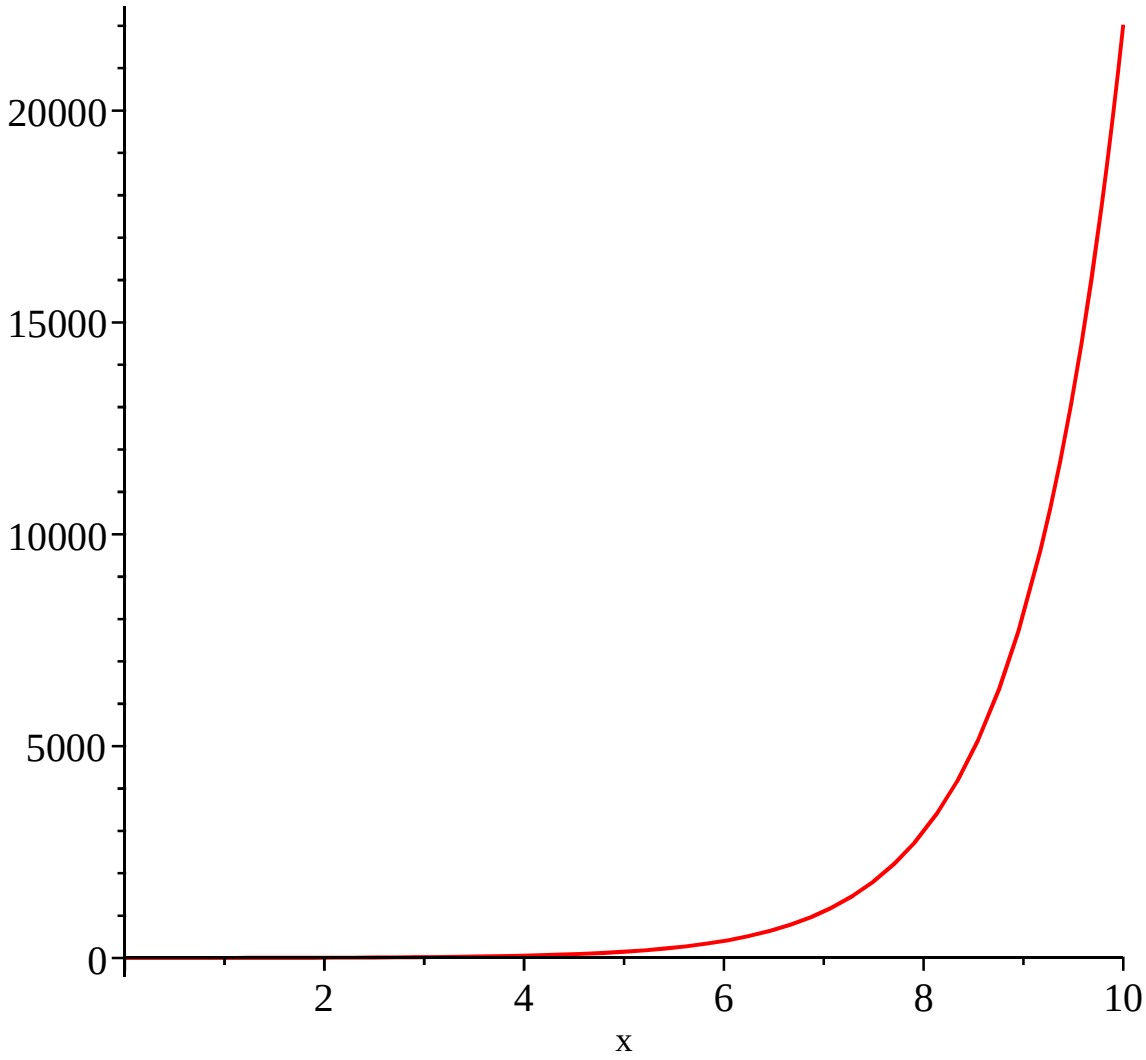
$$\begin{bmatrix} \alpha & 0 & 0 & 0 \\ \_ & 0 & 0 & 0 \\ \_ & \_ & 0 & 0 \\ \_ & \_ & \_ & 0 \end{bmatrix}$$

Partie 2 : Représentation graphiques.

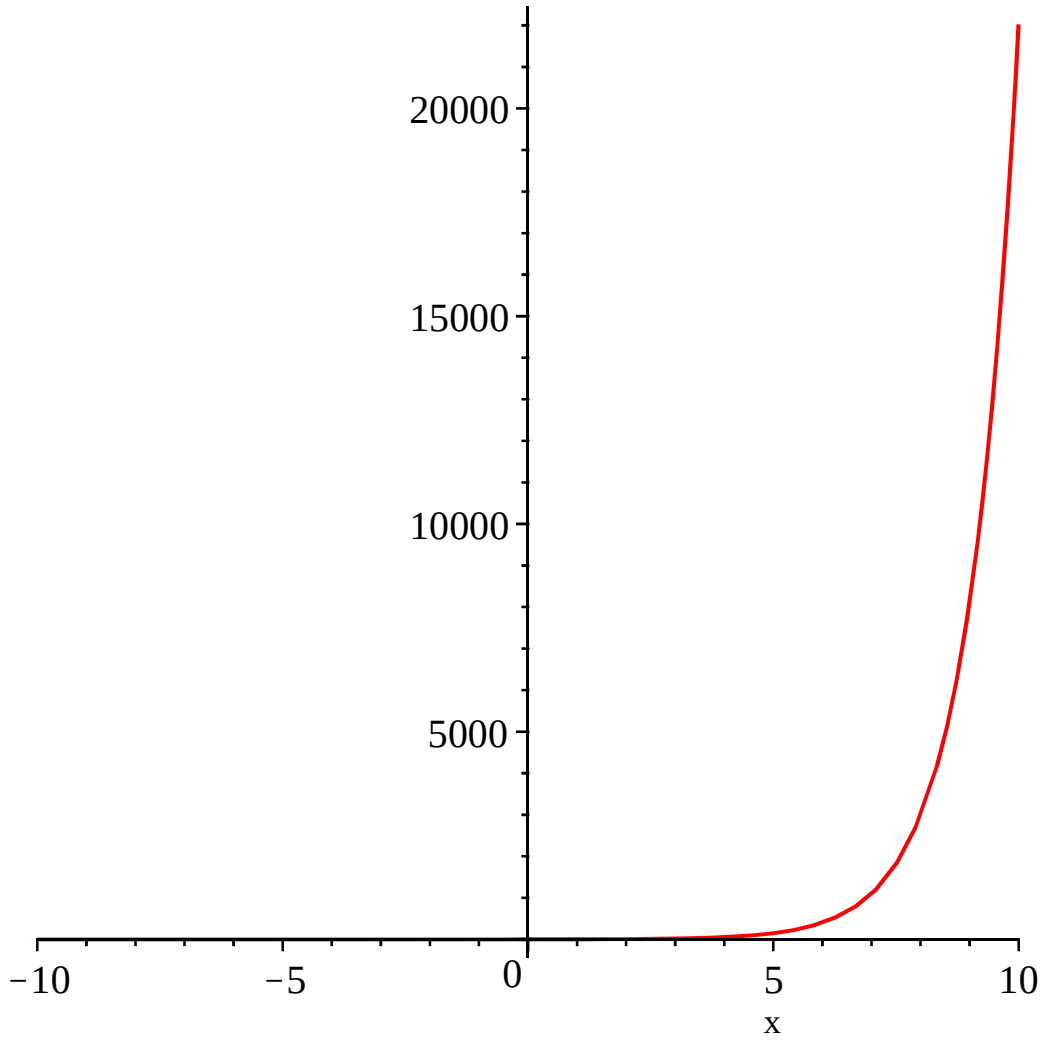
Graphe 2D

La commande plot permet de dessiner des graphes sous Maple, la syntaxe de base est la suivante
plot(*f(x)***,***x*=*a*..*b***)**
Où *f(x)* est une expression qui dépend de *x*.
a..b les bornes de l'intervalle pour lequel on souhaite afficher le graphe.
Si l'intervalle a..b n'est pas spécifié alors il est considéré par défaut comme ayant les bornes -10..10
Le mot **infinity** permet d'utiliser des bornes infinies pour les graphes.
Le graphe est obtenue aux bornes infinies en composant automatiquement la fonction souhaité avec arctan et de faire le graphe de la fonction composée aux bornes -1..1

```
> plot(exp(x),x=0..10);
```

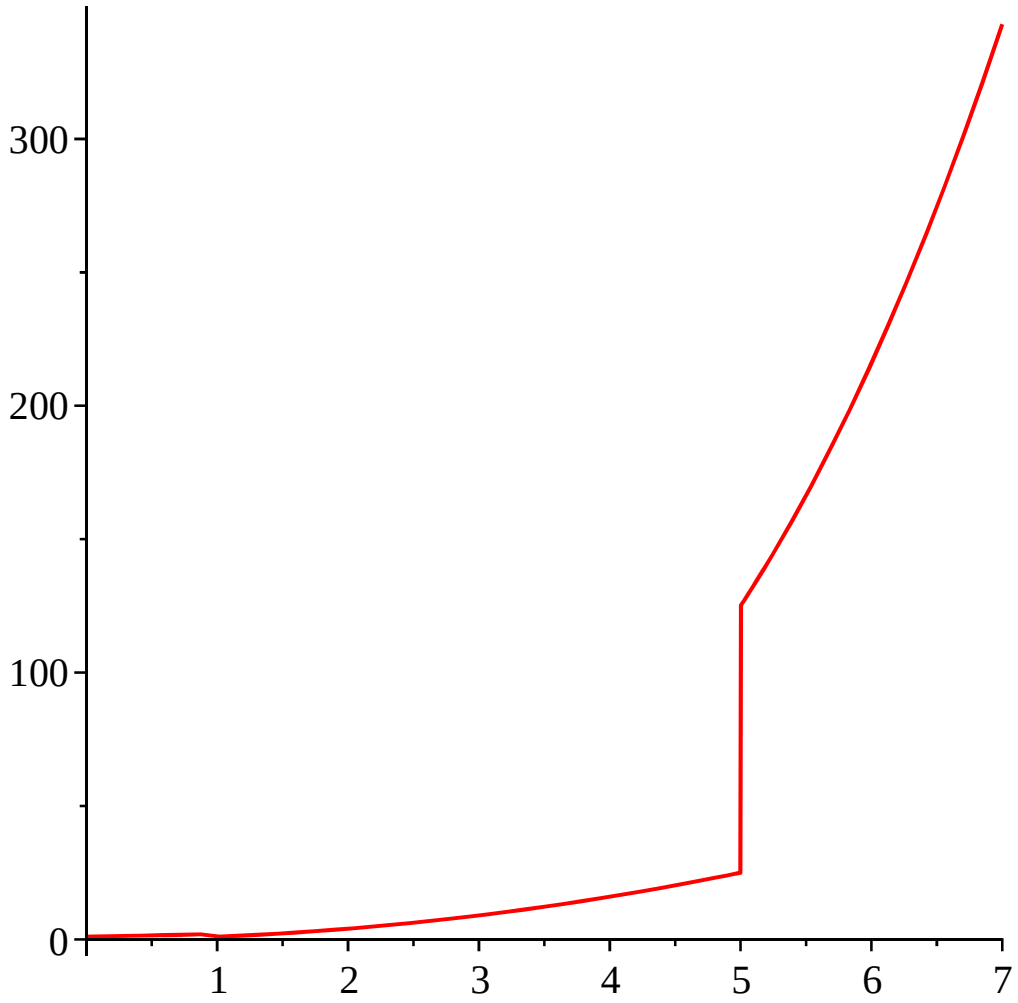



```
> plot(exp(x),x);
```



Graphe d'une procédure.
On peut tracer le graphe d'une procédure en donnant son nom et en ne précisant de variable pour l'axe des abssices.

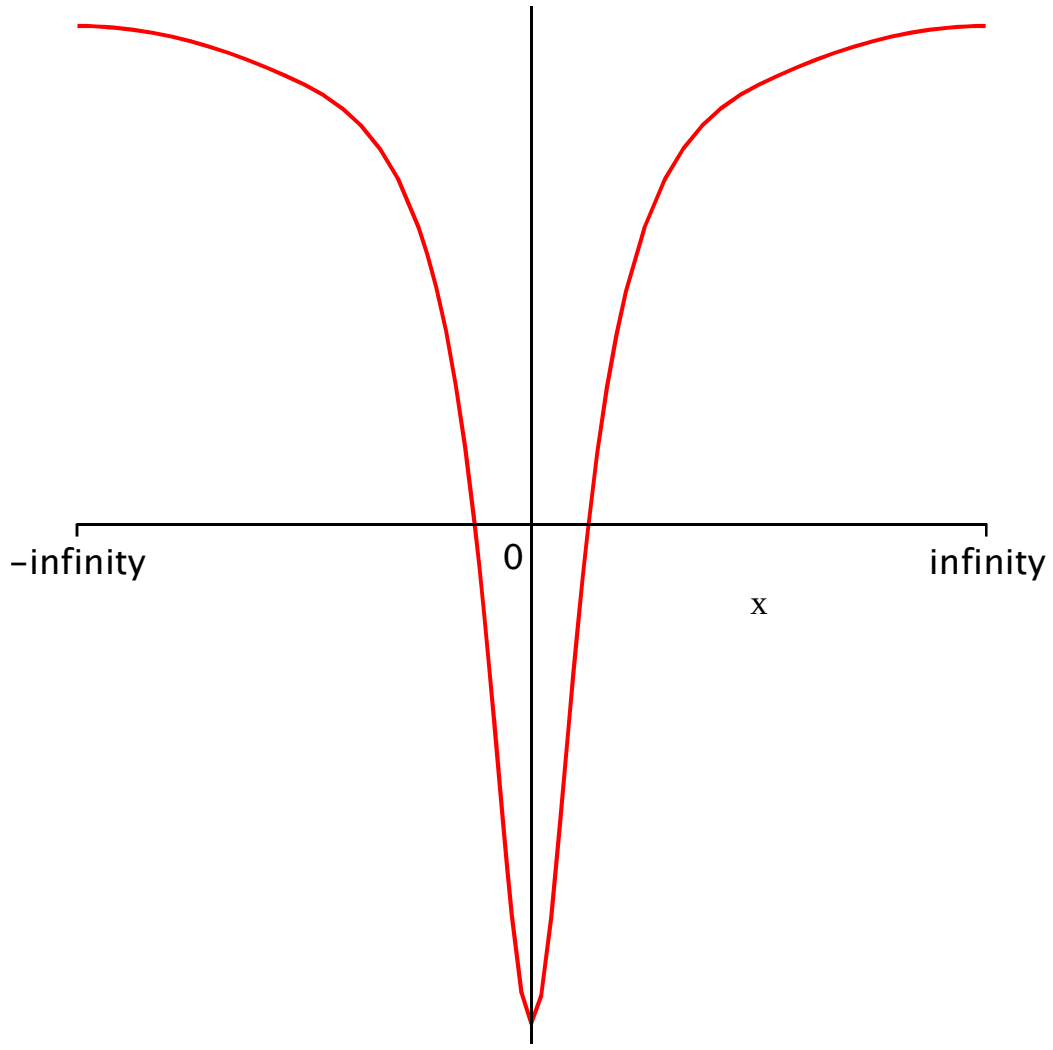
```
> f:=proc(x)
  if x<1 then x+1
  elif x>=1 and x<5 then x^2
  else x^3 fi;end:
> plot(f,0..7);
```

