Débuter avec pMaxima (Version 1.0)

Jean-Michel Sarlat
29 juin 2013

pmaxima *est un script* PERL *qui lit, en entrée, un fichier texte* `fichier.w`[1] *et qui restitue, en sortie, un autre fichier texte* `fichier.tex`. *Certaines lignes du premier ont été* interceptées *par le script et ont donné lieu à un traitement (généralement l'exécution des commandes* ***Maxima*** *embarquées) et le résultat est emballé dans le second. Il ne reste plus qu'à exploiter le fichier produit dans une compilation...*
Pour l'instant, ce script ne fonctionne que sous LINUX.

Table des matières

1 Installation

Voici plusieurs étapes à suivre pour vous assurer du fonctionnement du script.

1/ Téléchargez l'archive de la dernière version de ***pmaxima*** à la page :

1. L'extension peut-être quelconque, autre que `tex`, moi c'est `w`...

http://melusine.eu.org/syracuse/P/pmaxima/

2/ Décompressez l'archive et

- déplacez le fichier `maxima-init.lisp` dans le répertoire `.maxima` à la racine de votre HOME [2],
- déplacez le fichier `pmaxima.pl` dans votre répertoire `~/bin` ou dans tout autre répertoire accessible via la variable PATH de votre shell,
- renommez le fichier `pmaxima.pl` en `pmaxima` et rendez-le exécutable.

3/ Vérifiez que les applications `maxima` et `xmaxima` sont installées sur votre système. La deuxième n'est pas absolument nécessaire mais elle permet l'utilisation de la macro `plotdf`, très intéressante dans l'étude des systèmes différentiels.

4/ Vérifiez que le module PERL : `YAML::Tiny` est installé. On le trouve généralement dans les dépôts des distributions `Linux` ; l'installation peut être faite à partir de CPAN.

5/ Créez le répertoire `.wanda` à la racine de votre HOME, déplacez-vous dans ce répertoire puis :

- Exécutez la commande :
  ```
  $> pmaxima -c
  ```
  Le fichier `pmaxima.cfg` est écrit, il contient une configuration par défaut de ***pmaxima***. Vous pourrez l'adapter à votre usage, il sera automatiquement lu à chaque exécution du script.
- Ouvrez le fichier `pmaxima.cfg` avec votre éditeur favori et repérez la variable `MAXIMA`, vous allez devoir l'adapter à votre système.
  Localiser l'application `maxima` sur votre système [3], le chemin que vous trouverez dans la configuration par défaut correspond à celui d'une distribution DEBIAN, il ne doit pas y avoir beaucoup de différence avec une autre distribution, en dehors du numéro de version.

6/ Installez le script `pmp` qui utilise METAPOST pour la création de figures si vous souhaitez utiliser les macros de `gdd.mac`.

- Script `pmp` : http://melusine.eu.org/syracuse/P/pmp/.
- Macros de `gdd.mac` : http://melusine.eu.org/syracuse/P/pmaxima/gdd/.

2 Premier exemple

Considérons le fichier `un.w` dont le contenu est :

```
Voici un développement limité pour commencer...
. taylor(sin(tan(x)),x,0,5);
```

2. Il sera, peut-être, nécessaire de créer le répertoire `.maxima`. Il est possible aussi qu'un fichier `maxima-init.lisp` existe à cet endroit, auquel cas vous ajouterez le contenu du fichier de l'archive à celui qui est déjà présent.

3. Généralement ***Maxima*** est lancé via un script, ce n'est pas celui-ci qui nous intéresse mais le fichier binaire que lance ce script

Le point à noter est la ligne qui commence par... un point suivi d'un espace (au moins), elle contient une commande *Maxima* !

Exécutons *pmaxima* sur ce fichier.

```
$> pmaxima un.w
```

Nous obtenons alors le fichier `un.tex` :

```
Voici un développement limité pour commencer...

{\color{gray}
\begin{quote}
\begingroup\leavevmode
\llap{\footnotesize\textbf{\itshape m]\hspace{1em}}}%
\begin{minipage}[t]{\linewidth}\small
\begin{alltt}
taylor(sin(tan(x)),x,0,5);
\end{alltt}
\end{minipage}
\endgroup
\end{quote}
}

\begin{quote}
\begingroup\leavevmode
\llap{\footnotesize\color{orange}\%o1 : }
   {\small\color{teal}\(x+\frac{x^3}{6}-\frac{x^5}{40}+\cdots \)}
\endgroup
\end{quote}
```

Ce fichier n'est pas compilable directement mais il peut être inclu dans un autre fichier contenant ce qui est nécessaire pour une compilation avec le moteur TEX de votre choix.

