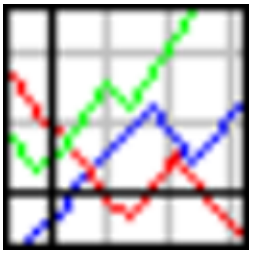
Maxima
& GnuPlot



Logiciel de calcul formel

& logiciel de tracé de courbes

Sommaire :

A. Présentation :

Maxima est un logiciel de calcul formel offrant de nombreuses possibilités : calcul sur les polynômes, calcul sur les matrices, dérivations, intégration... Son usage est plutôt destiné aux études supérieures.
Il est à noter qu'il existe d'autres logiciels de calcul formel plus facile à utiliser en classe : le logiciel de calcul formel intégré dans Geogebra, Xcas...
Ces logiciels sont appellés "*CAS*" : Computer Algebra System.

Maxima permet également de faire des calculs en virgules flottantes avec un grand nombre de chiffres dans la partie décimale *(la limitation viendra du temps de calcul accordé à Maxima)*.
En fait Maxima est une applcation en ligne de commande et on utilisera WxMaxima comme interface graphique au cours de cette formation.

Après une prise en main sommaire de ce logiciel, nous verrons au cours de ce tutorial comment utiliser Maxima pour faciliter la construction d'exercices pour le secondaire.
Par sa puissance de calcul, nous demanderons à Maxima de traiter un même type d'exercice mais avec des "*valeurs initiales différentes*" : la diversité des solutions *(entières, décimales, rationnelles...)* permettra à l'enseignant de choisir son exercice en fonction du niveau de complexité cherché.
Les valeurs initiales de l'exercice seront autant de variables didactiques à la disposition de l'enseignant.

1. La pile d'exécution :

Chaque requête passée à Maxima est numérotée à partir de 1 et reste accessible jusqu'à la fermeture de l'application : à tout moment, on peut récupérer un résultat antérieur.

Dans wxMaxima :

- Chaque ligne de saisie est précédée du symbole `%i` *(pour input)* suivi du numéro de l'instruction ;
- Les résultats retournés par Maxima sont affichés sur une ligne précédée du symbole `%o` *(pour output)* suivi du numéro de l'instruction associée.

Ainsi, lors de l'ouverture d'une session, la première commande à saisir est précédée de `(%i1)` et son résultat sera affiché sur une ligne commençant par `(%o1)`.

Lors de la saisie de la seconde opération dans WxMaxima, on intègre le résultat de la première opération à l'aide du symbole `%i1`.

2. La syntaxe :

La syntaxe utilisée par Maxima est très rigoureuse ; voici quelques règles à respecter lorsqu'on saisit des commandes :

- chaque commande doit se terminer par un point virgule `;` ou un dollar `$` :
  - Le point virgule "`;`" indique la fin d'une instruction et indique à Maxima d'afficher le résultat.
  - Le dollar "`$`" indique la fin d'une instruction mais son résultat ne sera pas affiché dans Maxima.
- Lors de l'écriture d'expressions algébriques, tous les signes doivent être écrits explicitement. L'écriture "`3x+2`" provoquera une erreur de syntaxe qu'on corrigera en : "`3*x+2`"
- Voici quelques unes des constantes prédéfinies dans Maxima :
  - `%e` : le nombre d'Euler $\exp(1)$;
  - `%pi` : le nombre π ;
  - `%phi` : le nombre d'or $\frac{1+\sqrt{5}}{2}$;
  - `%i` : l'unité des imaginaires.
- Maxima est un logiciel de calcul formel, mais si un nombre décimal est saisi lors d'une instruction, son résultat sera retourné sous forme décimale. En voici un exemple :
  - `2.5/3` $\rightsquigarrow$ 0,8333333
  - `(5/2)/3` $\rightsquigarrow \frac{5}{6}$
- Pour affecter une valeur à une variable, on utilise l'opérateur deux points "`:`".
  Voici deux exemples d'affectation :
  - `a:-1`
  - `m:(x+1)*(x^2-1)`
- Pour définir une fonction, on utilise l'opérateur "`:=`". Voici la définition de la fonction "*logarithme décimal*" qu'on nommera `log10` :
  - `log10(x):=log(x)/log(10)`
- On peut définir une fonction également à partir d'une expression existante. Pour cela, on utilise les deux opérateurs `'` et `''`.
  Considérons la variable `m` définie par : `m:x^2+3;`
  On définit une fonction à partir de l'expression de `m` à l'aide d'une des deux commandes suivantes : `f(x):=''m` `define('f(x),m)`

  Voici une description de ces deux opérateurs :
  - `'` : le simple apostrophe empêche l'évalution de l'expression lui succédant. Ainsi `'f(x)` ne sera pas évalué.
  - `''` : le double-simple apostrophe force l'évaluation de l'expression lui succédant. Ainsi,

`''m` renvoie l'expression de la variable `m`.

3. Quelques commandes :

1. *Calculs numérique :*

`sqrt(2)`	retourne la racine carrée de 2. Attention, Maxima utilise les puissances rationnelles pour simplifier les racines carrées : par exemple, `sqrt(8)` sera affichée en $2^{\frac{3}{2}}$
`3^2`	retourne le carré de 3
`6!`	retourne la factorielle de 6

`cos(%pi/4)`
`sin(%pi/4)`
`tan(%pi/4)`
	attention, les arguments des fonctions trigonométriques s'expriment en radians

`abs(-3)`	retourne la valeur absolue de -3
`divide(15,6)`	effectue la division euclidienne de 15 par 6. Cette commande retourne une liste de deux éléments : le quotient et le reste de la division euclidienne.
`mod(5,2)`	retourne le reste de la division euclidienne de 5 par 2
`ceiling(5.3)` $\rightsquigarrow 6$ `ceiling(-5.3)` $\rightsquigarrow -5$	retourne le plus petit des entiers supérieurs à l'argument
`floor(5.3)` $\rightsquigarrow 5$ `floor(-5.3)` $\rightsquigarrow -6$	retourne le plus grand des entiers inférieurs à l'argument
`truncate(-5.3)` $\rightsquigarrow -5$	de l'encadrement à l'unité près de l'argument, cette commande retourne l'entier le plus proche de 0
`round(5.6)` $\rightsquigarrow 6$ `round(-5.6)` $\rightsquigarrow -6$	de l'encadrement à l'unité près de l'argument, cette commande retourne l'entier le plus éloigné de 0
`float(sqrt(2))`	retourne une valeur approchée d'un nombre avec maximum 15 chiffres dans la partie décimale. On utilisera la fonction `bfloat()` pour passer cette limitation.
`bfloat()`	La commande `bfloat()` affiche la valeur décimale approchée du nombre passé en argument
`fpprec`	Le paramètre `fpprec` définit le nombre de chiffres utilisés dans la partie décimale par la commande `bfloat()`.