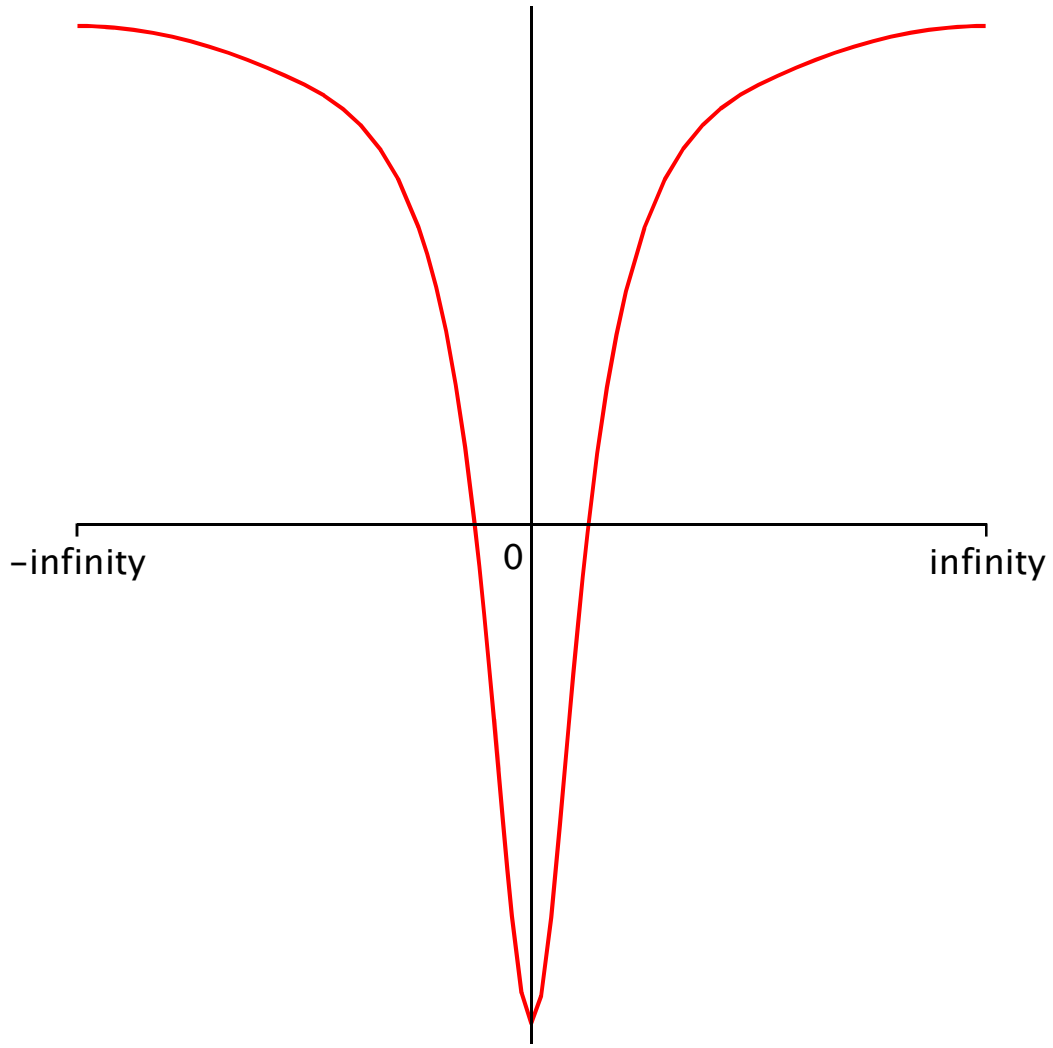


Si on définit une fonction par l'opérateur **arrow** -> alors il existe deux possibilités pour en faire le graphe, soit en la traitant comme une expression soit comme une procédure.

```
> f:=x->(x^2-1)/(x^2+1);
  plot(f(x),x=-infinity..infinity);
  plot(f,-infinity..infinity);
```

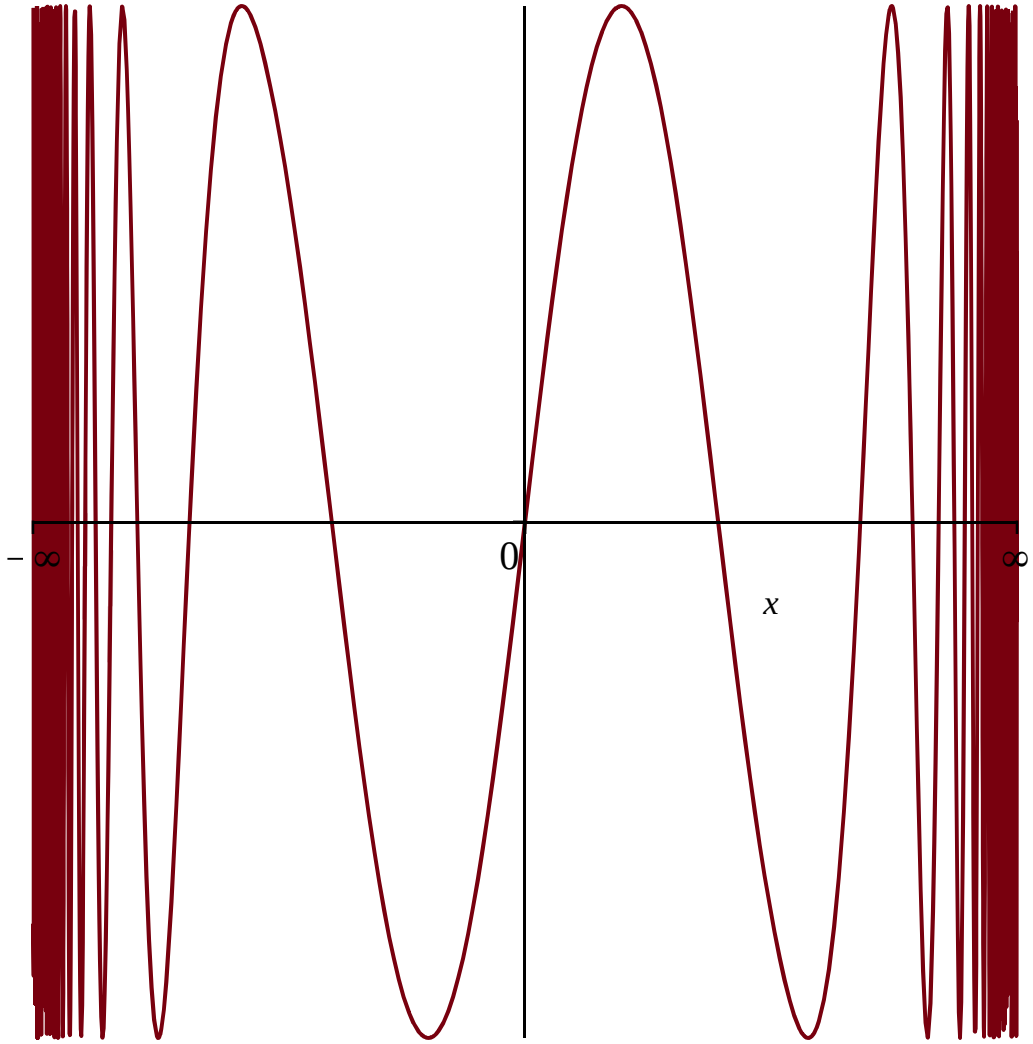
$$f := x \to \frac{x^2 - 1}{x^2 + 1}$$


–infinity
0
infinity
x



Il faut par contre éviter les mélanges.

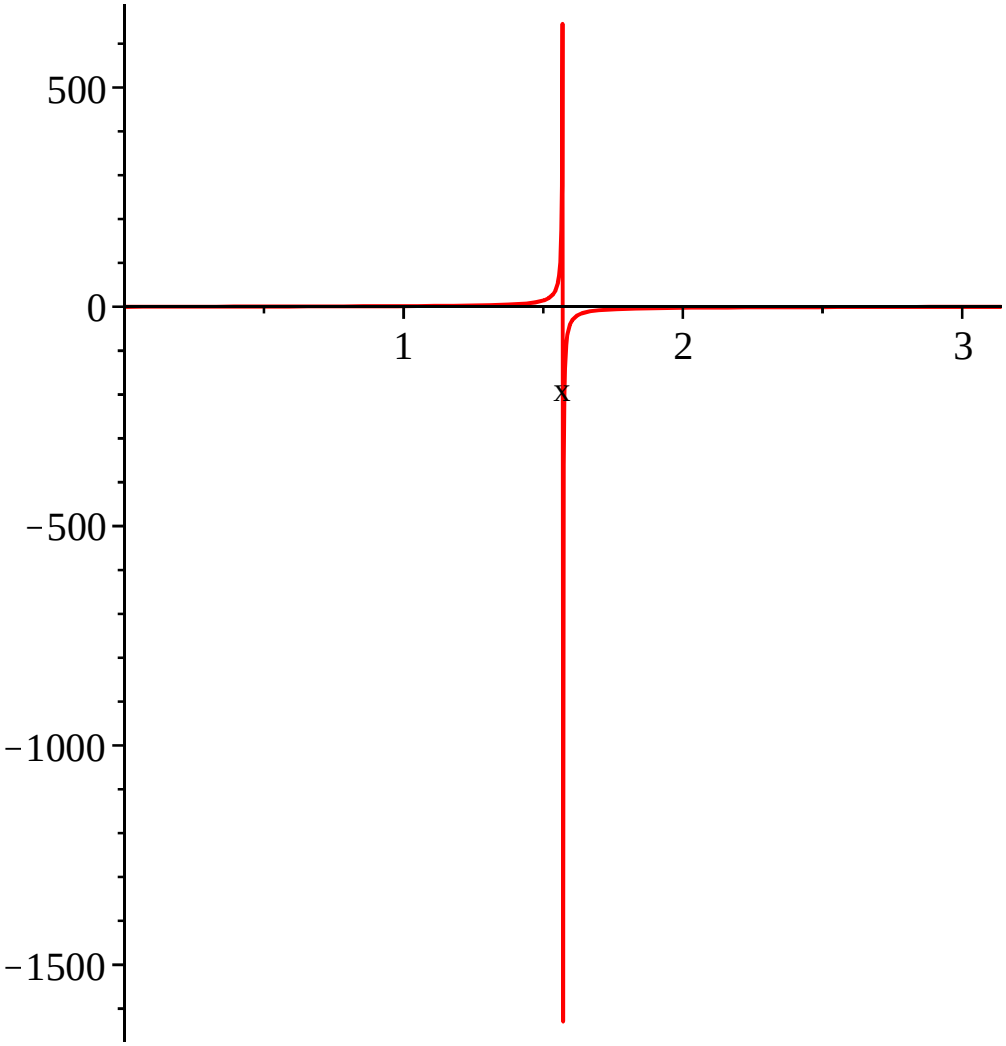
```
> plot(f,x=-infinity..infinity);
Error, (in plot) invalid plotting of procedures, perhaps you
mean plot(f, -infinity .. infinity)
> plot(f(x),-infinity..infinity);
Warning, unable to evaluate the function to numeric values in
the region; see the plotting command's help page to ensure the
calling sequence is correct

Error, empty plot
> plot(sin(x),x=-infinity..infinity);
```



