Chapitre M VI

Le graphisme

J.M. Janod

Maple offre des possibilités graphiques très performantes et variées. Le lecteur trouvera ici les éléments indispensables qu'il pourra compléter en consultant l'aide en ligne.

Introduction : les objets graphiques

Les graphiques sous Maple

Il existe plusieurs procédures Maple permettant les représentations graphiques. La procédure **plot** permet une représentation graphique dans le plan, la procédure **plot3d** permet une représentation en trois dimensions. Ces procédures sont directement accessibles et déterminent par défaut le nombre de points, la position des axes, les échelles, les couleurs (éléments qu'on peut aussi fixer soi-même).

Les autres procédures sont accessibles à partir de la bibliothèque **plots** qu'on peut appeler avec la commande **with(plots)** ou pour une procédure particulière **with(plots,procédure particulière)**.

Toutes les procédures graphiques de Maple sont évidemment des fonctions et retournent une structure de données graphiques dont le type est **PLOT** (respecter les majuscules) ou **PLOT3D**. Elles fonctionnent comme les autres procédures de Maple. Lorsque l'appel de la procédure est suivi du point virgule, la structure graphique est transmise à l'écran par l'intermédiaire d'un pilote graphique entraînant son affichage. On peut aussi l'enregistrer dans une variable Maple dont le type sera PLOT. Un graphique est donc un objet particulier (PLOT) construit à partir de l'appel d'une fonction. Sa représentation à l'écran est une insertion de cet objet graphique, conforme au système d'exploitation Windows, qu'on pourra modifier par les méthodes classiques applicables aux objets graphiques prévues par Maple, mais on peut aussi transférer le graphique dans un autre logiciel (winword par exemple) en utilisant le presse-papiers (copier, coller).

La notion d'objet

En informatique, il existe une notion assez récente, l'objet. L'objet est une notion abstraite comme les notions de type ou de fonction. On peut distinguer schématiquement dans un objet la définition d'une structure de données particulière comprenant les différents éléments mémorisables, et les méthodes permettant d'une part leur création en mémoire centrale et d'autre part les méthodes nécessaires à leur utilisation.

Dans la programmation classique, celle qu'on a utilisée jusqu'ici, on définit les structures de données par des types (réel, mais aussi tableau ou table), et on implante des fonctions ou des procédures ayant pour paramètres formels les structures définies. Dans la programmation objet on définit en une même entité les structures de données et les méthodes qui lui sont applicables. Parmi ces méthodes, il existe toujours une méthode dite construction permettant de définir un

représentant ou une instance de l'objet. On déclare des identificateurs par le nom de l'objet, on le crée en mémoire par l'appel de la méthode constructeur, et on le manipule par les fonctions ou procédures définies dans l'objet.

Un exemple classique d'objet est la fenêtre Windows, la structure de donnée correspondante est l'ensemble de ses caractéristiques (largeur, hauteur, emplacement...). Les méthodes liées aux fenêtres sont : le déplacement, sa réduction, sa restauration, son agrandissement etc.

Une des caractéristiques de la programmation objet est de définir de nouveaux objets en particularisant ou complétant des objets existants, le nouvel objet héritant alors de toutes les propriétés et méthodes de son ancêtre. Une autre possibilité est d'insérer des objets les uns dans les autres en héritant pour chacun de ses méthodes.

En Maple les représentations graphiques sont des objets. Pour manipuler sous Maple un graphique, il suffit de cliquer sur le graphe pour le sélectionner. Une barre d'outils adaptés à l'objet graphique correspondant apparaît. Chaque bouton permet l'accès à une méthode prévue pour l'objet représenté.

Problème de syntaxe

Il faut distinguer deux cas, selon qu'on donne la définition de la courbe sous forme d'une expression, ou sous forme d'une fonction informatique. Dans le premier cas il faut indiquer le nom de la variable avec le cas échéant l'intervalle de variation, dans le second cas la variable est le paramètre de la fonction et ne doit donc pas être précisée.

D'autre part il existe pour chaque commande graphique un certain nombre d'options permettant de modifier les valeurs par défaut. Les options ainsi que leurs valeurs par défaut seront détaillées dans un paragraphe et ne feront pas dans un premier temps l'objet d'une description détaillée.