2. *Les nombres complexes :*

Pour définir un nombre complexe, on utilise l'unité `%i` des nombres imaginaires. Voici l'expression de deux nombres complexes dans Maxima : `3+4*%i` ; `exp(%i*%pi/6)`

`realpart(1+%i)`	retourne la partie réelle d'un nombre
`imagpart(1+%i)`	retourne la partie imaginaire d'un nombre ;

`cabs(1+%i)`	affiche le module d'un nombre complexe
`carg(1+%i)`	affiche l'argument d'un nombre complexe
`rectform(e^(%i*2*%pi/3))`	retourne la forme algébrique du nombre complexe
`polarform(1+%i)`	retourne la forme exponentielle du nombre complexe
`conjugate(1+%i)`	retourne le conjugé du nombre complexe

3. *Calculs formels :*

`expand((x+1)*(x^2-1))`	développe une expression
`factor(x^2+2*x+1)`	factorise une expression
`divide(x^3+x^2-1,x+1)`	effectue la division d'un polynome par un autre et retourne une liste de deux éléments : le quotient et le reste
`partfrac((x+1)/(x-1),x)`	affiche les éléments simples composant une fraction rationnelle
`num((x+1)/(x^2))`	affiche le numérateur d'un quotient
`denom((x+1)/(x^2))`	affiche le dénominateur d'un quotient
`rat(...)`	affiche un polynôme ou une fraction rationnelle sous forme développée réduite en fonction de l'indéterminé passée en second argument : `rat(expand((2*x+1)*(a*x^2+b*x+c)),x)` $\rightsquigarrow$ `2*a*x^3+(2*b+a)*x^2+(2*c+b)*x+c`
`ratcoef(2*x^3+2*x-1,x,3)`	retourne le coefficient du monôme en "x" de degré 3.
`diff(x^2+x-1,x)`	dérive une expression en fonction de la variable passée en second argument
`integrate(x+1,x)`	déterminer la primitive d'une expression en précisant la variable d'intégration. En fournissant, en arguments supplémentaires, les bornes d'intégration, Maxima retourne la valeur de l'intégrale : `integrate(x+1,x,1,6)`

`solve(x^2-4*x+1)`	détermine les racines de ce polynôme.
	Elle permet également de résoudre des équations simples :
	`solve(3*%e^x-1=3)`
	Elle permet également de résoudre un système d'équations linéaires :
	`solve([x+y-1=0,2*x-y+1=0],[x,y])`
	Voici deux valeurs particulières retournées par cette commande :
	➪ `solve(0*x=0)` : retourne `all`
	➪ `solve(0*x=1)` : retourne `[]`

`ev(x^2+1,x=2)`	évalue l'expression pour `x=2`. Cette commande peut également être utilisée pour effectuer une substitution dans une expression : `ev(5*x^2-2*x+1,x=y+1)` $\rightsquigarrow$ $5(y+1)^2-2(y+1)+1$
`taylor(log(x),x,1,4)`	développement de Taylor pour x_0=1 à l'ordre 4.

4. *Commande pour la programmation :*

Dans ce paragraphe, sont présentées des commandes utiles pour des séquences de programmation :

`is(3>%pi)`	retourne `false`. La commande `is()` permet d'effectuer un test
`not(is(3>%pi))`	retourne `true` ; le résultat est l'opposé du test `is(3>%pi)`
`freeof(x^2+y*x,z)`	vérifie si la variable du deuxième argument est présent dans l'expression du premier argument. Cet exemple retourne `true`
`numberp(m)`	retourne `true` si `m` est un nombre numérique
`emptyp(m)`	retourne `true` si `m` est une liste vide
`display(x,y)`	cette commande permet de forcer l'affichage de résultats *(ici, les valeurs de* `x` *et* `y`*)*, notamment lors de l'exécution de boucles
`printf()`	Considérons la commande : `x:2.2; printf(false,"~d ~a ~h",x,x/3,x/3);` cette commande permet d'écrire les résultats de Maxima dans des fichiers, mais ici avec l'argument `false`, les résultats s'afficheront à l'écran. L'appel à cette commande affichera la valeur de `x` sous la forme d'un entier *(*`~d`*)* et deux fois la valeur de `x/3` respectivement avec la forme d'affichage par défaut de Maxima *(*`~a`*)*, puis sous la forme d'un grand nombre en virgule flottante *(*`~h`*)*
`kill(x)`	permet d'effacer l'affectation de la variable `x` dans la session courante de Maxima. La commande `kill(all)` permet d'effacer toutes les affectations de la session courante.

5. *Les listes :*

Maxima est construit à partir du langage de programmation LISP ; ainsi, tout objet, ou presque, est en fait définit par une liste : l'expression `x+4` est codée en mémoire sous la forme : `(+ x 4)`.

`length([a,b,c])`	retourne 3, le nombre d'éléments de la liste

`part([a,b,c],1)`	retourne le premier élément `a` de la liste fournie en premier argument. Voici d'autres exemples d'utilisation de cette commande : • `part([x=-2,x=5],1)` : retourne `x=-2` ; • `part([x=-2,x=5],1,2)` : retourne `-2` ; Les deux exemples précédentes se comprennent lorsqu'on observe l'implémentation en LISP de la liste `[x=-2,x=5]` qui est : `[ (= x -2) (= x 5) ]` L'expression `z+2*y+x` se code en mémoire dans Maxima sous la forme `(+ z (* 2 y) x)`. Ainsi, on a les exécutions suivantes dans Maxima : • `part(z+2*y+x,1)` : retourne `z` ; • `part(z+2*y+x,2)` : retourne `2*y` ; • `part(z+2*y+x,2,2)` : retourne `y`.