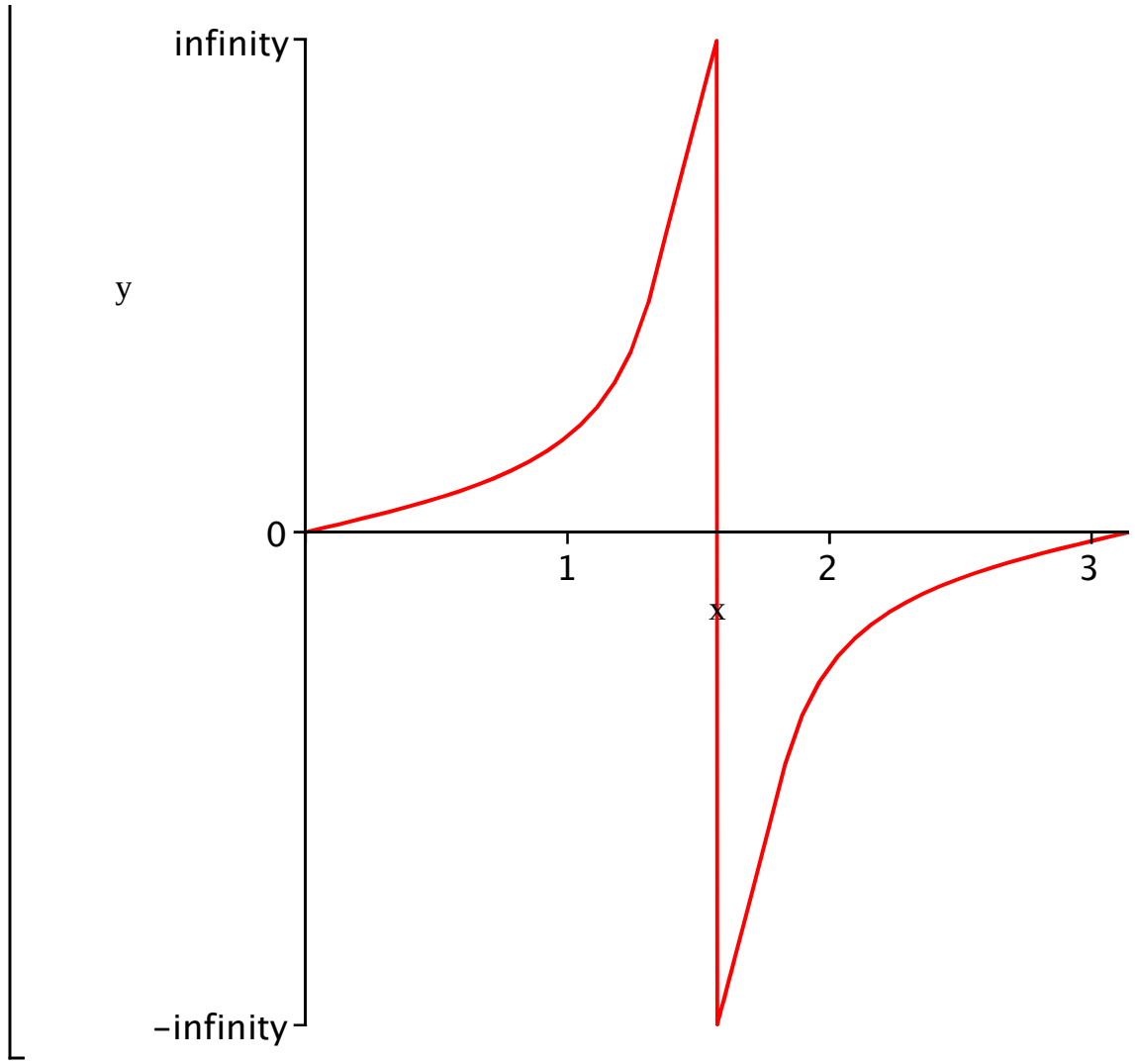

```
>
```

On remarquera que le graphe à l'infini est obtenue via une transformation vers l'intervalle -1..1 en utilisant la fonction arctan. Cette transformation donne une assez bonne idée du graphe mais crée un effet de distortion vers les bords du graphe.

```
> plot(tan(x),x=0..Pi);
```



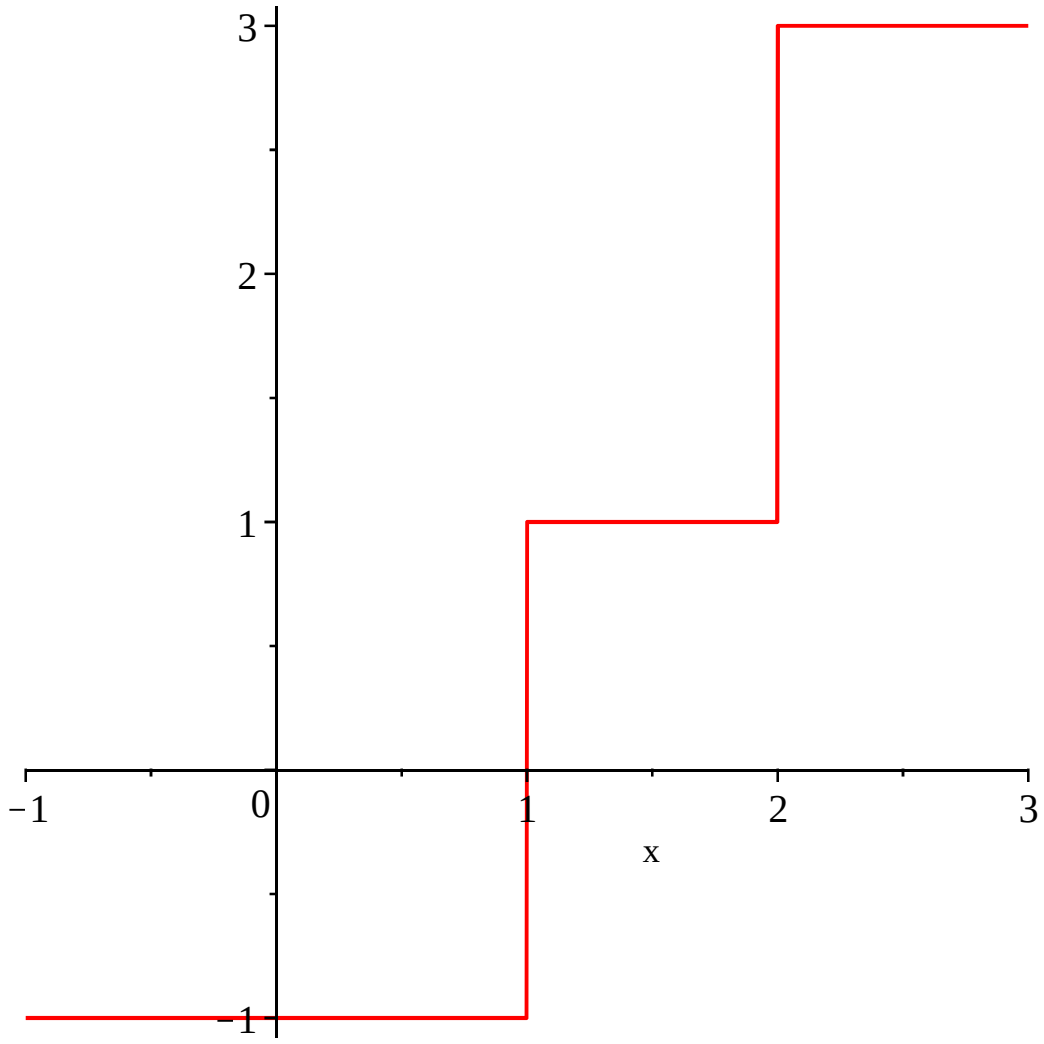

```
> plot(tan(x),x=0..Pi,y=-infinity..infinity);
```



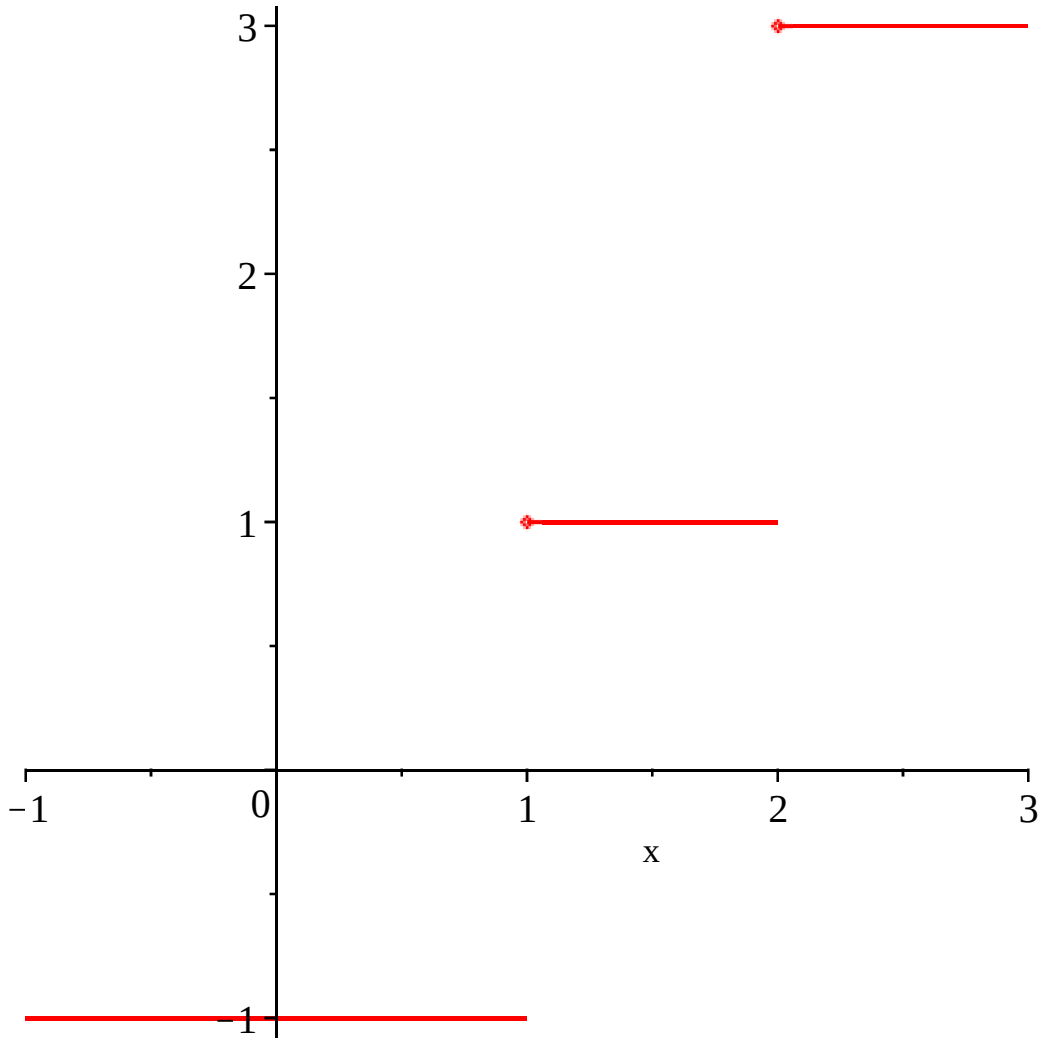

```
> h:= x -> piecewise( x<1, -1, x<2, 1, 3 );
```

$$h := x \rightarrow piecewise(x < 1, -1, x < 2, 1, 3)$$

```
> plot(h(x),x=-1..3);
```



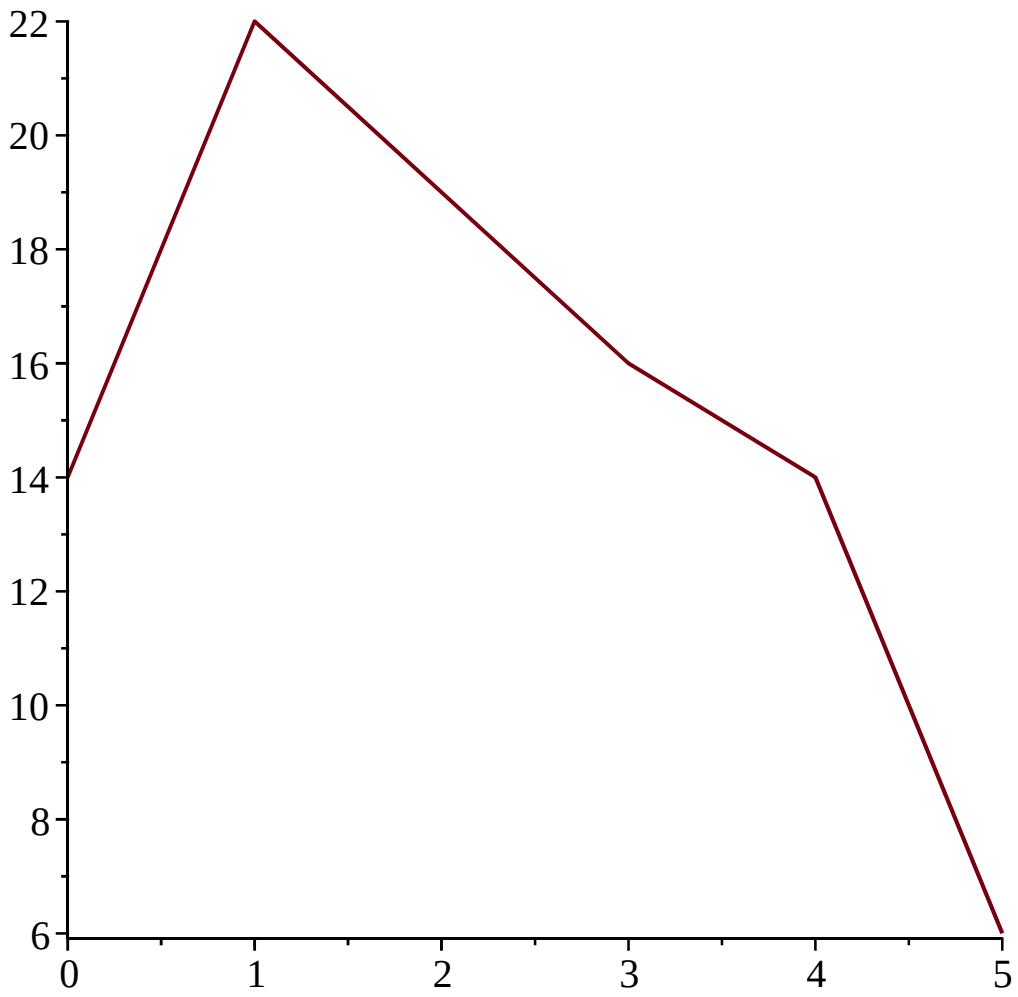

```
> plot(h(x),x=-1..3,discont=true);
```