Cela donne alors ceci :

Voici un développement limité pour commencer...

```
m]  taylor(sin(tan(x)),x,0,5);
```

%o1 : $x+\frac{x^3}{6}-\frac{x^5}{40}+\cdots$

Il y a plusieurs éléments de configuration utilisés ici :

EXTENSION — C'est l'extension du fichier de sortie.

```
[EXTENSION]
tex
```

COMMANDE — C'est l'emballage de la commande.

```
[COMMAND]
&nl{\color{gray}
\begin{quote}
\begingroup\leavevmode
\llap{\footnotesize\textbf{\itshape m]\hspace{1em}}}%
\begin{minipage}[t]{\linewidth}\small
\begin{alltt}
$0
\end{alltt}
\end{minipage}
\endgroup
\end{quote}
}&nl&nl
```

Le code ci-dessus est celui qui sera rendu par ***pmaxima*** pour inscrire la commande dans le fichier de sortie, `$0` étant remplacé par la commande elle-même. Les chaînes `&nl`, au début et à la fin, seront remplacées par des sauts de ligne.

En adaptant cet élément de configuration dans le fichier `~/.wanda/pmaxima.cfg` vous personnaliserez, à priori, le rendu des commandes dans vos documents produits à l'aide ***pmaxima***.

RESULTAT — C'est l'emballage du résultat.

```
[RESULTAT]
\begin{quote}
\begingroup\leavevmode
\llap{\footnotesize\color{orange}\%o$0 : }
        {\small\color{teal}$1}
\endgroup
\end{quote}
&nl&nl
```

Ici `$0` est remplacé par le numéro de la commande dans le fichier traité (vous pouvez, bien-sûr, ne pas le faire apparaître) et `$1` par le résultat qui aura été, au préalable, bordé par le contenu des éléments de configuration **BMATH** (`\(` par défaut) et **EMATH** (`\)` par défaut).

3 Deuxième exemple

Source de `deux.w` :

```
S!PRETEX!

Voici une macro permettant la mise en forme de la factorisation d'un
déterminant
. cdet([m]) := block(
    [mat:substpart(matrix,m,0)],
    'determinant(mat)=factor(determinant(mat))
);

Quelques applications:
. cdet([1,1,1],[a,b,c],[a^2,b^2,c^2]);
. cdet([1,x,x],[x,1,x],[x,x,1]);
. cdet([a,b,c],[b,c,a],[c,a,b]);
. cdet([(b-c)^2,b^2,c^2],[a^2,(c-a)^2,c^2],[a^2,b^2,(a-b)^2]);

S!POSTTEX!
```

Avec cet exemple, il y a deux choses à noter.

1/ Les commandes ***Maxima*** peuvent s'écrire sur plusieurs lignes, à la condition qu'il n'y ait pas de ligne vide intercalée. Le script ***pmaxima*** considère toute ligne vide suivant du code ***Maxima*** comme indication de la fin de l'inscription. Cela dit, toute nouvelle inscription ferme la précédente...

2/ Les chaînes de la forme `S!ELEMCONFIG!` sont remplacées par le texte de l'élément de configuration `ELEMCONFIG`.

Ici, si vous définissez deux nouveaux éléments de configuration : `PRETEX` et `POSTTEX` dans votre fichier de configuration `pmaxima.cfg`[4], leur contenu sera substitué aux chaînes les invoquant et vous pourrez alors disposer d'un fichier complet, compilable.

En adaptant l'élément de configuration `TEX` vous pourrez désigner le moteur TEX[5] que vous souhaitez utiliser pour la compilation.

Ainsi, il suffira d'éxécuter :

```
$> pmaxima -t deux.w
```

pour que le filtrage de votre fichier soit suivi d'une compilation. Vous pouvez ainsi utiliser ***pmaxima*** dans TEXWORKS...

Rendu de `deux.tex` :

Voici une macro permettant la mise en forme de la factorisation d'un déterminant

```
m] cdet([m]) := block(
       [mat:substpart(matrix,m,0)],
       'determinant(mat)=factor(determinant(mat))
   );
```

Quelques applications :

```
m] cdet([1,1,1],[a,b,c],[a^2,b^2,c^2]);
```

4. Elles n'y sont pas par défaut, vous pouvez les-y mettre et même sous un autre nom.
5. Et les options qui vous intéressent.