B. Première prise en main :

1. Du calcul numérique :

Voici quelques exemples d'utilisation des nombres avec Maxima.

```
(%i1) 2+3
(%o1) 5

(%i2) 2/3+1; 2.0/3+1;
(%o2) 5/3
(%o3) 1.666666666666667

(%i4) float(%pi); bfloat(%pi); fpprec:30; bfloat(%pi);
(%o4) 3.141592653589793
(%o5) 3.141592653589793b0
(%o6) 30
(%o7) 3.14159265358979323846264338328b0

(%i8) m:x^2+2; ev(m,x=2); m;
(%o8) x^2+2
(%o9) 6
(%o10) x^2+2
```

- `(%i1)` montre un exemple simple d'entrée et de sortie avec Maxima.

Dans les prochaines commandes, l'utilisation du point virgule " `;` " permet de saisir plusieurs commandes sur une même ligne.

- `(%i2)` présente un calcul fractionnaire de Maxima. Lors de l'exécution de la commande `(%i3)`, Maxima rencontre le nombre décimal `2.0` et effectue alors ces calculs en mode décimal.
- `(%i4)` la commande `float()` affiche la valeur décimale du nombre π.
  `(%i5)` affiche la valeur décimale du nombre π avec la fonction `bfloat()` ; le suffixe `b0` indique un exposant nul.
  Après la modification du paramètre `ffprec`, l'instruction `(%i7)` retourne la valeur arrondie de `%pi` avec 30 chiffres dans la partie décimale.
- L'instruction `(%i9)` évalue l'expression `m` pour la valeur `x=2`

Exercice 1

1. En utilisant l'exemple `(%i4)`, donner la $35^{\text{ième}}$ décimale du nombre $\sqrt{2}$:

.....

2. En suivant l'exemple `(%i8)`, donner l'image de 1,35 par la fonction f définie par :

$$f(x) = \frac{x^2 - 1}{x - 2}$$

2. Des nombres complexes :

Voici quelques exemples d'utilisation des nombres complexes :

```
(%i1) a:1+%i; cabs(a); carg(a);
(%o1) %i+1
(%o2) √2
(%o3) π/4

(%i4) a:%e^(%i*%pi/3); realpart(a); imagpart(a); rectform(a);
(%o4) %e^(%i π)
(%o5) 1/2
(%o6) √3/2
```

```
(%o7)  √3%i/2 + 1/2

(%i8)  a:1+%i; conjugate(a); polarform(a);
(%o8)  %i+1
(%o9)  1-%i
(%o10)  √2 %e^(%i π/4)

(%i11)  f(z):=(1+%i)*z+conjugate(z); f(1); expand(f(%));
(%o11)  f(z):=(1+%i)*z+conjugate(z)
(%o12)  %i+2
(%o13)  2%i+3
```

Peu de choses à commenter dans ces exemples. L'instruction `(%i11)` & Co :

- `f(z):=(1+%i)*z+conjugate(z)` définit une fonction `f` à variable complexe ;
- `f(1)` retourne l'image de 1 par la fonction `f` ;
- `f(%)` calcule l'image, par la fonction `f`, du résultat retourné par la dernière commande : Maxima retourne la valeur de `f[f(1)]`

Exercice 2

1. Déterminer le module et l'argument du complexe $\sqrt{3+i}$.
2. Déterminer la valeur approchée du module et de l'argument du complexe $1+2\cdot i$.
3. Déterminer les cinq premiers termes de la suite (z_n) définie par : $z_{n+1} = \dfrac{1+i}{2}\cdot z_n$

3. Du calcul algébrique :

Voici quelques exemples d'utilisation des expressions algébriques avec Maxima :

```
(%i1)  m:2*x^2+6*x+4; factor(m); solve(m);
(%o1)  2x^2+6x+4
(%o2)  2(x+1)(x+2)
(%o3)  [x=-2,x=-1]

(%i4)  m:2*x^2; diff(m,x); integrate(m,x); integrate(m,x,1,3);
```

```
(%o4)  2x^2
(%o5)  4x
(%o6)  2x^3/3
(%o7)  52/3

(%i8)  a:2$ m:x^3-x$ n:diff(m,x)$
       expand(ev(n,x=a)*(x-a)+ev(m,x=a));
(%o11)  11x-16

(%i12)  kill(all)$ rat((x-2)*(a*x^2+b*x+c),x);
(%o13)  ax^3+(b-2a)x^2+(c-2b)x-2c
```

Quelques commentaires sur l'exécution des lignes 12 et 16 :

- L'instruction `(%i11)` retourne l'équation réduite de la tangente à la courbe de la fonction f définie par $x \longmapsto x^3-x$ au point d'abscisse 2 :
  - `ev(m,x=a)` est la valeur de $f(2)$;
  - `ev(n,x=a)` est la valeur de $f'(2)$.

  La commande `expand` permet d'obtenir la forme réduite de cette tangente.

  Dans l'exécution de cette ligne, on remarquera la présence la présence du signe dollar "`$`" indiquant à Maxima de ne pas afficher le résultat d'une instruction : les sorties des instructions `(%o8)`, `(%o9)`, `(%o10)` ne seront pas affichées.
- A la ligne 16 :
  - La commande `kill(all)` efface toutes les variables de la session courante de Maxima : on efface ainsi la valeur de la variable `a`.
  - La commande `rat()` permet d'exprimer `(x-2)*(a*x^2+b*x+c)` sous la forme d'un polynome en `x`.

Exercice 3

1. a. Donner les expressions simplifiées de la dérivée des fonctions suivantes :

$$f(x) = \frac{x^2-3x+4}{x+1} \quad ; \quad g(x) = (x+\sqrt{x})(x+1)$$

   b. Dans l'expression de la fonction dérivée g', donner la signification du terme $x^{3/2}$

2. On considère la fonction f définie par : $f(x)=6x^3-7x^2+1$

a. Vérifier, à l'aide d'un calcul mental, que 1 est une racine de ce polynôme.

b. Déterminer le polynôme P vérifiant l'égalité :

$$6x^3 - 7x^2 + 1 = (x-1)\cdot P$$

3. En s'inspirant de l'instruction `(%i12)` établir que l'égalité :

$$5x^3 + 2x^2 + x + 4 = (x+1)(a\cdot x^2 + b\cdot x + c)$$

entraine le système d'équations ci-contre :

$$\begin{cases} a = 5 \\ a + b = 2 \\ b + c = 1 \\ c = 4 \end{cases}$$

4. Des listes :

Voici quelques exemples d'utilisation des listes d'objets :

```
(%i1) a:14/5; m:(x-1)*(x-4)+(y-1)*(y-4)=0;
      n:solve(ev(m,x=a)); b:part(n,1,2); c:part(n,2,2);
      [a,b]; [a,c];
(%o1) 14/5
(%o2) (y-4)(y-1)+(x-4)(x-1)=0
(%o3) [y=2/5,y=23/5]
(%o4) 2/5
(%o5) 23/5
(%o6) [14/5,2/5]
(%o7) [14/5,23/5]
```

Le but de ces instructions est d'obtenir les coordonnées des points d'intersections qui s'afficheront en `%o6` et `%o7` :

- de la droite d'équation $x = \frac{14}{5}$
- et du cercle admettant pour diamètre le segment dont les extrémités ont pour coordonnées $(1\,;1)$ et $(4\,;4)$.