Tracé d'un ensemble de points

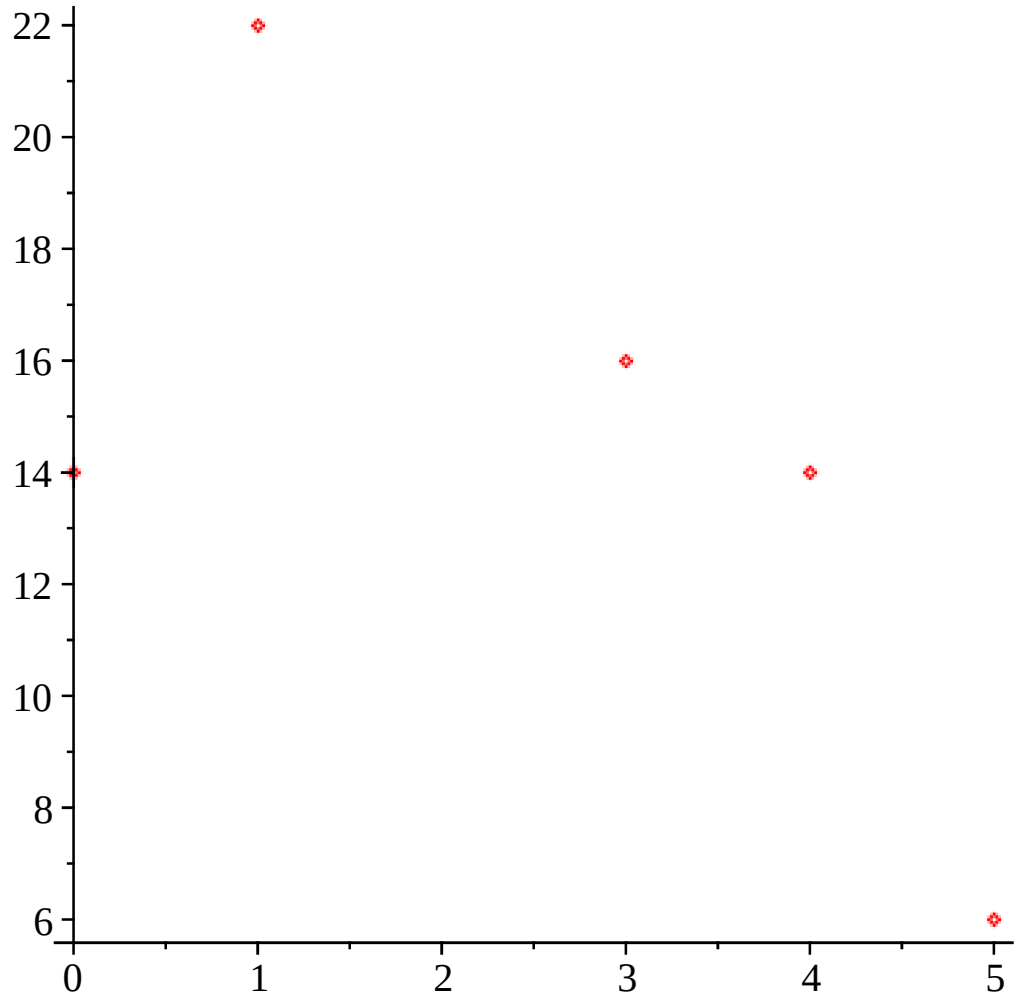
On peut obtenir le tracé d'une fonction à partir d'une liste de valeurs des coordonnées. Celles-ci seront données en écrivant les couples (x1, y1), (x2, y2),... sous la forme d'une liste de listes [[x1, y1], [x2, y2],...].

```
> plot([[0,14],[1,22],[3,16],[4,14],[5,6]]);
```



Par défaut le logiciel relie les points par des segments de droite, on peut obtenir les points seuls en utilisant l'option **style=point** et en choissisant la façon de marquer les points par l'option **symbol** où on peut choisir parmis "diamond,point..."
Ce changement peut être effectué de façon interactive en utilisant le menu qui apparait par clic droit de la souris sur le graphe ou bien dans la commande plot elle même.

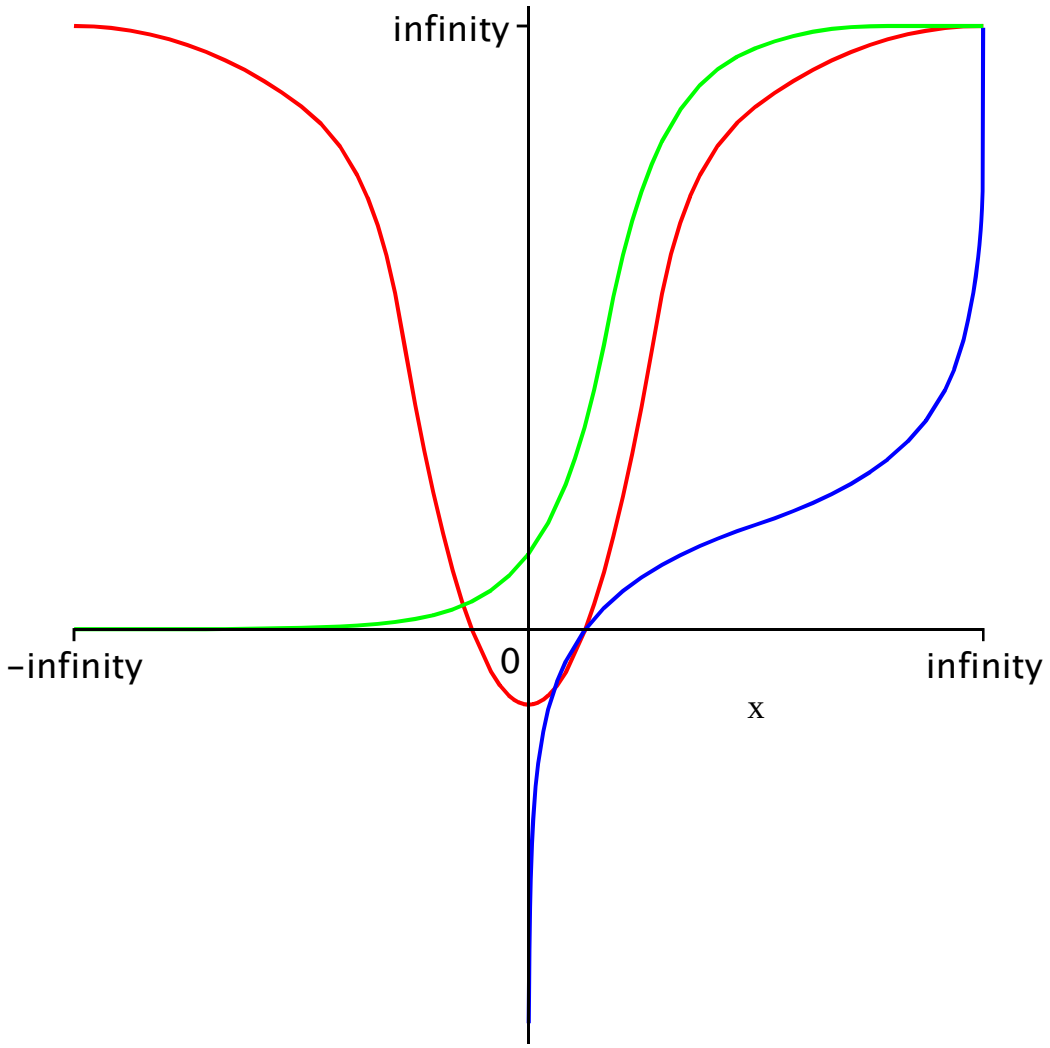
```
> plot([[0,14],[1,22],[3,16],[4,14],[5,6]],style=point,symbol=
  diamond);
```



Tracé de plusieurs graphes

On peut tracer plusieurs graphes sur un même repère il suffit de les placer entre deux crochets.On peut définir les couleurs des graphes avec l'option colour=[couleur1,couleur2,...].

```
> plot([x^2-1,ln(x),exp(x)],x=-infinity..infinity,colour=[red,
  blue,green]);
```

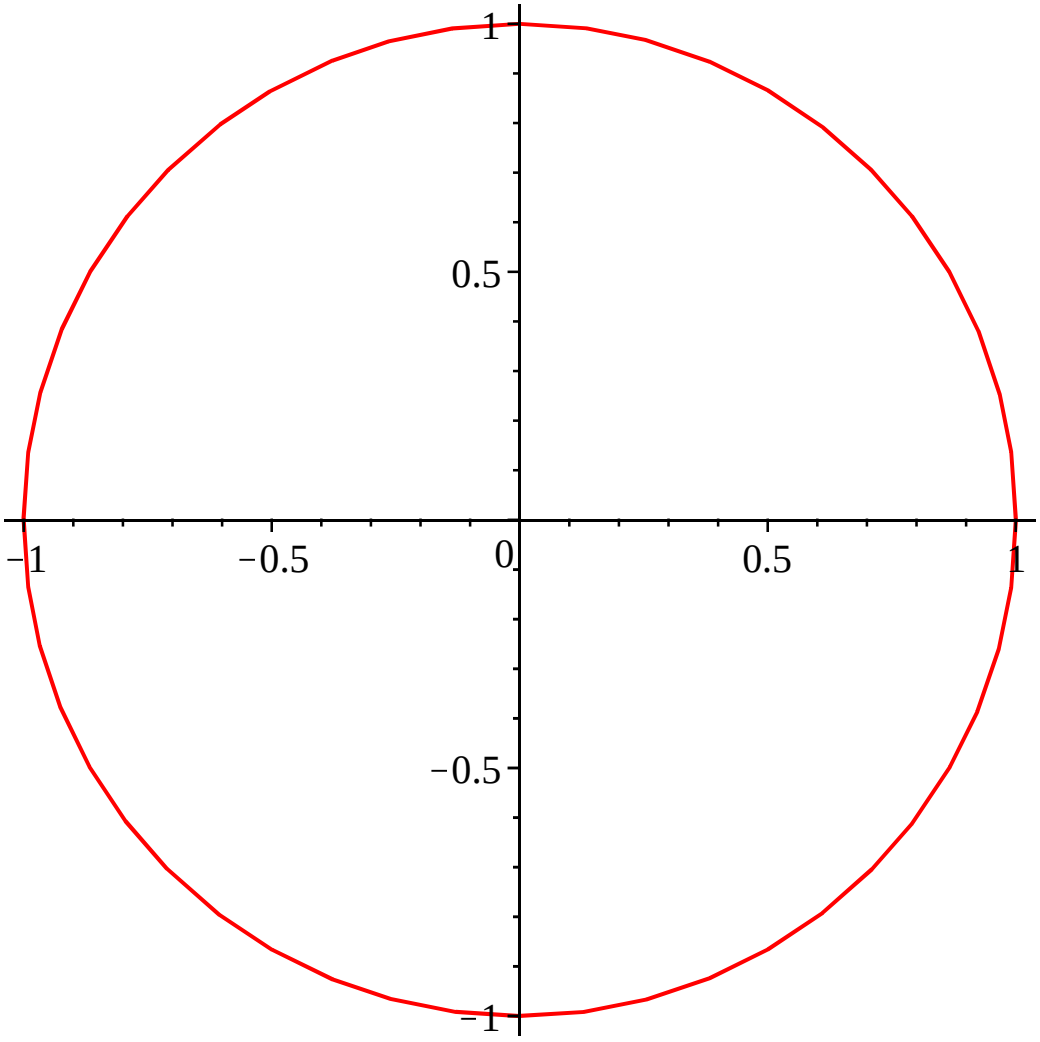


Courbes paramétriques

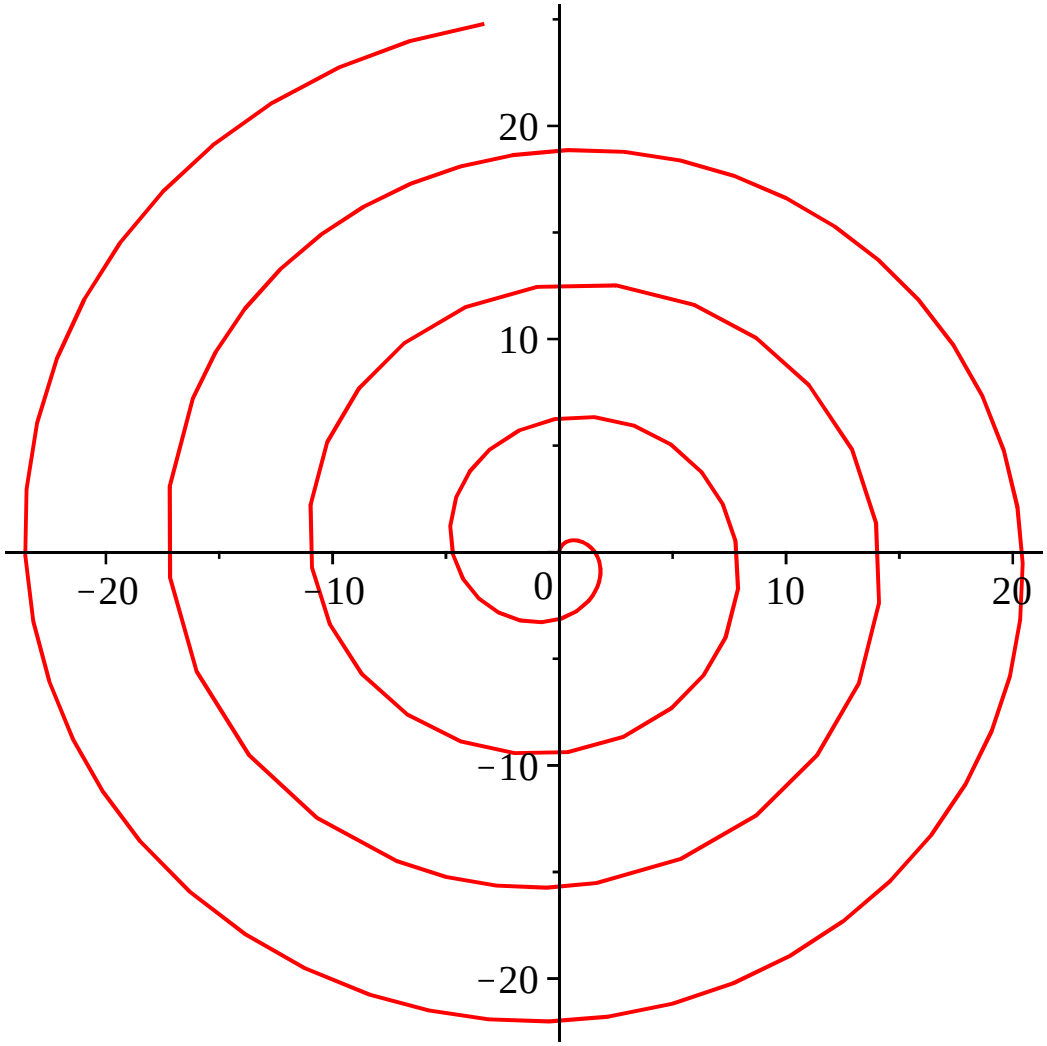
On peut tracer des courbes paramétriques par la syntaxe plot([x(t),y(t),t=a..b]); ou x(t) est l'expression décrivant les variations de l'abscisse en fonction du paramètre t et y(t) celles de l'ordonnée

MAPLE permet de tracer des courbes en coordonnées paramétriques toujours à l'aide de la fonction plot. Il suffit de remplacer l'expression et les variations de la variable par une liste de la forme [x(t), y(t), t = a .. b]. Ici t est le paramètre et a .. b son intervalle de variation, x(t) l'expression décrivant les variations de l'abscisse en fonction du paramètre et y(t) celles de l'ordonnée. On peut également imposer les intervalles de variation en abscisse et ordonnée avec l'option view=[x_min.. x_max,y_min..y_max]. Ces valeurs définiront les limites de la représentation graphique.