Graphique à deux dimensions

Représentation d'une expression

On peut représenter la courbe correspondant à une expression,à une liste ou à un ensemble d' expressions. Il faut alors préciser le nom de la variable. Par défaut l'intervalle de variation est de -10 à 10 et correspond à l'axe horizontal. On peut aussi fixer l'intervalle sur l'axe vertical (axe des ordonnées). la syntaxe est

plot(expression en lambda, lambda,options)

ou

plot(expression en lambda, lamda=valeur de début..valeur de fin,options)

ou

plot([expr1,expr2..],lambda,options)

On peut préciser
l'intervalle pour lambda
l'intervalle des ordonnées
et les options (ci-dessous)

La variable spécifiée doit être entre quotes si elle a une valeur, par contre les expressions sont évaluées. Les intervalles des axes sont des constantes entières, réelles ou rationnels ou des variables s'évaluant comme telles.

```
> b:=-0.5:x:=1;plot([x,'x'-b],'x'=-1..1,-2..2);# le
  dernier argument fixe les ordonnées de -2 à +2
  x:='x';
```

Dans l'exemple ci-dessus, le graphe obtenu correspond à la droite horizontale (y=1)et à la droite y=x+1/2, qui résultent des expressions x et 'x'-b, avec x et b affectés respectivement des valeurs 1 et -1/2. On peut constater que :

- l'identificateur précisant la variation doit être entre quote lorsqu'il est affecté d'une valeur car plot ne saura pas tracer les expressions sachant que "1(valeur de x) varie de -1 à 1" est une information incompréhensible entraînant logiquement une erreur.
- l'expression 'x'-b n'est représentable que si b a une valeur numérique (ici -1/2).
- les expressions sont d'abord évaluées comme tous les paramètres effectifs puis les calculs de l'ordonnée des points du graphe sont obtenus en substituant à x (là où cette variable n'a pas été évaluée), les valeurs de l'intervalle -1..1, dans les expressions précédemment évaluées. Ainsi la première, x, est évaluée à 1 donc à une constante indépendante de x, d'ou la droite horizontale. La deuxième est évaluée à x+1/2 car x est entre quotes, sans quotes son évaluation serait 3/2 et on obtiendrait encore une droite horizontale.

```
> plot([seq('x'^t,t=-1..3)],'x'=-2..2,-3..3);#on obtient
  cinq courbes x^-1,x^0(=1),x^1,x^2,x^3
```

```
> plot([seq(1/sqrt(2*Pi)/sigma*exp(-x^2/2/sigma^2),sigma=
  1..3)],x=-3..3);#densités d'une loi normale centrée
  d'ecart-type 1,2,3
```

Cet exemple montre qu'on peut générer une liste d'expressions puis obtenir leur répresentation.

Représentation des fonctions

La commande est toujours la commande **plot**. L'expression est remplacée par une fonction à un paramètre. Compte tenu que le paramètre de la fonction est automatiquement la variable, la syntaxe de la commande interdit toute spécification de cette variable même dans l'intervalle de variation. Comme pour les expressions, la fonction ne doit pas comporter de constante sans valeur. On a donc les emplois suivants :

plot(x->x^2,options) ou plot(f,options)

avec f:=x->.. ou f:=proc(x)...end;

ou encore

plot(x->x^2,début..fin,vdébut..vfin,options) ou plot(f,début..fin,vdebut..vfin,options)

et aussi

plot({f,cos,x->...},début..fin,vdébut..vfin,options)

On peut toujours utiliser l'expression f(x) même si f est définie comme une fonction informatique sous réserve que f(x) puisse être évaluée lors de l'appel de plot (voir l'exemple ci-dessous).

plot(f(x),x:=..)

```
> plot([cos,sin,x->x],-Pi..Pi);
```

```
> f:=proc(x)
  if x>0 then RETURN(x+1)
         else cos(x);
  fi:
  end;
```

```
> plot(f,-1..1);
```

```
> plot(f(x),x=-1..1);# f(x) ne peut pas être évalué (x>0)
  lors de l'appel.
Error, (in f) cannot evaluate boolean
```

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

Représentation d'une liste de points

Pour représenter une liste de points, il suffit de remplacer l'expression par une liste de listes de points. Sans option les points sont reliés par des segments, on peut préciser l'option style par point si on veut uniquement les points. La syntaxe est:

plot(liste de listes de point,style=point);

```
> plot([[-1,-1],[-1,1],[1,1],[1,-1],[-1,-1]]);
```

```
> L:=[[-1,-1],[-1,1],[1,1],[1,-1],[-1,-1]]:plot(L,style=
  point);
```