Cet exemple illustre la nécessité de l'usage des listes : la commande `solve()` retourne deux valeurs dans `(%o3)`v sous forme de liste. Nous avons besoin d'extraire ces deux valeurs pour obtenir les coordonnées `(%o6)` et `(%o7)` de ces deux points. Pour cela, nous allons extraire les valeurs de la liste contenue dans la variable `n`.

En Lisp, la variable `n` est codée de la manière suivante :

```
[ [ = y [ / 2 5 ] ] [ = y [ / 14 5 ] ] ]
```

Voici l'explication de l'affichage de `(%o5)` :

- `part(n,2)` : extrait le second élément de la liste `n` : `[ = y [ / 14 5 ] ]`
- `part(n,2,2)` : prendra le second élément de la sous-liste précédente : `[ / 14 5 ]`

Exercice 4

1. Saisissez le code suivant :

```
1 a:1; b:2; m:a*x+b;
2 n:x^3-2*x^3;
```

2. Exprimer dans Maxima les coordonnées d'au moins un point d'intersection de la fonction affine définie par l'expression `m` et la fonction polynomiale de degré 3 définie par l'expression `n` :

.....

.....

.....

.....

.....

On ne peut pas prédire par avance le nombre de points d'intersection de deux courbes quelconques.
Pour afficher toutes les coordonnées des points de deux courbes, Maxima devra s'adapter au résultat de la commande `solve()`.
Pour cela, nous aurons besoin de notions de programmation et de la commande `length()` retournant la taille d'une liste.

C. Programmation avec Maxima :

1. La boucle `for` *:*

Cette commande permet de répéter plusieurs fois les mêmes actions en faisant varier la valeur d'un paramètre.

Exercice 5

1. a. Saisissez le code suivant dans Maxima :

```
1 m:x^2+x+1;
2 for a:-2 thru 2 step 1 do (
3   ev(m,x=a)
4 );
```

b. Quel est l'affichage obtenu par l'exécution de la boucle ?

.....

.....

Dans l'affichage retourné, apparait `done` : cela signifie que l'instruction, s'étalant sur les lignes 2 et 3, s'est bien déroulée mais elle n'a retourné aucun résultat à afficher.

Lors de l'exécution de la boucle `for`, les instructions contenues dans les parenthèses suivant `do` sont exécutés plusieurs fois mais n'engendreront pas d'affichage : comme si chacune d'elles se terminée avec le caractère `$`.
Nous utiliserons plusieurs commandes dont `disp()`, `display()`, `print()` ou `printf()` pour forcer leur affichage.

2. a. Dans le code, modifier la ligne 3 en : `(y:ev(m,x=a),display([a,y]));`

b. Combien de résultats sont alors affichés par ce code ? Justifier.

3. a. Dans la seconde ligne du code, changer la commande "`step 1`" en "`step (1/2)`".

b. Après évaluation de votre code dans Maxima, quel est le changement sur l'affichage de sortie ?

La commande `for` admet la syntaxe suivante :

```
for a:xxx thru yyy step zzz do
```

où :

- `a` est la variable utilisée par la boucle et s'initialisera avec la valeur `xxx`
- La boucle s'exécutera tant que la valeur de `a` ne dépassera pas la valeur `yyy` ;
- A chaque fin d'exécution de la boucle, la variable `a` est augmentée de la valeur `zzzz`.

Exercice 6

On peut également imbriquer les boucles afin de faire varier plusieurs paramètres :

1. a. On considère le code ci-dessous :

```
for a:-3 thru 3 step 1 do
for b:-3 thru 3 step 1 do
for c:-3 thru 3 step 1 do (
  display([a,b,c])
)
```

Avant d'exécuter le code, pouvez vous prévoir le nombre de triplets qui seront affichés par Maxima.

.....

b. Saisissez ce code.

2. Modifiez le code ci-dessus afin d'afficher les racines de tous les polynômes $a{\cdot}x^2{+}b{\cdot}x{+}c$ où a, b et c sont des entiers appartenant à l'intervalle $[\![-3\,;3]\!]$.

- Un message d'erreur peut apparaître lors de l'exécution de ce code : un espace doit obligatoirement suivre l'instruction `do`.
  Dans Maxima, le retour à la ligne n'est pas considéré comme un séparateur de commandes.
- Pour répondre à la question 2., il suffisait de changer la liste `[a,b,c]` en `[a,b,c,solve(a*x^2+b*x+c)]`

2. *Travaux pratiques :*

Exercice 7

On considère la fonction complexe z définie sur $\mathbb{C}$ par la relation :

$f(z) = (1-2i){\cdot}z - i$

Déterminer avec Maxima, les dix premiers termes de la suite complexe (u_n) définie par :

$u_{n+1} = f(u_n) \quad ; \quad u_0 = 1+i$

Pensez à définir la fonction f dans Maxima à l'aide de la syntaxe suivante :

```
f(x):=(1-2*%i)*x-%i
```

Exercice 8

Considérons la fonction f définie par : $f(x)=x^3-2x^2+1$
et $\mathscr{C}_f$ sa courbe représentative dans un repère orthonormé.

Ecrire un programme avec Maxima affichant les équations réduites de toutes les tangentes à la courbe $\mathscr{C}_f$ aux points d'abscisse a où a parcours l'ensemble des entiers appartenant à l'intervalle $[\![-3\,;5]\!]$.

2. La structure conditionnelle `if` :

1. Travaux dirigés :

Exercice 9

1. Saisissez le code suivant :

```
for a:0 thru 100 step 1 do (
  if (a>50) and (a<70) then (
    display(a)
  )
)
```

2. Après exécution de ce script, justifier l'affichage des nombres par Maxima :

.....