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

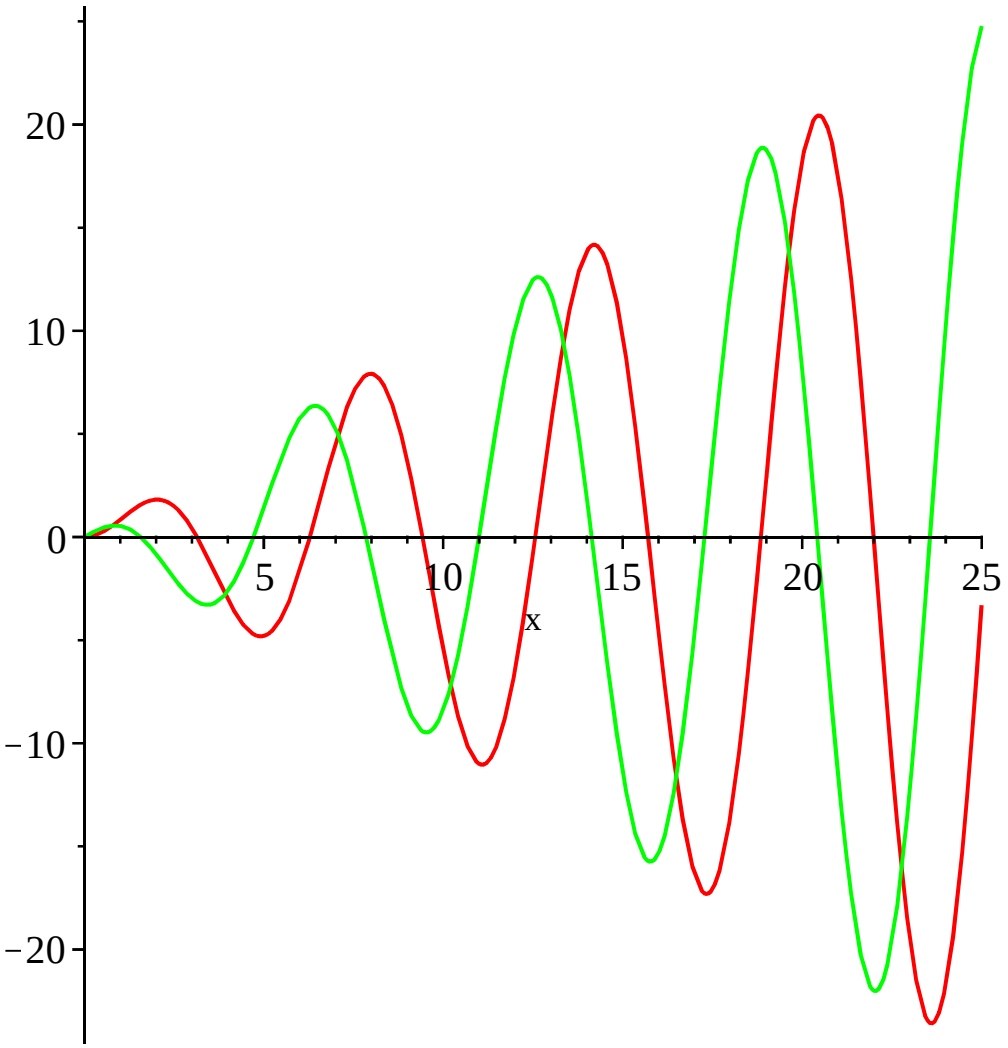



```
> plot([x*sin(x),x*cos(x),x=0..25]);
```



Attention si le crochet est fermé avant l'intervalle, la commande est interpreté comme deux courbes tracés sur le même repère.

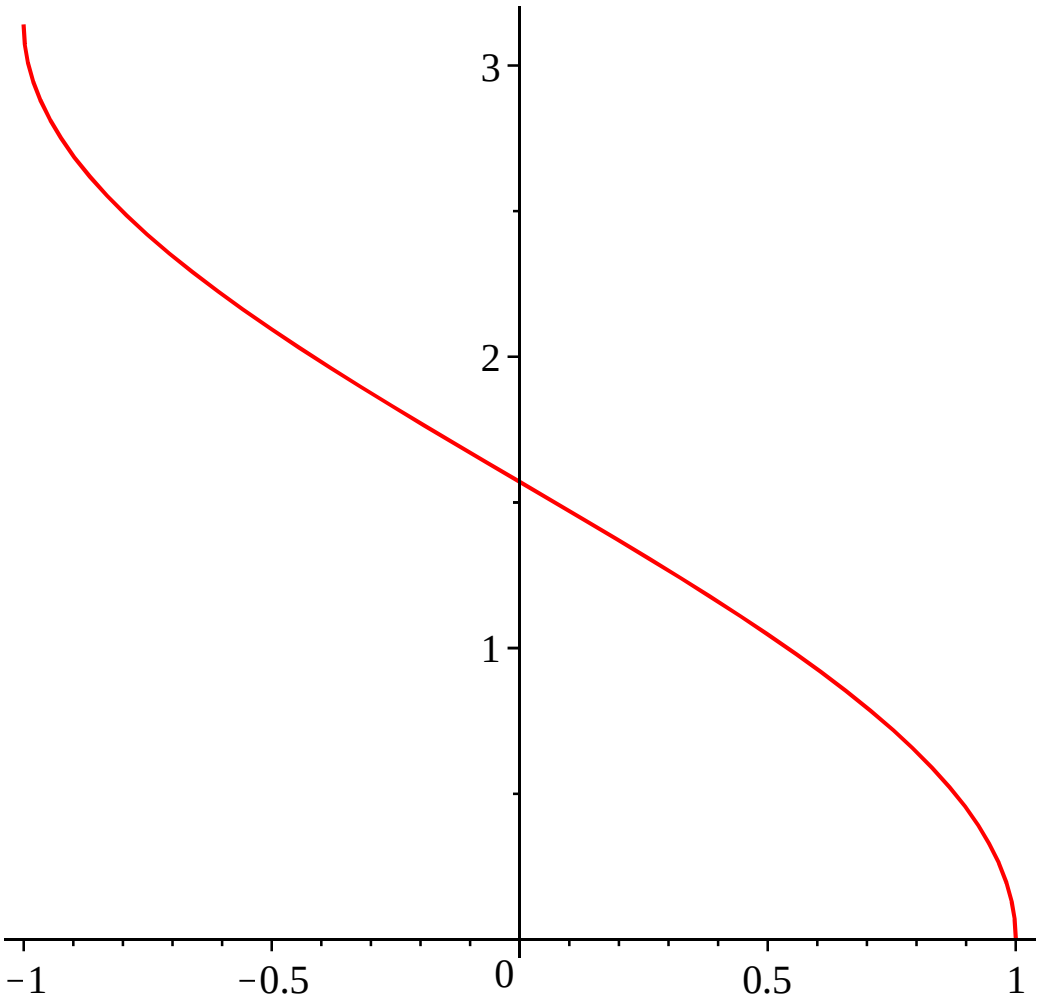
```
> plot([x*sin(x),x*cos(x)],x=0..25);
```



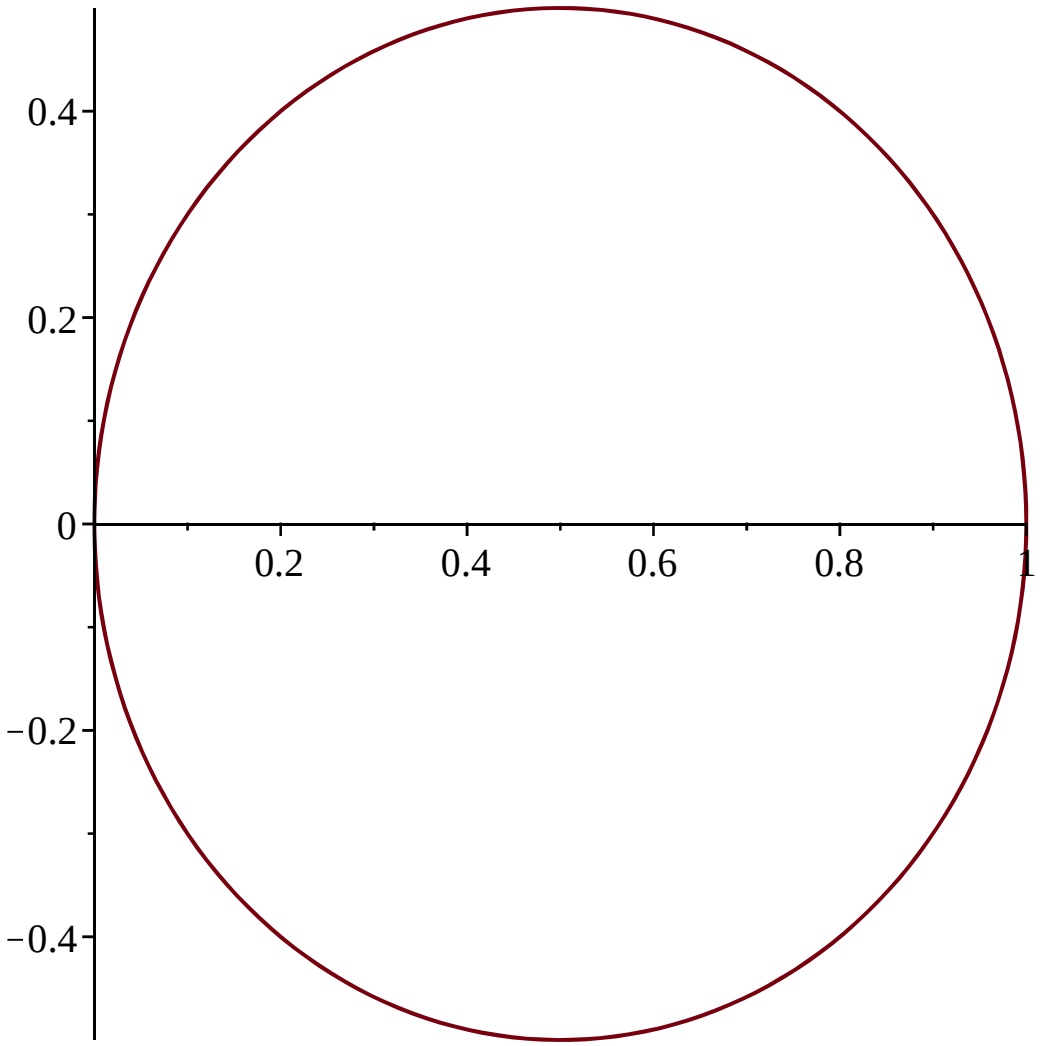
Coordoonés polaires

Les coordonnés polaires sont implenenté via l'option coords=polar, à noter que si cette option n'est pas spécifié le logiciel traitera la courbe comme une courbe paramétrique.

```
> plot([cos(t),t,t=0..Pi]);
```




```
> plot([cos(t),t,t=0..Pi],coords=polar);
```