%o2 : $$\begin{vmatrix} 1 & 1 & 1 \\ a & b & c \\ a^2 & b^2 & c^2 \end{vmatrix} = (b-a)\,(c-a)\,(c-b)$$

```
m] cdet([1,x,x],[x,1,x],[x,x,1]);
```

%o3 : $$\begin{vmatrix} 1 & x & x \\ x & 1 & x \\ x & x & 1 \end{vmatrix} = (x-1)^2\,(2\,x+1)$$

```
m] cdet([a,b,c],[b,c,a],[c,a,b]);
```

%o4 : $$\begin{vmatrix} a & b & c \\ b & c & a \\ c & a & b \end{vmatrix} = -(c+b+a)\left(c^2-b\,c-a\,c+b^2-a\,b+a^2\right)$$

```
m] cdet([(b-c)^2,b^2,c^2],[a^2,(c-a)^2,c^2],[a^2,b^2,(a-b)^2]);
```

%o5 : $$\begin{vmatrix} (b-c)^2 & b^2 & c^2 \\ a^2 & (c-a)^2 & c^2 \\ a^2 & b^2 & (a-b)^2 \end{vmatrix} = -2\,a\,b\,c\,(c-b-a)\,(c-b+a)\,(c+b-a)$$

J'utilise ici la macro :
`\def\det{\renewenvironment{pmatrix}{\begin{vmatrix}}{\end{vmatrix}}}`
pour rendre le déterminant sous sa forme habituelle.

4 Comment procéder pour l'écriture des sources ?

Je n'ai pas de méthode à proposer, le plus souvent je suis dans l'un des trois cas suivants :

1/ Je suis suffisamment sûr des commandes à utiliser pour répondre à mes besoins et le document envisagé est peu volumineux. J'écris directement le source dans un éditeur de texte et j'exécute ***pmaxima*** avec l'option `-t` [6]. C'est un plaisir toujours renouvelé de découvrir, en ouvrant le **pdf** obtenu, l'ensemble du document.

2/ Je suis assez sûr des commandes à utiliser mais le document est copieux et je préfère suivre pas à pas sa construction. J'utilise TeXworks pour y aller progressivement.

3/ J'ai des doutes. Dans ce cas j'ouvre `xmaxima`, je tâtonne et quand je parviens à atteindre le résultat attendu, je sauvegarde ma session que je reprends ensuite avec un éditeur ; il me suffit alors de pointer les commandes et d'y insérer le texte nécessaire.

6. Éventuellement `-rt` pour la mise au point du texte autour des commandes et de leur résultat.

5 Réutilisation de résultats obtenus

Le dernier résultat obtenu peut-être réutilisé dans le texte du document avec la chaîne `.%.`, il convient alors de l'insérer dans un environnement adapté pour qu'il soit rendu convenablement.

```
m] 20!;
```

```
%o1 : 2432902008176640000
```

On a 20! = 2432902008176640000.

En terminant une commande par `-`, on évitera toute inscription directe dans le document, on dispose quand même de la possibilité de rappeler le résultat ($\pi \approx 3.141592653589793$).

Lorsqu'une commande se réduit à une affectation simple, l'évaluation rendue par ***Maxima*** est réutilisable dans le texte à l'aide d'une chaîne pointée.

```
m] V:4/3*%pi*r^3$
```

Le volume d'une sphère de rayon r est $V = \frac{4\pi r^3}{3}$.

Voici le codage de cette section :

```
Le dernier résultat obtenu peut-être réutilisé dans le texte du document
avec la chaîne \texttt{.\%.}, il convient alors de l'insérer dans un
environnement adapté pour qu'il soit rendu convenablement.
. 20!;
\noindent{}On a \(20! = .%. \).

En terminant une commande par \texttt{-}, on évitera toute inscription
directe dans le document, on dispose quand même de la possibilité de
rappeler le résultat
. ev(%pi,numer); -
% <--- Pour terminer la commande sans inscrire de saut de ligne dans
% le source
(\(\pi \approx{} .%. \)).

Lorsqu'une commande se réduit à une affectation simple, l'évaluation
rendue par \Maxima{} est réutilisable dans le texte à l'aide d'une
chaîne pointée.

. V:4/3*%pi*r^3$

Le volume d'une sphère de rayon \(r\) est \(V=.%V.\).
```

6 Reprise d'un document

Un document qui a été « *calculé* », autrement dit *filtré* par ***pmaxima*** peut être modifié sans que les calculs soient repris. Pour rester cohérent il suffit que les commandes (ou bloc de commandes) ***Maxima*** inscrites ne soient pas modifiées et que leur ordre reste inchangé. On peut ainsi, au cours d'une deuxième passe, éditer le texte séparant les commandes, réutiliser des résultats dans le texte du document, masquer des calculs, etc.