Les courbes paramétriques

Lorsque l'abscisse x et l'ordonnée y d'un point varient en fonction du temps par exemple, on trace la courbe correspondant au point (x(t),y(t)) quand t varie. On dit que la courbe est paramétrée et elle représente l'ensemble des valeurs possibles des deux quantités. Par exemple (sin(t),cos(t)) pour t variant de 0 à 2Pi est un cercle. La commande **plot** permet d'obtenir ce genre de représentation, sa syntaxe est:

plot([exp en t, exp en t, t=début..fin])

attention aux crochets

ou avec des fonctions

plot([fonction1,fonction2,début..fin],option)

```
> plot([cos(t),sin(t),t=0..2*Pi],scaling=constrained);
  #l'option scaling permet des axes orthonormés
```

```
> S:=t->100/(100+(t-Pi/2)^8):
  R:=t->S(t)*(2-sin(7*t)-cos(30*t)/2):
  plot([R,t->t,-Pi/2..3/2*Pi],coords=polar,axes=none,color=
  green,numpoints=100);
```

Les options

Les options sont séparées par des virgules et ont pour syntaxe

nom de l'option=valeur

Les options sont les suivantes (la valeur par défaut est en premier) :

- **adaptive=**true (Maple adapte au mieux le nombre de point) ou false.
- **style=**line ou style=point les points sont reliés par une ligne ou non.
- **linestyle=**1 (trait continu) ou 2 (trait formé de tirets) ou 3 (trait formé de points) ou 4 (trait formé de tirets-points).
- **thickness=**1 ou 2 ou 3 selon l'épaisseur du trait souhaité.
- **axes=**NORMAL ou BOXED (dans un cadre) ou FRAMED (axe vertical à droite) ou NONE (sans axe).
- **scaling=**UNCONSTRAINED (les échelles des axes sont choisies au mieux) ou CONTRAINED (les axes sont orthonormés) .
- **xtickmarks=**n permet d'afficher n valeurs sur axe horizontal n peut être un entier ou une liste de valeurs.
- **ytickmarks=**n même chose sur l'axe vertical.
- **title=**` le titre du graphe`
- **color=**red ou black (couleur noir pour toutes les courbes) voir l'aide pour les couleurs.
- **labels=**[`nom de l'axe`,`nom de l'autre`]
- **symbol=**CROSS ou ... (voir l'aide) permet de changer le symbole pour les points.
- **discont=**false ou true (donne une représentation plus adaptée sur des points de discontinuité)
- **coords=** par défault cartésien permet des systèmes d'axes autres notamment polaire (polar)
- **numpoint=**50 ou un nombre entier de points souhaités; peut être modifié automatiquement si adaptive=true.
- **view=[**xmin..xmax,ymin..ymax**]** précise l'amplitude des coordonnées affichées.

Représentation des fonctions implicites

Pour représenter des courbes, dans le plan, définies par une équation du type f(x,y)=0 ou f(x, y)=constante, il n'est pas toujours possible d'expliciter la courbe par une équation (explicite) du type y=g(x) permettant le calcul de l'ordonnée en fonction de l'abscisse. Les mathématiciens ont cherché là encore des techniques de calcul permettant le tracé de ces courbes, (comme d'ailleurs pour le tracé du graphe des fonctions à une variable, la génération précédente étudiait la continuté, la dérivabilité, construisait les tableaux de variation puis à l'aide de ces informations traçait l'allure de la courbe).
Maple permet par la commande **implicitplot** de la bibliothèque **plots** d'obtenir le tracé d'une courbe définie implicitement par une équation.

Il faut appeler la bibliothèque plots :

with(plots) ou with(plots,implicite)

La syntaxe est :

implicitplot(f(x,y),x=debut..fin,y=vdebut..vfin,options)

f(x,y) étant une expression ou l'expression de la fonction

La commande trace la courbe implicite définie par f(x,y)=0 pour les intervalles fixés

ou

implicitplot(f(x,y)=c,x=début..fin,y=vdebut..vfin,options)

ou en utilisant une fonction de deux variables

implicitplot(nom de la fonction, debut..fin, vdébut..vfin,options)

```
> z:=1:with(plots,implicitplot):implicitplot(x^2+y^2-z,x=
-1..1,y=-1..1);
```

```
> c:=(x,y)->x^2+y^2-1:implicitplot(c,-1..1,-1..1);
```

```
> restart;
```