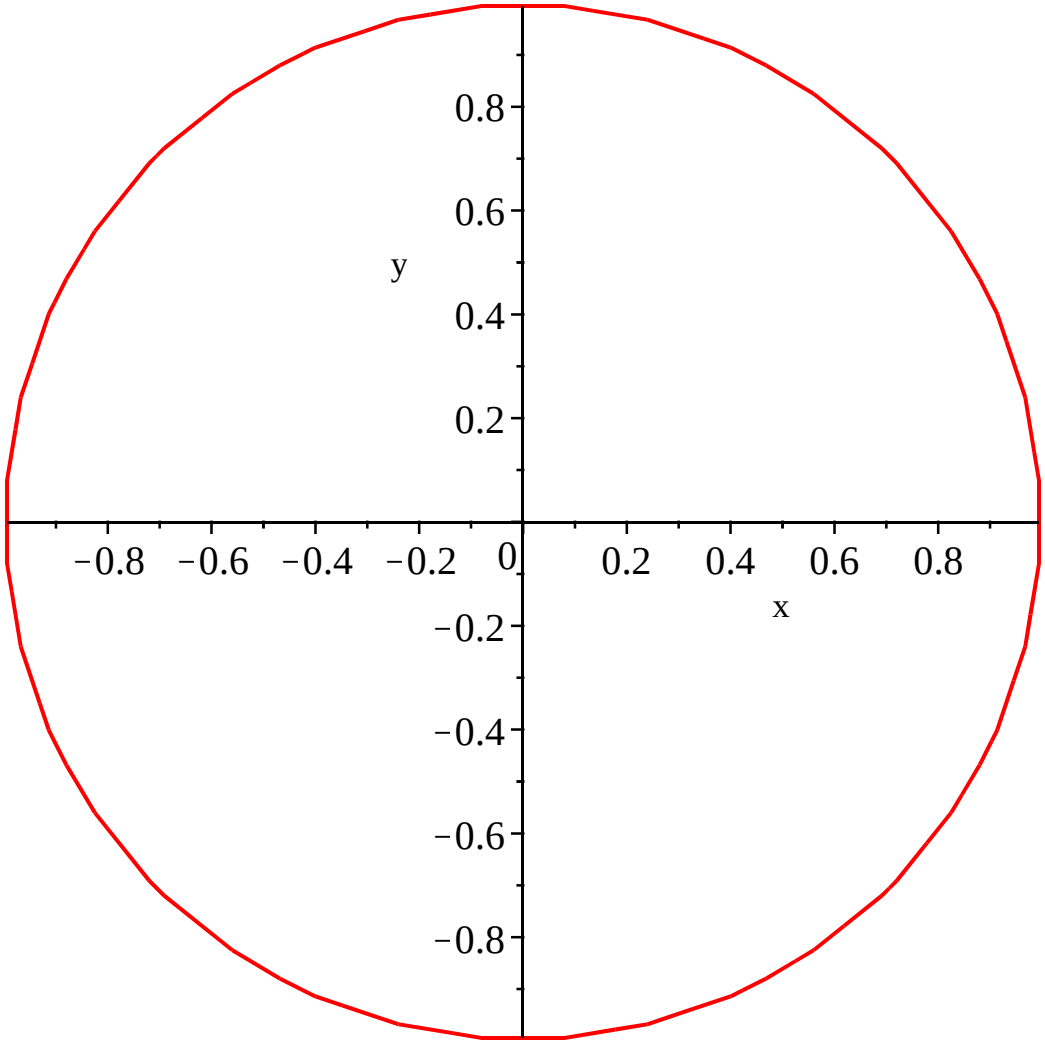


Bibliothèque plots

Commande implicitplot

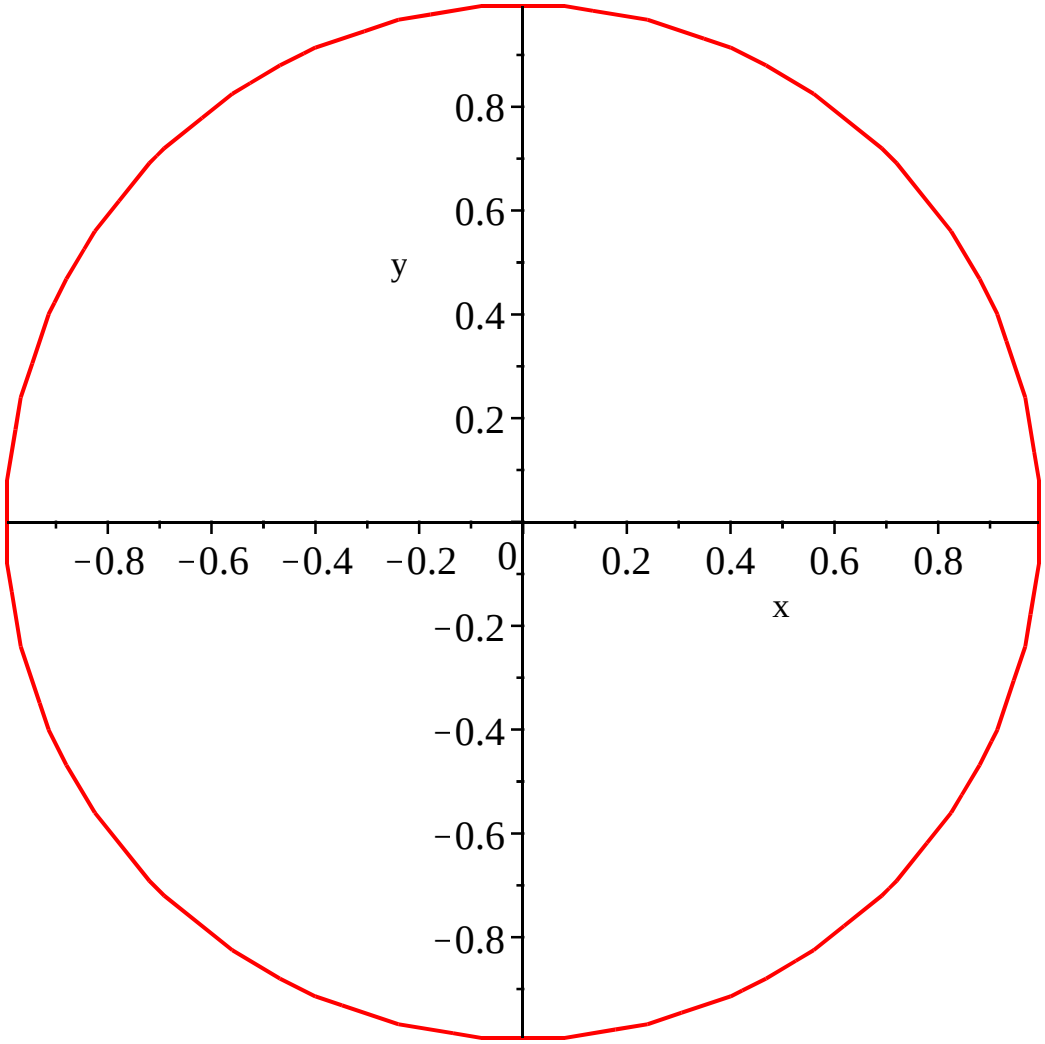
La commande implicitplot comme son nom l'indique permet de tracer des graphes de fonctions définies implicitement, la syntaxe est la suivante:
implicitplot(equation définissant une fonction,x=a..b,y=c..d);
A noter que l'intervalle pour l'axe des y devient obligatoire pour permettre de calculer les valeurs de la fonction.

```
> with(plots):
  implicitplot(2*x^2+y^2=1,x=-2..2);
Warning, the name changecoords has been redefined
Error, (in ProcessOptions) ArgList must have at least 3
entries, but only has 2
> implicitplot(x^2+y^2=1,x=-2..2,y=-2..2);
```



On peut éviter de charger la bibliothèque plots à chaque fois en utilisant la notation courte.

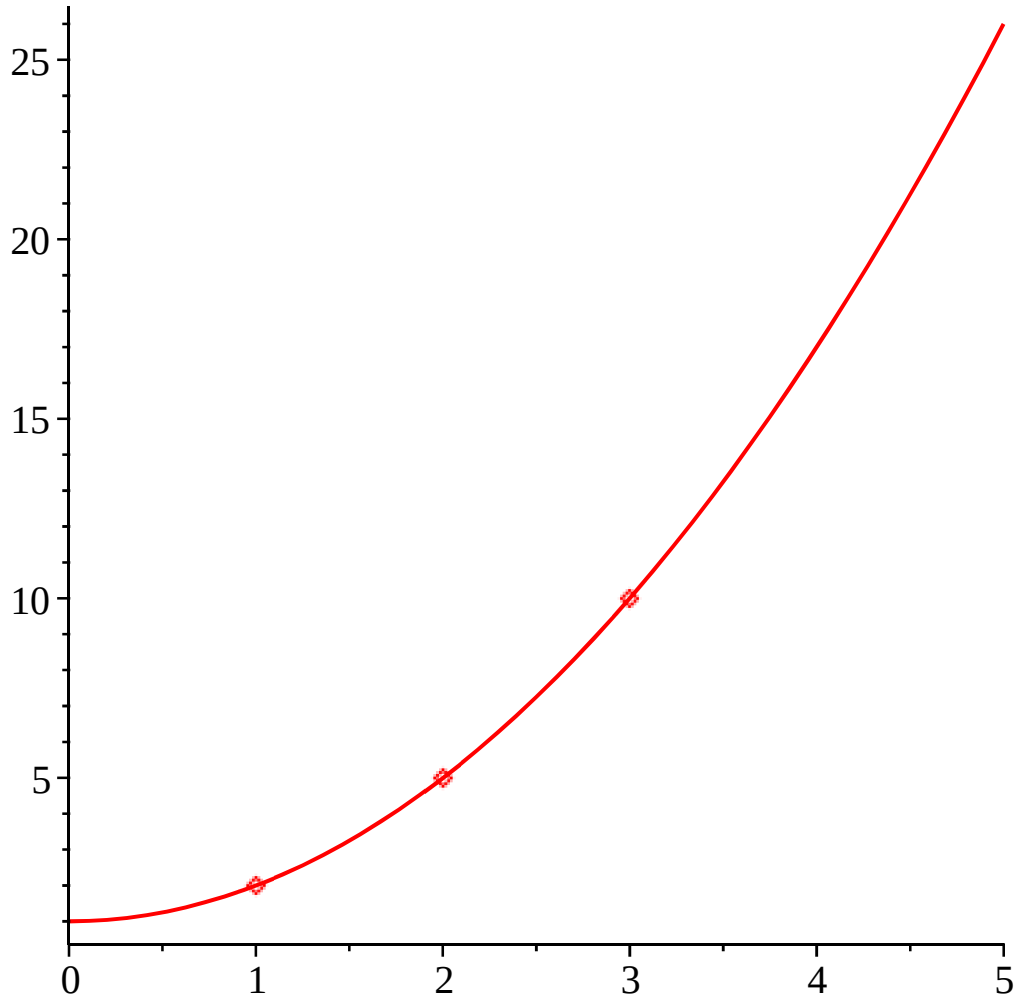
```
> restart:
  plots[implicitplot](x^2+y^2=1,x=-2..2,y=-2..2);
```



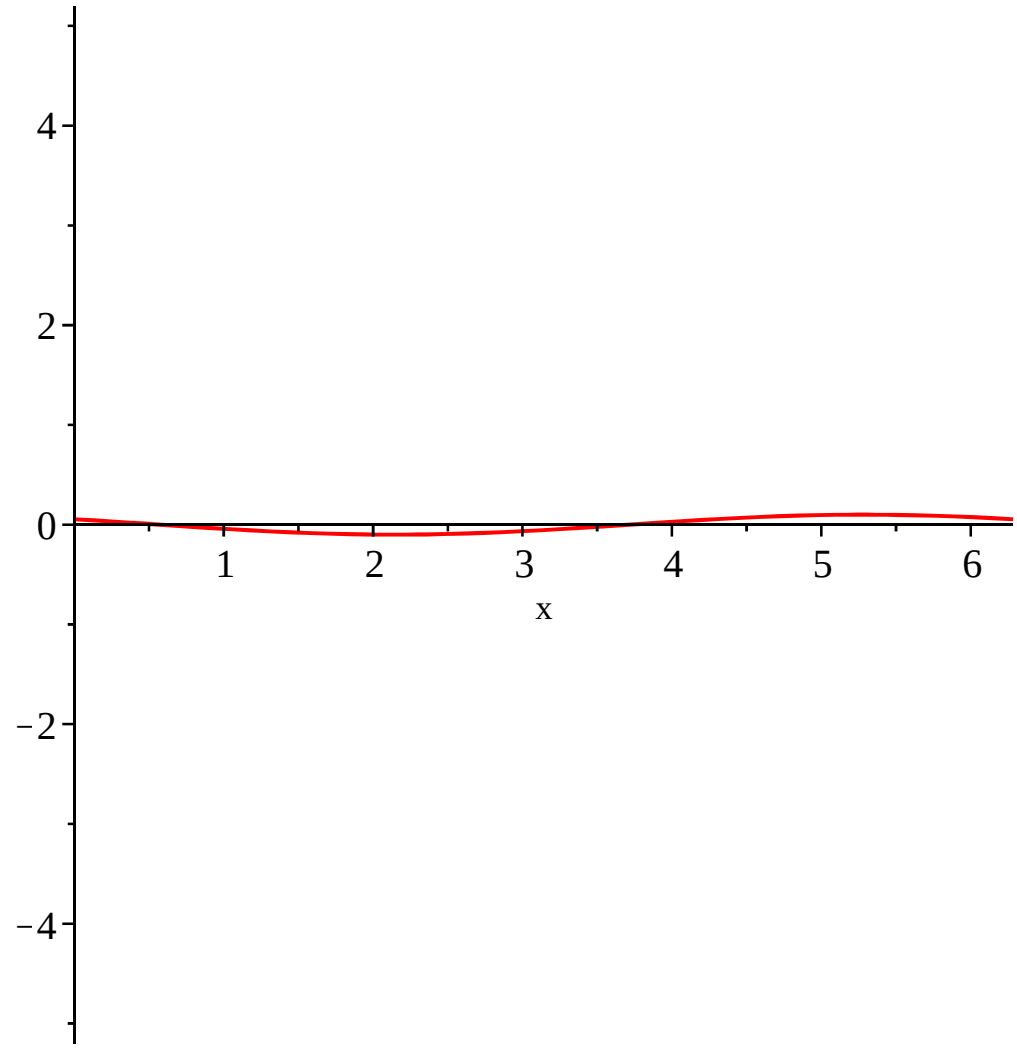
Commande display

La commande **display** permet d'afficher plusieurs graphes issues de commandes Maple différentes, la syntaxe générale est la suivante.
display(ListeDeGraphes, insequence=true/false, options)
Si le mot clé insquence est posé =true alors la commande crée une animation avec les différents graphes, si le mot clé insequence est posé =false alors les graphes sont affichés sur le même repère, par défaut la valeur est false.

```
> with(plots):
  Graphe1:=plot([[1,2],[2,5],[3,10]],style=point,symbol=
  diamond,symbolsize=15):
  Graphe2:=plot(x^2+1,x=0..5):
  display([Graphe1,Graphe2]);
Warning, the name changecoords has been redefined
```




```
> display([seq(plot((i/10)*cos(x+i),x=0..2*Pi),i=1..50)],
  insequence=true);
```



Exemple : Illustration de la méthode de la dichotomie

```
> f:=x->x^3+x^2-x+1:
> Dicho:=proc(alpha,beta,epsilon)
  local a,b,e:
  global m,i:
  if f(alpha)*f(beta)>0 then error "il n'existe pas de
  racine simple dans cet intervalle [%1,%2]",alpha,beta;
  else
  a:=alpha:b:=beta:e:=b-a:
  for i while e>epsilon do
  m[i]:=(a+b)/2.;
  if f(a)*f(m[i])<0 then b:=m[i] else a:=m[i]:
  fi:
  e:=b-a:od:fi:
  end:
> Dicho(-2.,-1.,10^(-5));
```