L'option à utiliser pour réutiliser les résultats d'une passe précédente est `-r`[7].

```
$> pmaxima -r calculs.w
```

pmaxima se servira des résultats stockés dans `calculs.yml` pour filtrer le fichier `calculs.w`.

7 Lecture des fichiers de configuration

Le script ***pmaxima*** contient une définition des éléments de configuration par défaut dans sa section `__DATA__`.

À l'exécution il cherche à lire les fichiers de configuration suivants, dans cet ordre :

- `~/.wanda/pmaxima.cfg` (répertoire à la racine du `HOME`),
- `../pmaxima.cfg` (répertoire immédiatement supérieur),
- `./pmaxima.cfg` (répertoire d'exécution du script).

Le principe est simple : dès qu'un élement est lu dans l'un de ces fichiers, son contenu vient se superposer à celui de l'élément existant ou l'élément est créé s'il n'existe pas. En pratique cela laisse beaucoup de possibilités pour des variations[8], décuplées par le fait d'utiliser des liens symboliques.

8 Graphisme

8.1 Commandes `plot2d` et `plot3d`

Les instructions `plot2d` et `plot3d` sont interprétées par ***pmaxima*** comme faisant appel au terminal `ps` de GNUPLOT (voir l'élément de configuration `PLOTGNUEPS`). Les figures produites[9] sont directement insérées dans le flot du document.

Source de `trois.w` :

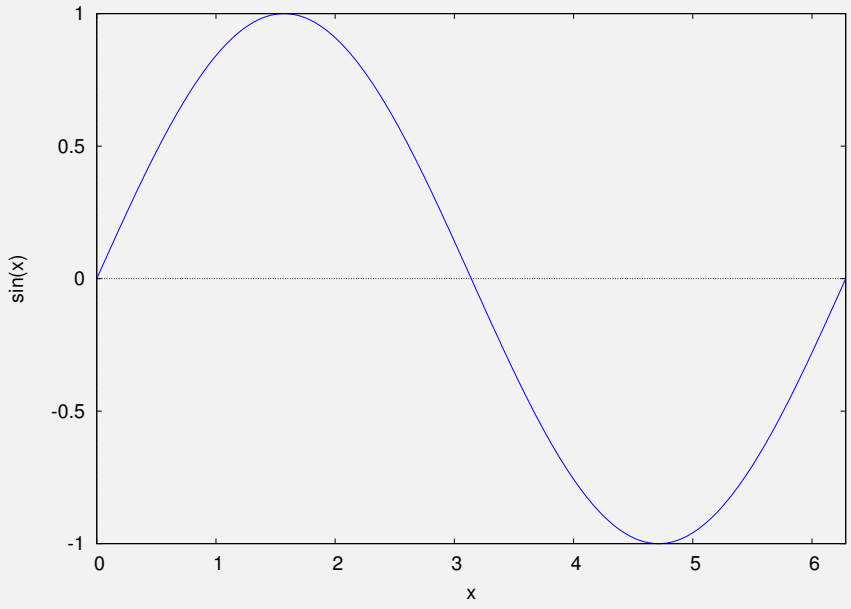
```
Une courbe...
. plot2d(sin(x),[x,0,2*%pi]);

Une surface...
. plot3d(x^2-y^2,[x,-2,2],[y,-2,2],[grid,12,12]);
```

7. Elle peut être utilisée en conjonction avec `-t` : `-rt`.
8. Je dois reconnaître que cela me suffit compte tenu de la hiérarchie que j'utilise pour mes documents.
9. Elles sont aux formats `EPS` et `PDF` dans le répertoire d'exécution ou celui spécifié par la variable de configuration `DIRFIG`.