L'animation

On peut aussi obtenir une animation visuelle mettant en évidence la déformation d'une courbe en fonction d'une constante de l'expression. C'est la commande **animate**. Il faut préciser l'expression, la variable, le paramètre avec son intervalle de valeurs. La commande trace une représentation correspondant à la première valeur de la constante. On peut alors en sélectionnant le graphe, déclencher le processus grâce au menu surgissant en appuyant sur la touche droite de la souris. Ce menu propose play pour l'animation, et d'autres options (cycle, en continue, le réglage de la vitesse). Pour arrêter on clique sur le bouton droit de la souris et on choisit stop. On peut aussi utiliser la barre de menu qui apparaît lors de la sélection du graphe.

```
> with(plots,animate):animate(1/sqrt(2*Pi)/sigma*exp(-
x^2/2/sigma^2),x=-4..4,sigma=0.5..3);
```

Les courbes de niveau

Les courbes de niveau, bien connues des géographes, représentent l'ensemble des points du plan (x,y) dont les altitudes h(x,y) sont les mêmes, chaque courbe étant définie par h(x,y)=c. On peut donc de la même manière pour toute expression ou fonction à deux variables définir les courbes de niveau. La commande **contourplot** de la bibliothèque **plots** de Maple, trace directement ces courbes h(x,y)=c pour différentes valeurs de la constante c (8 par défaut). La syntaxe est:

with(plots) ou with(plots,contourplot)

Appel de la bibliothèque

contourplot(expr(x,y),x=a..b,y=c..d)

Les intervalles doivent être précisés

L'option contours=n pour obtenir n courbes

ou avec une fonction

contourplot(f,début..fin,vdébut..vfin)

f une fonction à 2 variables

```
> plots[contourplot](x^2+y^2,x=-10..10,y=-10..10,color=
  black);
```

```
> f:=(x,y)->x^(alpha)*y^(beta);# fonction de production de
  Cobb-Douglas
```

```
> alpha:=1/2:beta:=1/2:with(plots):
  contourplot(f,0..1,0..1,color=black);# Isoquantes de la
  fonction de production de Cobb-Douglas
```

Variable de type PLOT et Affichage

Les commandes graphiques comme plot appellent une fonction PLOT (remarquer les majuscules) avec des paramètres effectifs construits à partir des paramètres effectifs de la commande graphique utilisée, sans évaluer la fonction PLOT. Par exemple plot([0,2],[2,0]) construit l'expression de la fonction suivante

PLOT(CURVES([[0, 2], [2., 0.]],COLOUR(RGB,1.0,0,0)))

Si on a fait suivre la commande plot d'un point virgule (plot([0,2],[2,0]);) Maple évalue et transmet l'expression de la fonction PLOT ci-dessus au pilote graphique qui crée alors le graphisme. On peut donc mémoriser l'objet graphique dans une variable informatique G donc la valeur est PLOT(CURVES([[0, 2], [2., 0]],COLOUR(RGB,1.0,0,0))). G est alors une fonction qui est du type PLOT. Son évaluation ou l'utilisation de print(G) crée le graphisme. On peut aussi directement taper PLOT(CURVES([0,2],[2,0]],COLOUR(RGB,1.0,0,0))); mais la syntaxe est très vite complexe, par contre on peut programmer ainsi de véritables procédures graphiques. Le lecteur pourra consulter des ouvrages plus spécialisés dans ce domaine.

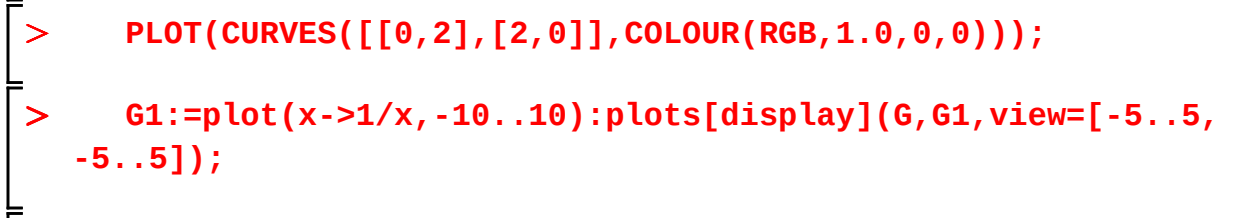
La mémorisation des graphismes dans des variables Maple permet d'obtenir leur affichage simultanée grâce à la commande **display** de la bibliothèque **plots**. La syntaxe est:

display(liste des variables graphiques,options)