.....

3. Quel test faut-il mettre dans ce programme afin afin que les entiers affichés soient seulement ceux de l'intervalle $[0\,;49]\cup[71\,;100]$?

.....

.....

4. Changez le test de la structure conditionnelle de ce programme en :

```
(a-floor(a/2)*2#0)
```

Comment caractériser les résultats affichées par la boucle ? Justifier ce comportement :

.....

.....

Une structure conditionnelle permet de contrôler l'exécution d'une partie du programme ou d'une autre en fonction de l'évaluation d'un test.

L'écriture d'une structure conditionnelle est de la forme suivante :

```
if (test) then (
   ...Evaluer si le test est vrai
)
else (
   ...Evaluer si le test est vrai
)
```

La clause `else` peut être omise ; la structure conditionnelle aura la forme :

```
if (test) then (
   ...Evaluer si le test est vrai
)
```

Le signe `#` permet de tester si deux valeurs sont différentes.

Voici quelques exemples d'utilsiation de la commande `is(...)` ; cette commande permet de connaître la valeur d'un test. Voici quelques exemples d'utilisation et la valeur du boléen retourné :

`is(1>2)` ⇝ `false` ; `is(1<2)` ⇝ `true` ; `is(1<1)` ⇝ `false`

`is(1<=1)` ⇝ `true` ; `is(1#1)` ⇝ `false` ; `is(1#2)` ⇝ `true`

La commande `floor` a été commenté à la page **??** de ce manuel.

Dans la question 4., l'instruction "`floor(a/2)*2`" retourne :

- `a` si `a` est pair ;
- `a-1` si `a` est impair.

Ainsi, l'instruction "`a-floor(a/2)*2`" retourne le reste de la division euclidienne de a par 2 : c'est-à-dire 0 ou 1.

Exercice 10

On considère le polynôme x^2+x+1 admettant deux racines complexes. On suppose que la variable `m` est définie par "`m:solve(x^2+x+1)`".

1. A l'aide de la commande `part` *(page 9)* et `imagpart` *(page 6)*, écrire un test renvoyant `true` si la première racine de ce polynôme est réelle et `false` sinon.

.....

.....

2. Pour afficher les racines de tous les polynômes de la forme $a{\cdot}x^2 + b{\cdot}x + c$ où les coefficients des monômes sont à valeurs entières dans l'intervalle $[\![-2\,;2]\!]$.

```
for a:-2 thru 2 step 1 do
for b:-2 thru 2 step 1 do
for c:-2 thru 2 step 1 do (
  m:solve(a*x^2+b*x+c),
  display([a,b,c,m])
);
```

a. Saisissez ce code dans Maxima.

b. Ajoutez une structure conditionnelle afin que ce script n'affiche que les polynômes ayant des racines réelles.

3. Considérons le test : `is(ceiling(n*100)=n*100)`.
Tester la valeur de ce test lorsque la variable `n` prend différentes valeurs décimales.
Que pouvez-vous dire des nombres décimaux retournant la valeur `true` pour ce test ?

.....

.....

4. a. Modifiez le script de la question 2. afin que le pas d'incrémentation des variables `a`, `b`, `c` soit de $\frac{1}{5}$.

b. Modifier la clause conditionnelle, ajouté dans la question 2. b. , en :

```
if (imagpart(part(m,1,2))=0) and (...) then
```

où les pointillés représentent un test permettant de sélectionner seulement les polynômes dont la première racine a un valeur décimale n'excédant pas deux chiffres dans la partie décimale.

Ainsi, le script demandé au cours de cet exercice pourrait ressembler au code suivant :

```
for a:-2 thru 2 step (1/5) do
for b:-2 thru 2 step (1/5) do
for c:-2 thru 2 step (1/5) do (
  m:solve(a*x^2+b*x+c),
  n:part(m,1,2),
  if (imagpart(n)=0) and (ceiling(n*100)=n*100) then
    display([a,b,c,n])
);
```

2. *Travaux pratiques :*

Exercice 11

Ecrire un programme donnant tous les polynômes de degré trois possédant trois racines réelles et dont leurs valeurs décimales n'excèdent pas deux chiffres dans la partie décimale.

3. Eviter certaines conditions :

1. *Travaux dirigés :*

Exercice 12

1. a. Saisissez le code suivant dans Maxima :

```
for a from -2 thru 2 step 1 do
for b from -2 thru 2 step 1 do
for c from -2 thru 2 step 1 do
for d from -2 thru 2 step 1 do (
  e:1,
  m:(2*x-1)/(x+1)-(a*x+b)/(c*x+d),
  display(solve(m))
);
```

b. Maxima arrête l'exécution de ce code à cause d'un erreur. Pouvez-vous interpréter cette erreur ?

.....

.....

- Maxima arrête l'exécution du script car il tombe sur l'erreur :
  "`expt: undefined: 0 to a negative exponent.`"
  lorsque le dénominateur `c*x+d` est nul : c'est-à-dire lorsque les variables `c` et `d` prennent la valeur 0 simultanément.
- En programmation pour débugger un code, on parsème le code d'indicateurs afin de connaître les valeurs avec lesquelles le programme produit une erreur.
  Remplacez l'instruction "`e:1`" par "`display([c,d])`".

2. a. Modifier le bloc d'instructions exécuté par les boucles `for` par le code suivant :

```
block(
  if (c=0) and (d=0) then go(fin),
  m:(2*x-1)/(x+1)-(a*x+b)/(c*x+d),
  display(solve(m)),
  fin
);
```

b. Vérifier maintenant que le script s'exécute entièrement.

- La commande `go()` oblige Maxima à déplacer son exécution d'un endroit à l'autre du code. Ainsi, lorsqu'on remarque que les valeurs de `c` et `d` s'annule, on déplace l'éxécution à la fin de la boucle grâce à l'instruction `go(fin)` qui déplace l'exécution vers la "*marque*" `fin`.
  Les instructions entraînant un dénominateur nul ne seront pas exécutées.
- Pour cela, nous devons être dans une structure `block(...)` et nous mettons un marqueur à la fin du bloc : ici, on l'appelle `fin`.
  Lorsque les conditions à éviter sont détectées, on déplace l'éxécution avec la commande `go(fin)`.