Rendu de `trois.tex` :

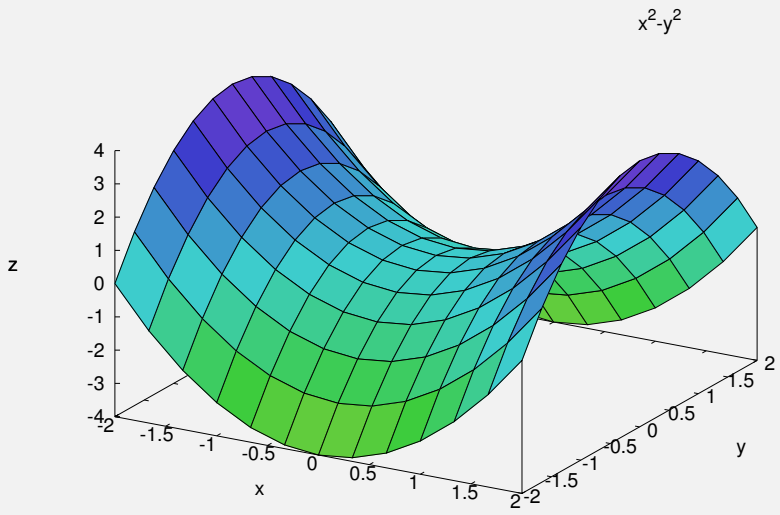
Une courbe...

```
m] plot2d(sin(x),[x,0,2*%pi]);
```



Une surface...

```
m] plot3d(x^2-y^2,[x,-2,2],[y,-2,2],[grid,12,12]);
```



8.2 Commande `plotdf`

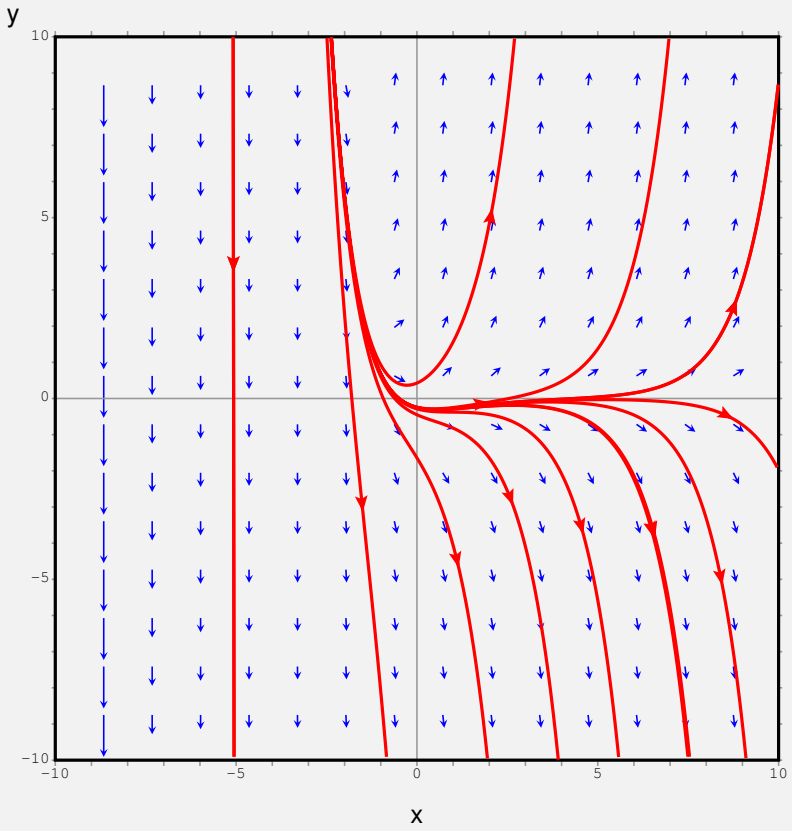
La commande `plotdf` de ***Maxima*** est très intéressante. Elle déclenche [10] l'apparition d'une fenêtre présentant le champ de vecteur associé à une équation différentielle. Il ne reste alors qu'à choisir un point à la souris pour définir des *conditions initiales*, la courbe intégrale associée est alors dessinée. Vous pouvez répéter l'opération jusqu'à clore l'interface. ***pmaxima*** capture l'image que vous pouvez sauvegarder [11] à partir de cette interface et l'insère dans le document.

Source de `quatre.w` :

```
. load("plotdf")$
. plotdf(x*exp(-x)+y,[trajectory_at,2,-0.1]);
```

Rendu de `quatre.tex` :

```
m] load("plotdf")$

m] plotdf(x*exp(-x)+y,[trajectory_at,2,-0.1]);
```



10. À la condition que `xmaxima` soit installé.
11. À la condition de ne pas changer le nom proposé : `maxplot.eps`.