Signalons l'option view qui permet une représentation limitée du graphe des variables sans recalcul (i.e. limite la représentation de PLOT(...)).

```
> lprint(plot([[0,2],[2,0]]));
PLOT(CURVES([[0, 2.], [2., 0]],COLOUR(RGB,1.0,0,0)))
```

```
> G:=plot([[0,2],[2,0]]);whattype(G);type(G,PLOT);G;
```



```
> PLOT(CURVES([[0,2],[2,0]],COLOUR(RGB,1.0,0,0)));
```



```
> G1:=plot(x->1/x,-10..10):plots[display](G,G1,view=[-5..5,
  -5..5]);
```


L'exemple suivant trace la tangente à la courbe 1/x au point d'abscisse 1, le calcul de la dérivée,

et de la fonction tangente étant calculée par Maple. On utilise la fonction D(f) (1) qui calcule la dérivée de f au point 1, puis solve qui resout a*x+b=f(1) avec a=D(f)(1).

```
> restart:with(plots):f:=x->1/x:DTG:=x->a*x+b:a:=D(f)(1);b:=
  solve(DTG(1)=f(1));DTG(x);G:=plot(f):DG:=plot(DTG):display(
  [G,DG],view=[-5..5,-5..5]);
```

Graphique à trois dimensions

Le graphique à trois dimensions est une représentation graphique dans notre espace. Chaque point est repéré par trois données (x,y,z), x étant l'abscisse, y l'ordonnée et z la cote, x,y étant un point du plan. Ainsi une surface est la donnée pour tout point du plan (x,y) d'une cote z définie par exemple par une équaton du type z=f(x,y). On peut aussi considérer la représentation d'un point dont chaque coordonnée est définie par rapport à une quatrième variable, le temps par exemple. Dans ce cas le point est défini par x=f(t), y=g(t), z=h(t). On dit que c'est une représentation paramétrique (paramètré par rapport au temps). On peut aussi chercher l'ensemble des points vérifiant une relation du type R(x,y,z)=constante. Il peut être difficile de trouver la fonction f telle que z=f(x,y) si R(x,y,z)= constante. On dit alors que la surface correspondante est définie de manière implicite (z=f(x,y) n'étant pas explicité).

Le principe est le même que pour des représentations à deux dimensions, mais au lieu d'utiliser la commande plot, on utilise la commande plot3d. Elle s'utilise soit avec des expressions soit avec des fonctions soit avec des listes ou ensembles d'expressions ou de fonctions, comme à deux dimensions. La syntaxe est donc la suivante

plot3d(expression en x et y, x=xdébut..xfin, y=ydébut..yfin, options)

options et bornes facultatives

ou plus simple

plot3d(expression en x et y, x, y,options)

options facultatives

et pour les fonctions

plot3d(nom de la fonction à deux variables, intervalle, intervalle, options)

les intervalles sont facultatifs ainsi que les options

on peut aussi utiliser une liste de listes de points.

Les options reprennent les mêmes que dans le graphique à deux dimensions en les adaptant. On consultera l'aide en ligne plot3d,option. Signalons :

- **view=**zmin..zmax ou view=[xmin..xmax,ymin..ymax,zmin..zmax] qui indique les intervalles restreignant le tracé et met en évidence des particularités des surfaces non visibles compte tenu des l'échelles (utile avec display3d).
- **grid=**[n,m] précise les dimensions de la grille du plan horizontal, Maple calcule les hauteurs aux noeuds de la grille par défaut n=m=25.
- **coords=** permet aussi les réprésentations par rapport à un repère cylindrique ou sphérique.

Il existe comme pour les représentations dans le plan des commandes de la bibliothèque plots. Signalons :

- **impliciteplot3d** pour les surfaces définies de façon implicite dont la syntaxe est semblable à impliciteplot.
- **display3d** pour l'affichage de plusieurs surfaces enregistrer dans des variables informatiques

de type PLOT3D.

– **spacecurve** représente les courbes dans l'espace au lieu des surfaces.

– **animate3d** etc..

```
> plot3d((x,y)->x**(1/2)*y**(1/3),0..4,0..4);
```

```
> f:=(x,y)->piecewise(x>0 and x<1 and y>0 and y<1,1,0);
```

```
> plot3d(f,-2..2,-2..2,grid=[50,50],style=patch);
```

```
> plot3d(x^2-y^2,x=-1..1,y=-1..1);
```

```
> with(plots):spacecurve([t*cos(t),t*sin(t),t^2],t=0..6*Pi,
color=black);
```