2. *Travaux pratiques :*

Exercice 13

Déterminer l'ensemble des fonctions admettant l'expression $f(x)=\dfrac{1}{a\cdot x^2+b\cdot x+c}$ et dont la représentation passe par le point de coordonnées $(1\,;1)$ où les paramètres `a`, `b` et `c` prenant des valeurs entières comprises dans l'intervalle $[\![-5\,;5]\!]$.

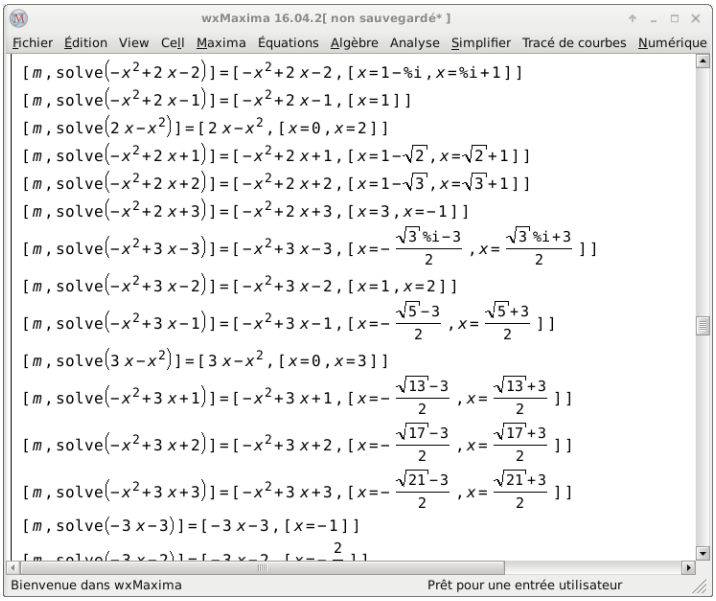
D. Exemples de codes :

La puissance de calcul *(et non pas l'intelligence de calcul)* de WxMaxima et le fait qu'il opère sur du calcul formel, nous permet de le programmer afin de traiter plusieurs fois un même exercice mais avec des conditions initiales différentes.
En observant la multitude de résultats fournis par WxMaxima, nous pouvons choisir un exercice répondant à un besoin pédagogique précis.

Par exemple le code suivant :

```
for a:-3 thru 3 do
for b:-3 thru 3 do
for c:-3 thru 3 do (
  m:a*x^2+b*x+c,
  display([m,solve(m)])
);
```

L'exécution de ce code affiche les racines de tous les polynômes de degré 2 dont les coefficients sont des entiers parcourant les valeurs de -3 à 3. Voici une partie de l'affichage de Maxima et quelques idées à extraire de ces résultats :



- $-x^2+2x-1$ est factorisable par une identité remarquable.
- Lors de la recherche des racines de $-x^2+2x+1$, l'élève devra effectuer une simplification du quotient pour obtenir la forme simplifiée.
- La forme factorisée de $-x^2+2x+3$ admet une expression factorisée assez simple.
- Le polynôme $-x^2+2x-3$ est un exemple classique de recherche des deux racines complexes pour un élève de terminale.
- Le polynôme $-x^2+3x-1$ est un exemple classique de recherche des deux racines réelles pour un élève de première.

1. Théorème de Pythagore - Quatrième :

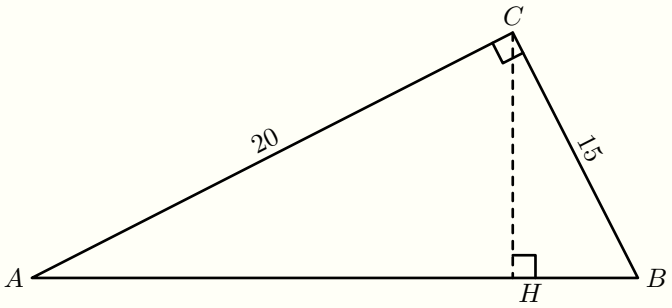
Voici trois situations extraite des données de WxMaxima

Exercice 1

On considère le triangle ABC rectangle en C dont on connait les mesures :

$AC = 20 \quad ; \quad BC = 15$

On note H le pied de la hauteur issue du sommet C dans le triangle ABC. Déterminer la mesure de la longueur du segment $[AH]$.

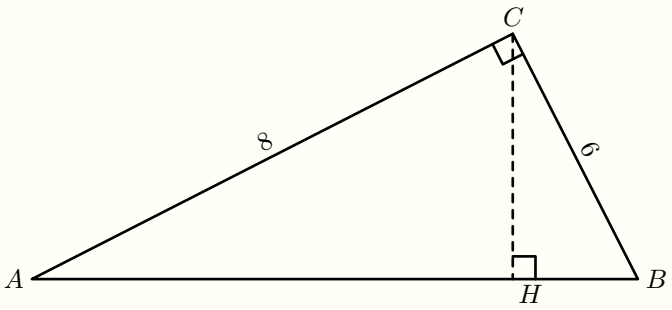


Exercice 2

On considère le triangle ABC rectangle en C dont on connait les mesures :

$AC = 8 \quad ; \quad BC = 6$

On note H le pied de la hauteur issue du sommet C dans le triangle ABC. Déterminer la mesure de la longueur du segment $[AH]$.

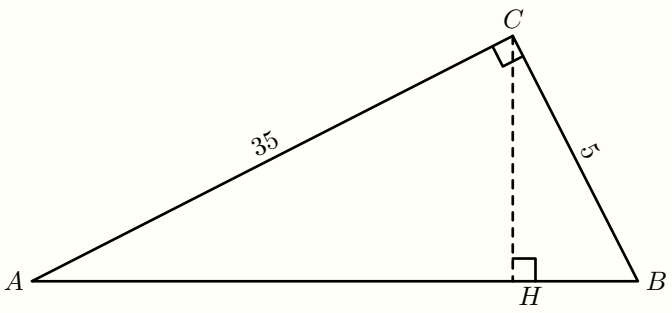


Exercice 3

On considère le triangle ABC rectangle en C dont on connait les mesures :

$AC = 35 \quad ; \quad BC = 5$

On note H le pied de la hauteur issue du sommet C dans le triangle ABC. Déterminer la mesure de la longueur du segment $[AH]$.