8.3 Commandes `draw2d`, `draw3d`, `draw`

Le paquet `draw` qui accompagne la distribution de ***Maxima*** est une excellente interface à GNUPLOT. En choisissant l'un des terminaux `eps_color` ou `eps`, une figure sera produite (`maxima_out.ps`), ***pmaxima*** vous demandera éventuellement d'attendre la fin du processus de céation puis l'insérera dans le flot du document.

Important — Au moment du traitement, par ***Maxima***, de ces commandes, quelques fichiers pour GNUPLOT sont produits à la racine de votre `HOME`. ***pmaxima*** en fait une copie dans votre répertoire en adaptant les noms et les chemins inclus. Il y en a, principalement, deux, l'un avec l'extension `.gp` (commandes GNUPLOT) et l'autre avec l'extension `.dta` (données). Il est alors possible de les adapter, éventuellement reproduire les figures en utilisant le terminal `epslatex` de GNUPLOT...

Les exemples donnés ici proviennent du site

http://riotorto.users.sourceforge.net/index.html

maintenu par *Mario Rodríguez Riotorto* qui est l'auteur du paquet `draw`.

Source de `cinq.w` :

```
. load(draw)$

Commençons par une courbe et ses asymptotes.
. draw2d(
       terminal = eps_color,
           grid = true,
            key = "y = x^2/(x-2)",
         yrange = [-10,20],
         color  = red,
       explicit(x^2/(x-2),x,-9,15),
       /* asymptotes */
       key = "", line_type = dots, color = blue,
       explicit(x+2,x,-9,15),
       nticks = 70,
       parametric(2,t,t,-10,20),
       head_length = 0.3, color = black, line_type = solid,
       vector([5.35,2.45],[-1.53,3.25]),
       vector([-1,7.5],[3,0]),
       label_alignment = left,  label(["y = x+2",6,2.5]),
       label_alignment = right, label(["x = 2",-1.7,7.5])
)$

Représentation de surfaces.
. draw3d(
       surface_hide = true,
       terminal     = eps,
       palette      = gray,
       xu_grid      = 50,
       yv_grid      = 50,
```

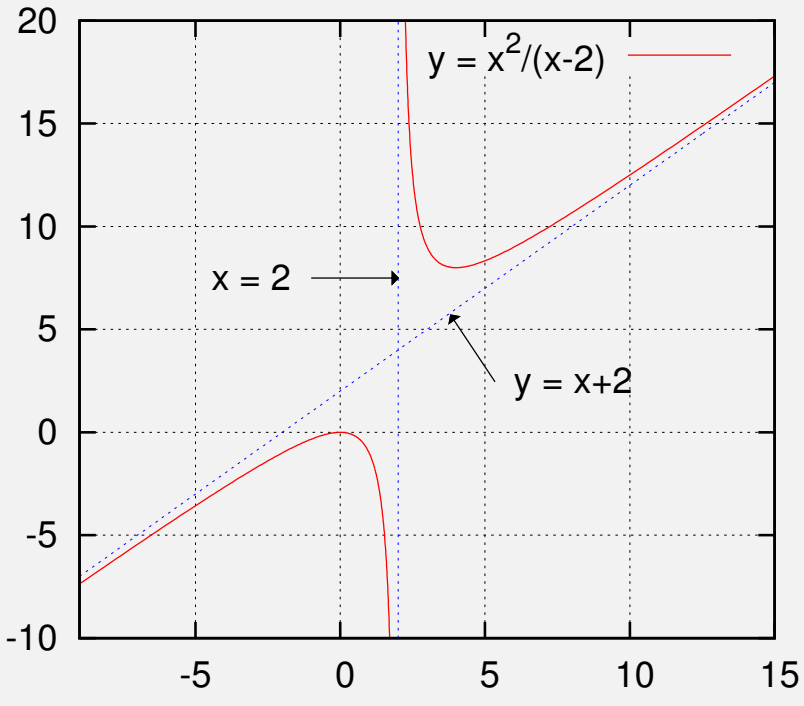
```
       enhanced3d   = true,
       explicit(5*sin(x*y), x, -5, 5, y, -5, 5),
       enhanced3d = 5 * sin(v*u),
       explicit(u^2+v^2+10, u, -5, 5, v, -5, 5)
)$
```