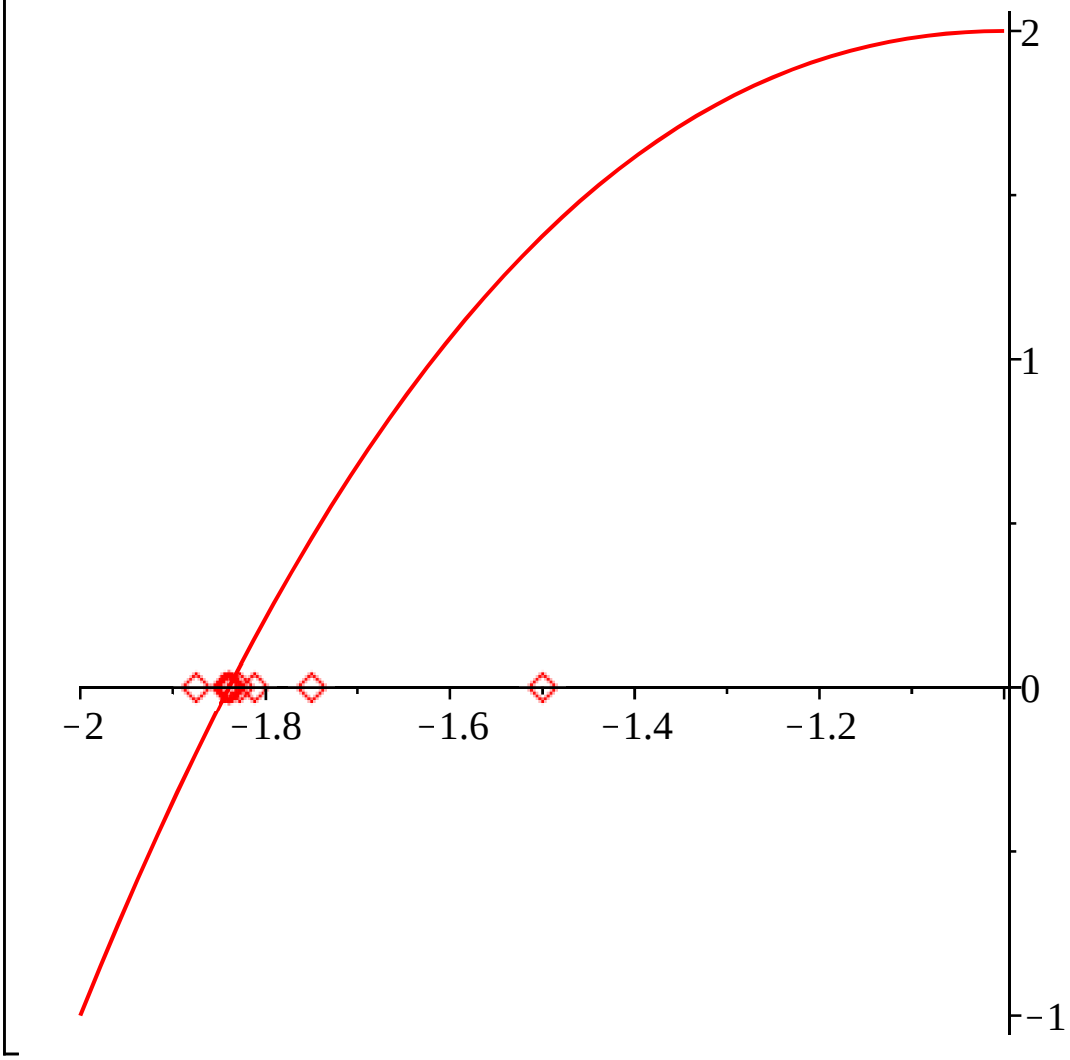
0.000007629

```
> eval(m);i;
```

table([1 = -1.500000000, 2 = -1.750000000, 3 = -1.875000000, 5 = -1.843750000, 4 = -1.812500000, 7 = -1.835937500, 6 = -1.828125000, 10 = -1.838867188, 11 = -1.839355469, 8 = -1.839843750, 9 = -1.837890625, 15 = -1.839263916, 14 = -1.839294434, 13 = -1.839233398, 12 = -1.839111328, 16 = -1.839279175, 17 = -1.839286804])

18

```
> g1:=plot([seq([m[j],0],j=0..i-1)],style=point,symbol=
  diamond,symbolsize=20):
  g2:=plot(f(x),x=-2...-1):
> display([g1,g2]);
```

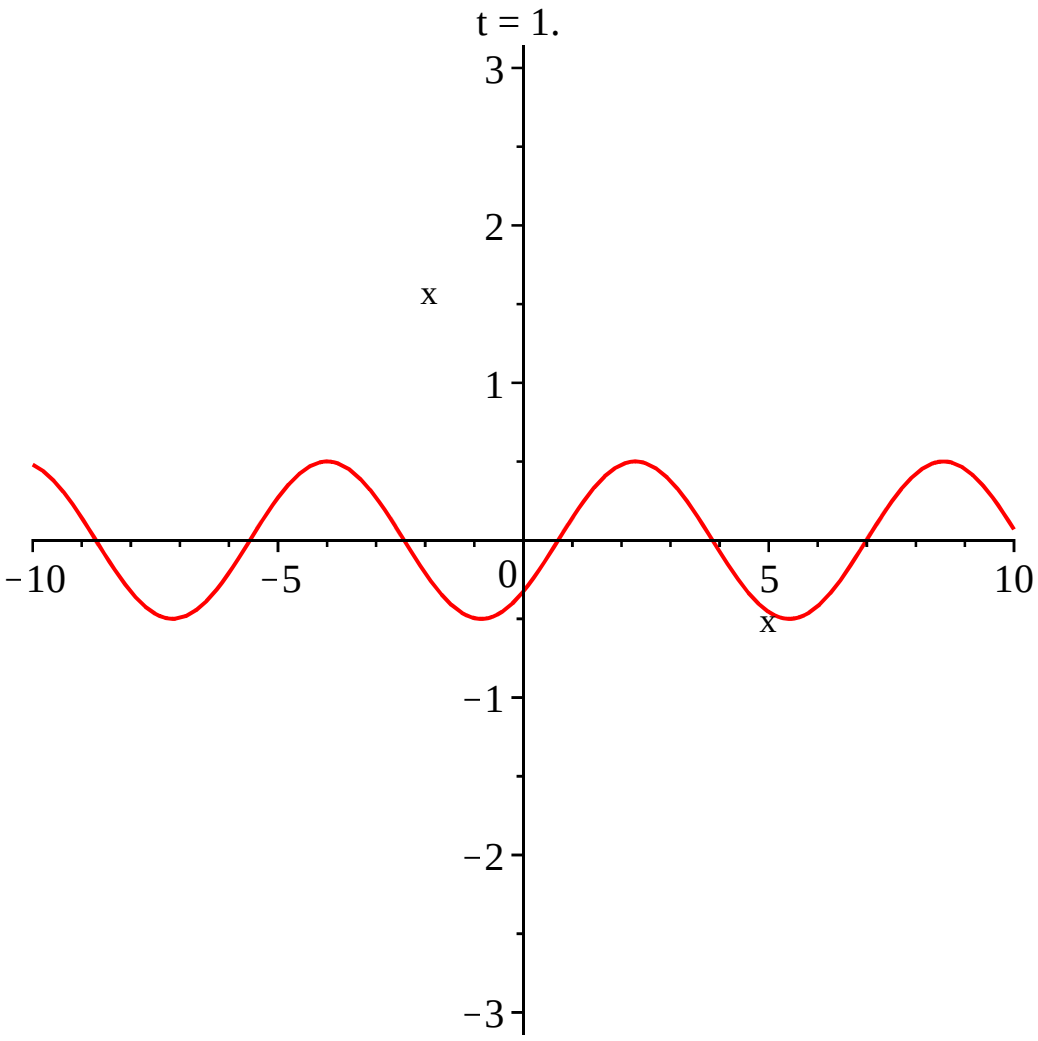


Représentations animées

Syntaxe : **animate(**une commande plot, t=a..b, options**)**
La fonction **animate** permet de construire un graphique animé d'une fonction dépendant d'un paramètre, l'exécution de la commande provoque le calcul des différents graphes paramétrés par

la variable t et de leurs affichage.
L'option frames permet de fixer le nombre d'images à créer, ce nombre est réparti de façon uniforme sur l'intervalle donné pour t.

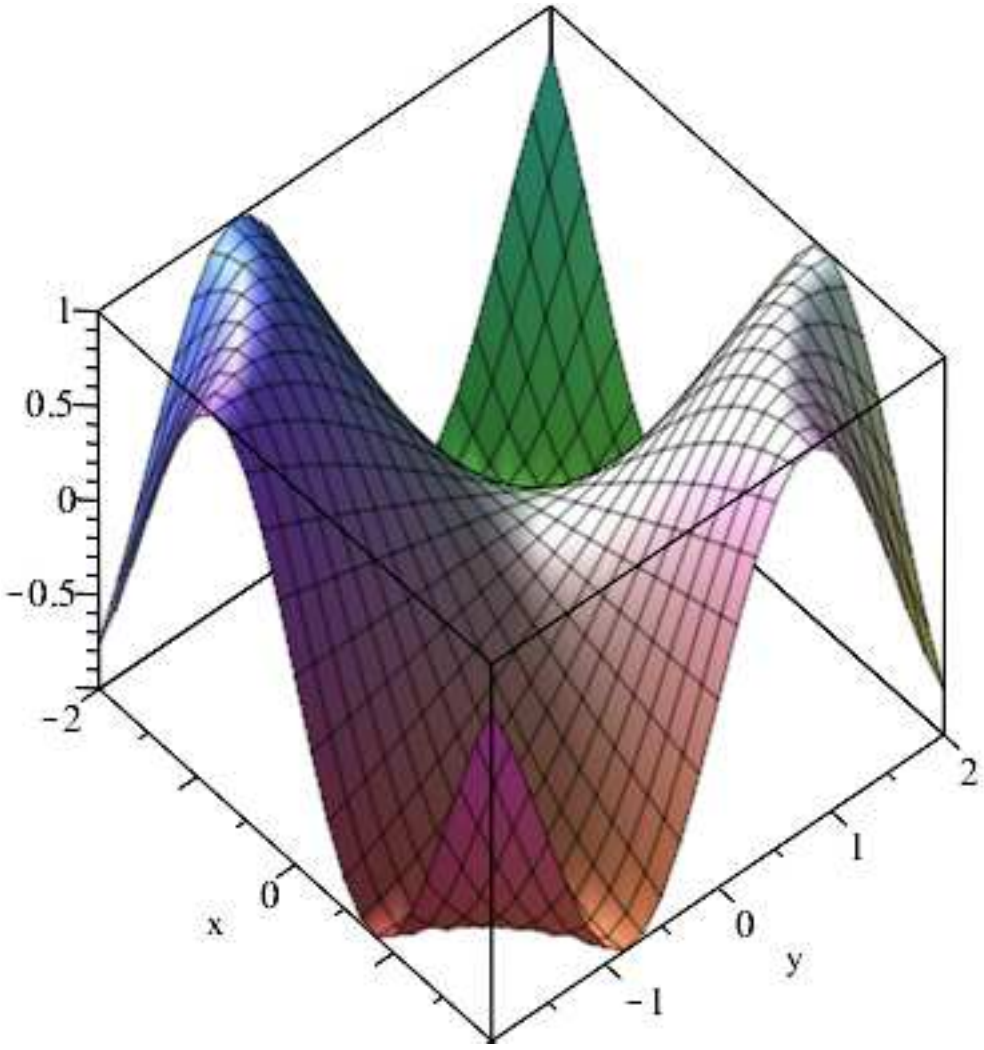
```
> with(plots):
  animate(plot,[(t/2)*cos(x+4*t),x,x=-Pi..Pi],t=1..4,frames=
  250);
```



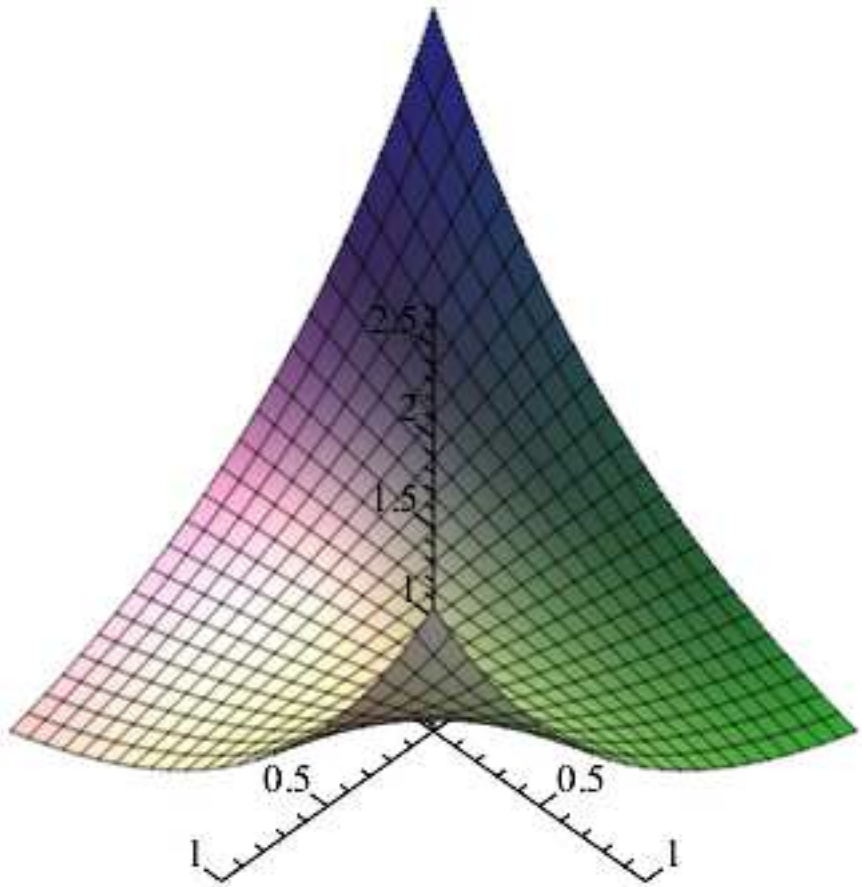
Graphe 3d

la commande **plot3d** permet de tracer des graphes de fonctions à deux variables et plus gérénaralement de créer des graphes en 3 dimensions, la syntaxe obéit aux même règles expliqués précédement dans la commande plot

```
> plot3d(sin(x*y),x=-2..2,y=-2..2);
```