Le choix des valeurs initiales dans ces trois exercices permet de modifier la difficulté de chacun d'eux :

- Exercice 1 : lors de son traitement, toutes les valeurs trouvées seront entières.
  $AC = 20 \quad ; \quad BC = 15 \quad ; \quad AB = 25 \quad ; \quad CH = 12 \quad ; \quad AH = 16$
- Exercice 2 : des quotients apparaîtront
  $AC = 8 \quad ; \quad BC = 6 \quad ; \quad AB = 10 \quad ; \quad CH = \dfrac{24}{5} \quad ; \quad AH = \dfrac{32}{5}$
- Exercice 3 : le traitement des radicaux sera nécessaire pour résoudre cet exercice.
  $AC = 35 \quad ; \quad BC = 5 \quad ; \quad AB = 25\cdot\sqrt{2} \quad ; \quad CH = \dfrac{7}{2}\cdot\sqrt{2} \quad ; \quad AH = \dfrac{49}{2}\cdot\sqrt{2}$

Pour construire l'algorithme permettant d'afficher les différents traitement de cet exercice, nous commençons par identifier les différentes variables mises en jeux dans l'algorithme *(voir figure ci-contre)*

Voici le code qui a produit les conditions initiales des deux premiers exercices

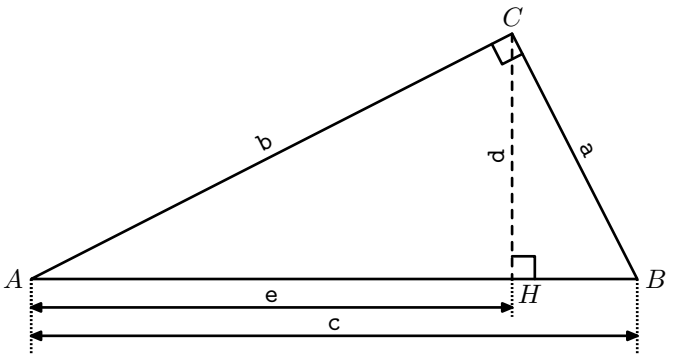
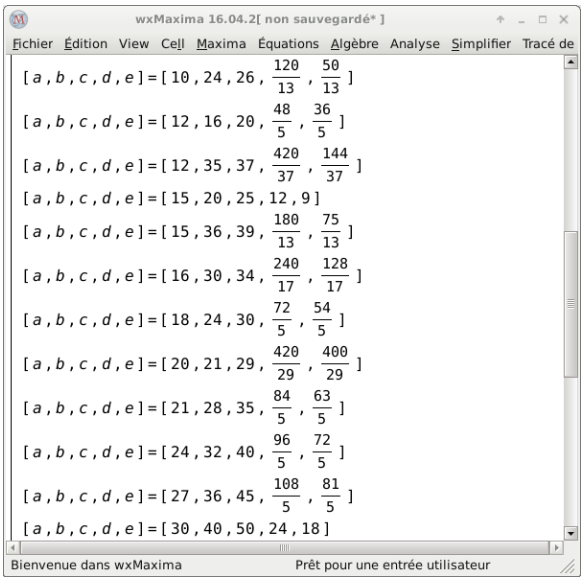


Fig. 3

```
1  mmax:40;
2  for a:1 thru mmax step 1 do
3  for b:a+1 thru mmax step 1 do block(
4    c:sqrt(a^2+b^2),
5    if(ceiling(c)#c) then go(fin),
6    d:a*b/(c),
7    e:sqrt(b^2-d^2),
8    display([a,b,c,d,e]),
9    fin
10 );
```

Ci-contre un extrait de l'affichage proposé par WxMaxima, on y repère des conditions initiales *(*`a`*,* `b`*)* avec des résultats à valeurs entières ou rationnelles.



La ligne `if(ceiling(c)#c) then go(fin),` permet de ne pas traiter les cas où la valeur de `c` n'est pas entière. Pour obtenir des couples de valeurs initiales entraînant l'apparition du radical de 2, il fallait remplacer la ligne 5 par :

```
if(ceiling(c*sqrt(2))#c*sqrt(2)) then go(fin)
```

2. Théorème de Pythagore - Quatrième :

Dans cette partie, nous allons demander à WxMaxima de construire deux types d'exercices basés sur la même configuration :

- Sur le premier exercice, les élèves doivent conclurent que le triangle ABC est rectangle en ne travaillant qu'avec des entiers *(facilitant le traitement d'un tel exercice pour un élève de*

quatrième)

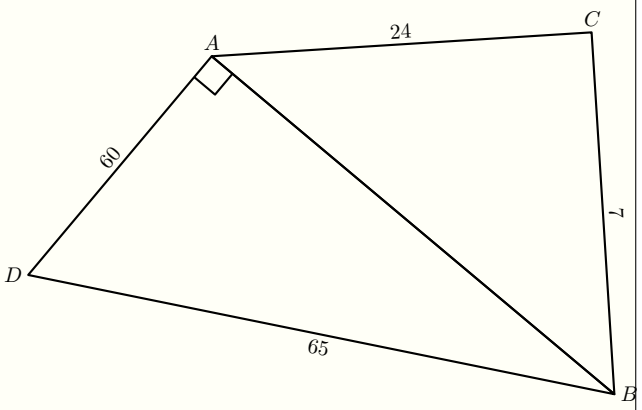
- Sur le second exercice, l'égalité de Pythagore ne sera pas vérifiée dans le triangle ABC avec une différence entre les deux membres de 10^{-2}.
  Ainsi, nous confronterons les élèves au dilemme : l'erreur étant peu "significative", le triangle est-il quand même rectangle ?

Exercice 1

On considère la figure ci-contre composée de deux triangles ABC et ABD ayant pour côté commun $[AB]$. On connait les mesures suivantes

$AC=24$; $BC=7$; $AD=60$; $BD=65$

Le triangle ABC est-il rectangle ? Justifier votre réponse

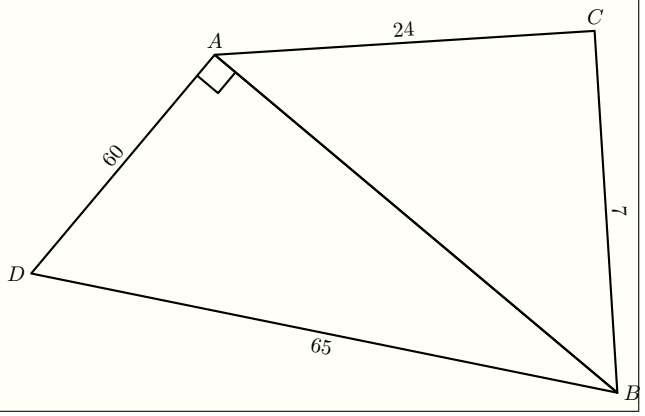


Exercice 2

On considère la figure ci-contre composée de deux triangles ABC et ABD ayant pour côté commun $[AB]$. On connait les mesures suivantes

$AC=18{,}5$; $BC=7{,}6$; $AD=25$; $BD=15$

Le triangle ABC est-il rectangle ? Justifier votre réponse



- Exercice 1 :
  L'égalité de Pythagore est vérifiée : $65^2-60^2=24^2+7^2$
  Le triangle ABC est un triangle rectangle en C.
- Exercice 2 :
  L'égalité de Pythagore ne sera pas vérifiée dans le triangle ABC :
  $25^2-15^2=400$; $18{,}5^2+7{,}6^2=400{,}01$
  On en déduit que le triangle ABC n'est pas un triangle rectangle.