Rendu de `cinq.tex` :

```
m] load(draw)$
```

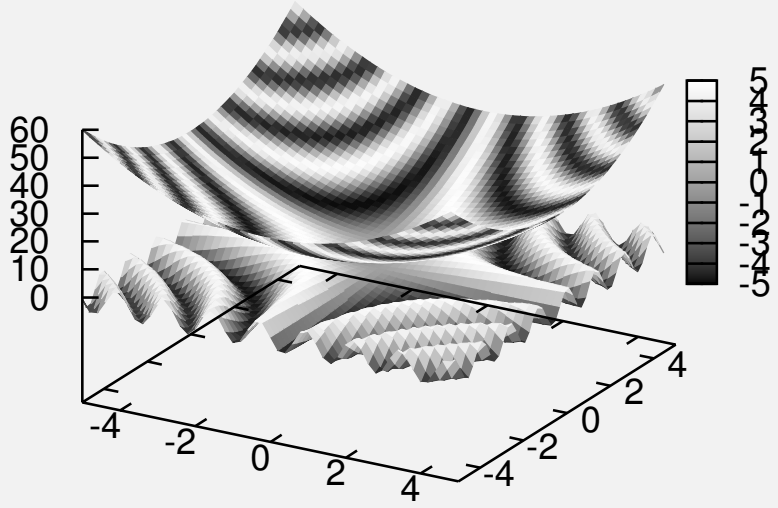
Commençons par une courbe et ses asymptotes.

```
m] draw2d(
           terminal = eps_color,
               grid = true,
                key = "y = x^2/(x-2)",
             yrange = [-10,20],
             color  = red,
           explicit(x^2/(x-2),x,-9,15),
           /* asymptotes */
           key = "", line_type = dots, color = blue,
           explicit(x+2,x,-9,15),
           nticks = 70,
           parametric(2,t,t,-10,20),
           head_length = 0.3, color = black, line_type = solid,
           vector([5.35,2.45],[-1.53,3.25]),
           vector([-1,7.5],[3,0]),
           label_alignment = left,  label(["y = x+2",6,2.5]),
           label_alignment = right, label(["x = 2",-1.7,7.5])
    )$
```



Représentation de surfaces.

```
m]  draw3d(
            surface_hide = true,
            terminal     = eps,
            palette      = gray,
            xu_grid      = 50,
            yv_grid      = 50,
            enhanced3d   = true,
            explicit(5*sin(x*y), x, -5, 5, y, -5, 5),
            enhanced3d = 5 * sin(v*u),
            explicit(u^2+v^2+10, u, -5, 5, v, -5, 5)
    )$
```



8.4 Macros de `gdd.mac`

`gdd.mac` est un ensemble de macros pour construire et étudier des figures de géométrie plane [12]. C'est un fichier que je développe et, comme ***pmaxima***, c'est inachevé, le but est d'obtenir des résultats, ici des figures, assez rapidement. Il reste beaucoup à faire...

http://melusine.eu.org/syracuse/P/pmaxima/gdd/

Source de `six.w` :

12. En attendant `gddd.mac`.

```
. load("gdd.mac")$
. (B:Origine, A:Point(3,6.5), C:Point(11.5,0))$
. T:Triangle(A,B,C)$
. (H:PointTriangle(T,"H"), G:PointTriangle(T,"G"), O:PointTriangle(T,"O"))$
. CC:Cercle(O,Distance(O,A))$
. ([A1,B1,C1]:PiedsHauteurs(T), Ha:Hauteurs(T))$
. Options(["B1,C","urt"],["H", "rt"],["A1","bot"])$
. Figure('A,'B,'C,'CC,'T,'A1,'B1,'C1,'H,'O,'G,'Ha);
```

Rendu de `six.tex` :

```
m] load("gdd.mac")$

m] (B:Origine, A:Point(3,6.5), C:Point(11.5,0))$

m] T:Triangle(A,B,C)$

m] (H:PointTriangle(T,"H"), G:PointTriangle(T,"G"), O:PointTriangle(T,"O"))$

m] CC:Cercle(O,Distance(O,A))$

m] ([A1,B1,C1]:PiedsHauteurs(T), Ha:Hauteurs(T))$

m] Options(["B1,C","urt"],["H", "rt"],["A1","bot"])$

m] Figure('A,'B,'C,'CC,'T,'A1,'B1,'C1,'H,'O,'G,'Ha);
```