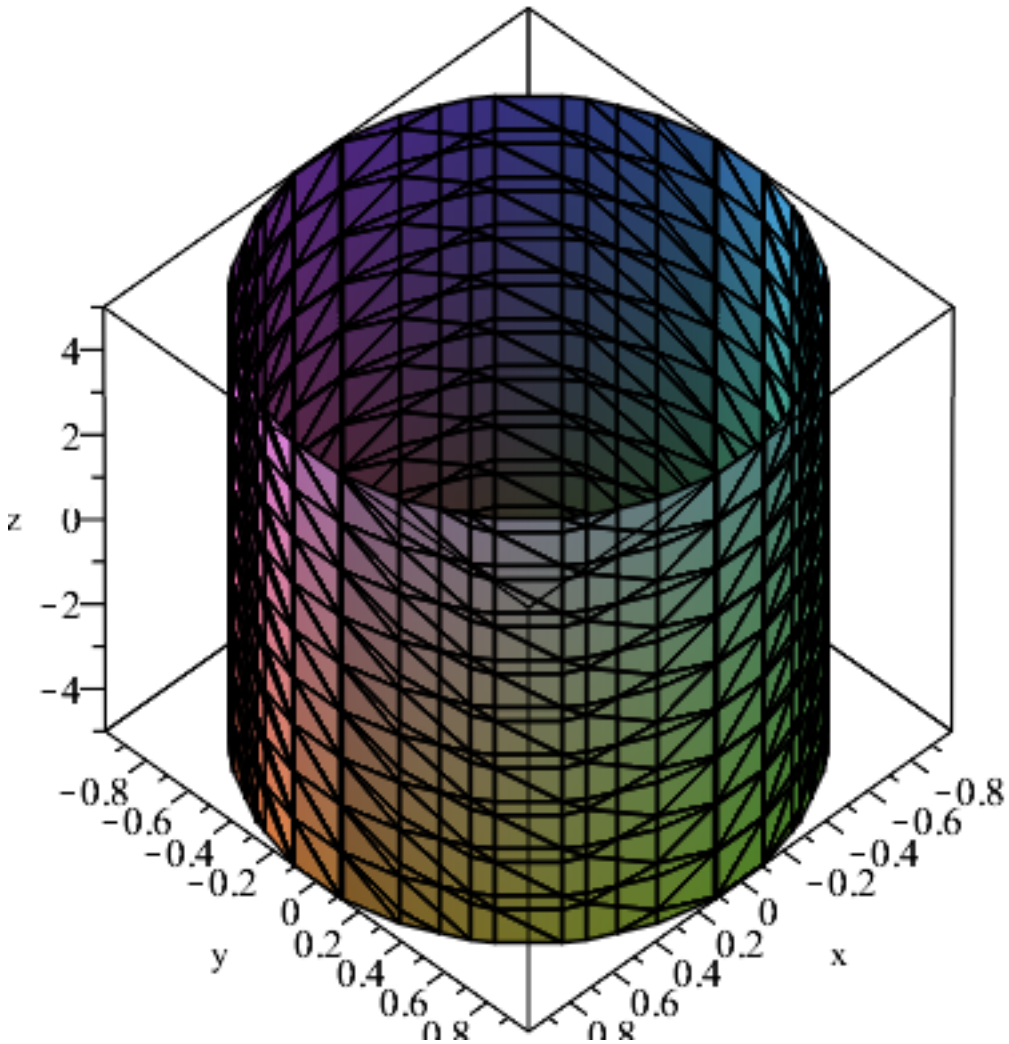

```
> plot3d(exp(x*y),x=-1..1,y=-1..1,axes=normal);
```



Commande implicitplot3d

La commande implicitplot3d permet de tracer des graphes de fonctions définies implicitement en 3 dimensions, la syntaxe est la suivante:
implicitplot3d(equation définissant une fonction,x=a..b,y=c..d,z=e..f);
A noter que l'intervalle pour l'axe des z devient obligatoire pour permettre de calculer les valeurs de la fonction.

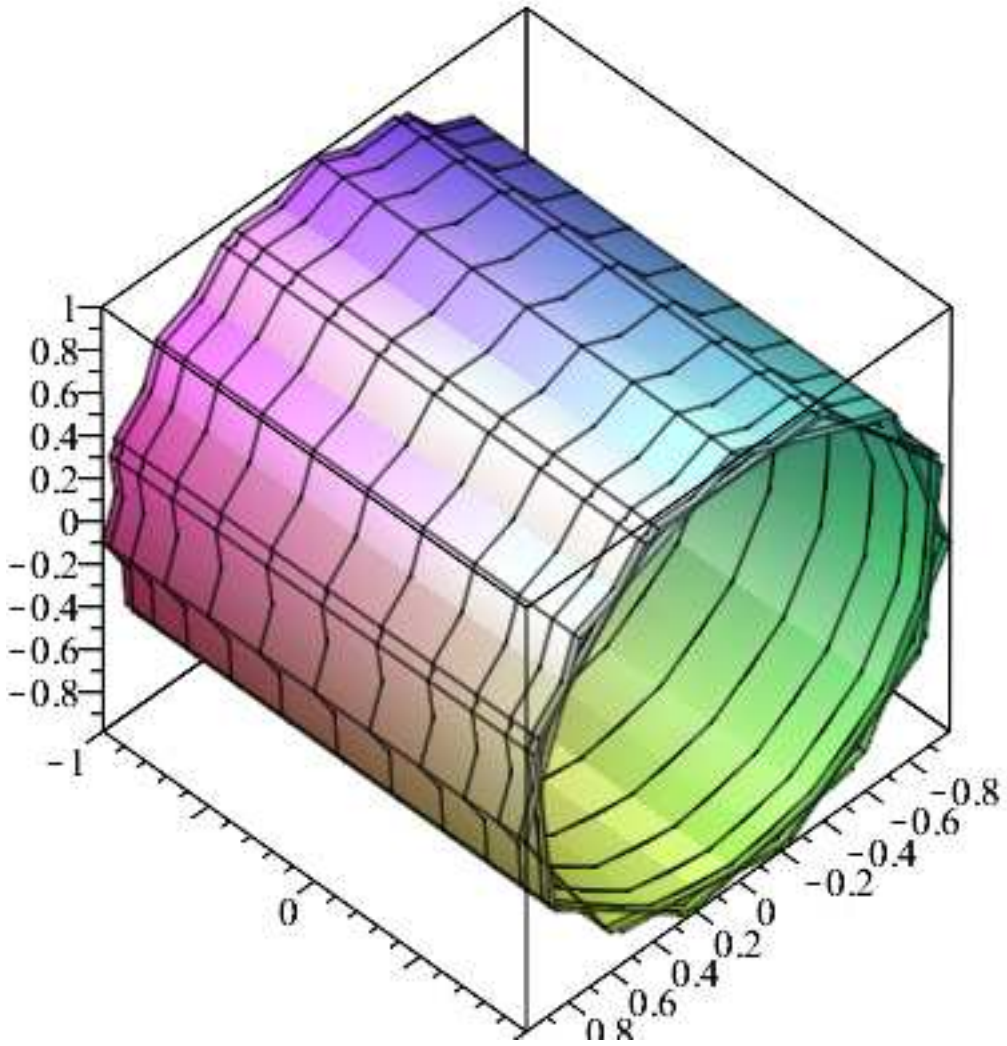
```
> implicitplot3d(x^2+y^2=1,x=-1..1,y=-1..1,z=-5..5);
```



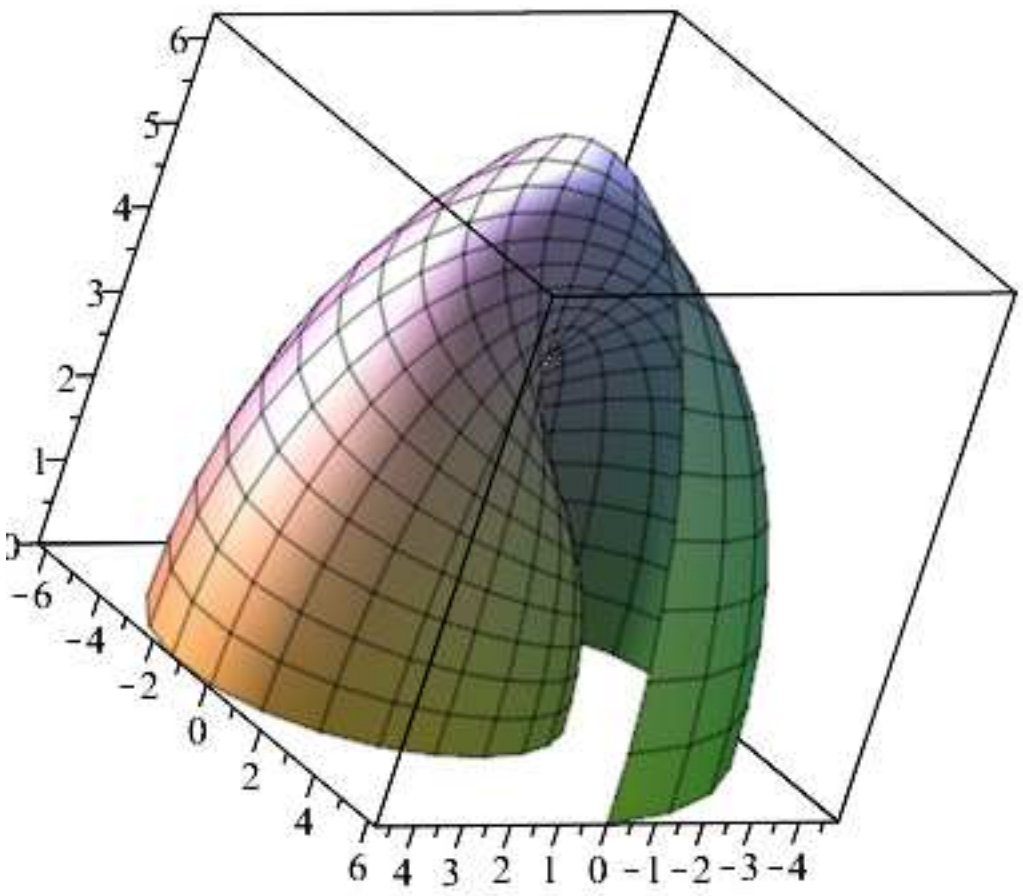
Graphes 3d en coordonées paramétriques

Il existe un équivalent 3d pour tracer des courbes paramétriques, dans ce cas, il faut donner 3 composantes et deux intervalles.

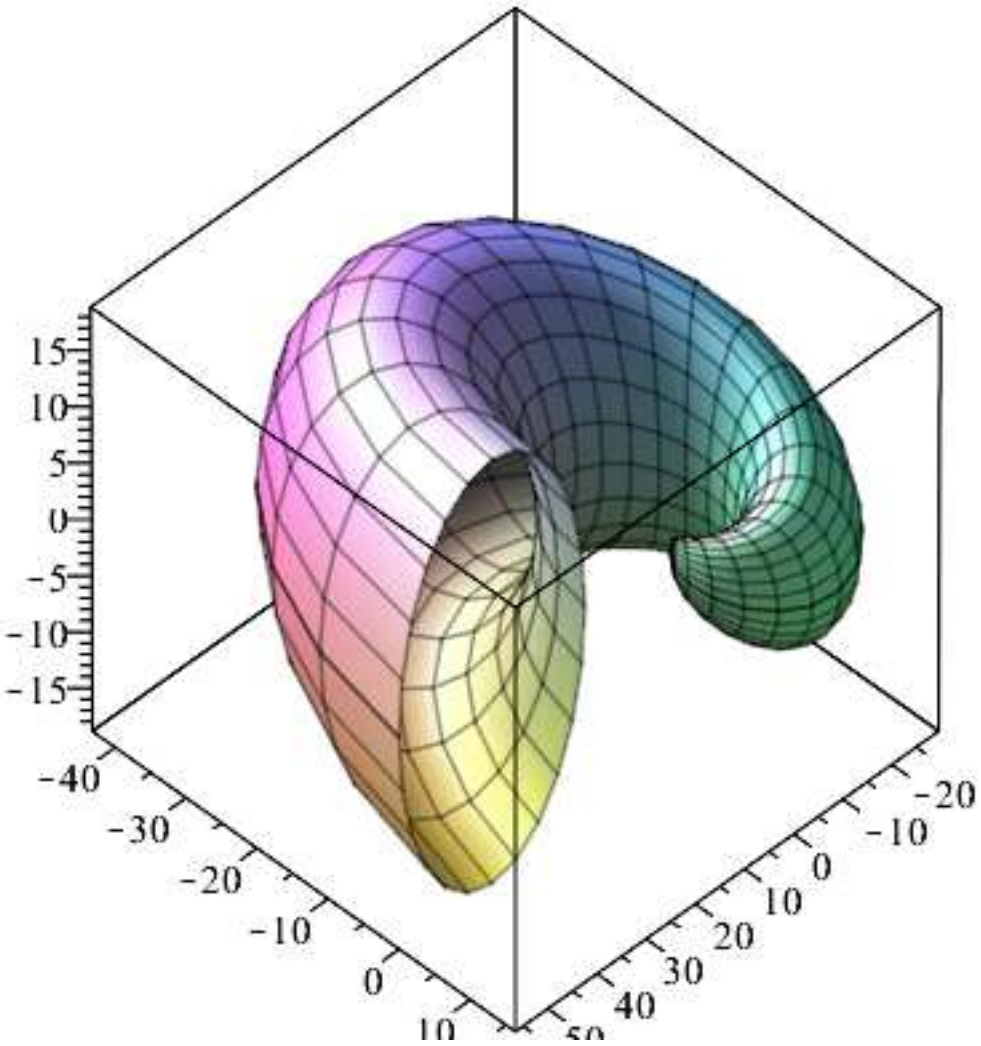
```
> plot3d([sin(x),cos(y),cos(x)],x=-10..10, y=-Pi..Pi );
```




```
> plot3d( [ x*sin(x)*cos(y), x*cos(x)*cos(y), x*sin(y) ],x=0..2*
  Pi, y=0..Pi );
```




```
> plot3d([(2+cos(r))*3*s*cos(s),(2+cos(r))*3*s*sin(s),3*s*sin(r)
  ],r=0..2*Pi,s=0..2*Pi);
```


15
10
5
0
-5
-10
-15
-40
-30
-20
-10
0
10
-20
-10
0
10
20
30
40