Pour construire l'algorithme, j'ai associé une variable à chaque longueur des segments composant la configuration *(voir figure ci-contre)*.

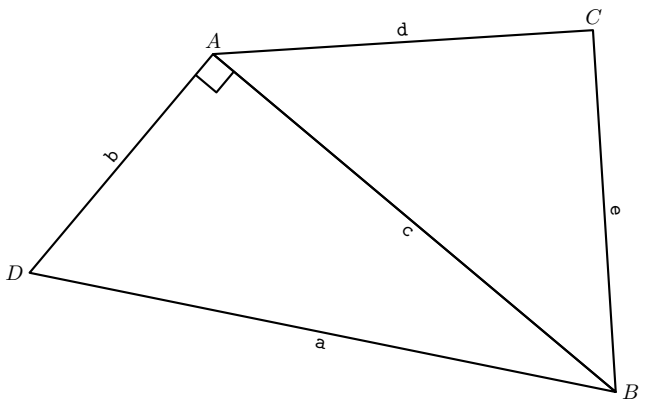


Fig. 4

- Une double-boucle teste plusieurs valeurs des longueurs AB et AD afin que le triangle ABC soit rectangle et que la longueur BD soit entière.
  Voici le test utilisé :
  `if (ceiling(c)#c) then go(fin),`
- Une seconde double-boucle teste des valeurs possibles des longueurs AC et BC afin qu'on ait l'égalité : $AB^2 = AC^2 + BC^2 \pm 1$
  Les couples de valeurs ne vérifiant pas cette contrainte sont écartés par la condition :
  `(f=0) or (abs(f)>1)`
  De plus, pour accentuer l'effet "erreur d'arrondi" devant les élèves, je demande à WxMaxima de n'afficher que les valeurs ayant 0 sur leur chiffre des unités et des dizaines par la condition :
  `floor(c^1000)#c^2/1000)`
  Ainsi, le triangle ABC n'est pas rectangle et l'erreur apparaîtra aux élèves de 0,01

Voici les deux codes ayant servis pour ce programme :

- Avec le premier programme, on conclut l'exercice en disant que ABC est rectangle ;

```
for a:1 thru 120 step 1 do
for b:1 thru a step 1 do block(
  c:sqrt(a^2-b^2),
  if (ceiling(c)#c) then go(fin),
  for d:1 thru (c-1) step 1 do block(
    e:sqrt(c^2-d^2),
    if (ceiling(e)#e) then go(fin2),
    display([a,b,c,d,e]),
    fin2
  ),
  fin
);
```

- tandis qu'avec le second programme, la valeur trouvée pour e permet d'obtenir un triangle ABC non-rectangle, mais sa valeur est proche à 10^{-2} de celle rendant le triangle ABC rectangle induisant en erreur les élèves utilisant des valeurs approchées.

```
for a:1 thru 300 do
for b:1 thru (a-1) do block(
  c:sqrt(a^2-b^2),
  if (ceiling(c)#c) then go(fin),
  for d:2 thru (c) do
  for e:2 thru (d-1) do block(
    f:(c^2-d^2-e^2),
    if (f=0) or (abs(f)>1) or (floor(c^2/1000)#c^2/1000)
      then go(fin2),
    display([a,b,c,c^2,d,e,f]),
    fin2
  ),
  fin
);
```

3. Fractions rationnelles - Seconde :

Dans cette partie, nous allons utiliser également les valeurs initiales de l'énoncé comme variables didactiques.

Exercice 1

Résoudre l'équation : $\frac{3\cdot x-2}{2-2\cdot x}+\frac{3-x}{3\cdot x+3}=0$

Exercice 2

Résoudre l'équation : $\frac{x-3}{x+1}+\frac{1}{x}=0$

Exercice 3

Résoudre l'équation : $\frac{x-2}{x-3}+\frac{x-3}{3\cdot x}=0$

Exercice 4

Résoudre l'équation : $\frac{x-2}{x+2}+x=0$

Exercice 5

Résoudre l'équation : $\dfrac{2\cdot x-1}{3\cdot x}-\dfrac{x-3}{3\cdot x+3}=0$

- Exercice 1 :

  La mise en commun au même dénominateur donne l'expression :
  $$\frac{3\cdot x-2}{2-2\cdot x}+\frac{3-x}{3\cdot x+3}=\frac{x\cdot(5-11\cdot x)}{6\cdot(x-1)(x+1)}$$
  La factorisation du numérateur est facilitée par la valeur nulle du terme numérique.

- Exercice 2 :

  La mise en commun au même dénominateur donne l'expression :
  $$\frac{x-3}{x+1}+\frac{1}{x}=\frac{(x-1)^2}{x\cdot(x+1)}$$
  L'expression du dénominateur est une identité remarquable famillière aux élèves.

- Exercice 3 :

  La mise en commun au même dénominateur donne l'expression :
  $$\frac{x-2}{x-3}+\frac{x-3}{3\cdot x}=\frac{(2\cdot x-3)^2}{3x\cdot(x-3)}$$
  On peut rendre plus difficile la résolution en trouvant un numérateur factorisable par une identité remarquable moins triviale.

- Exercice 4 :

  La mise en commun au même dénominateur donne l'expression :
  $$\frac{x-2}{x+2}+x=\frac{(x+2)(2x-1)}{2\cdot(x+1)}$$
  La résolution ne peut s'achever qu'avec l'utilisation du discriminant. L'exemple choisit ici donne des racines entières

- Exercice 5 :

  La mise en commun au même dénominateur donne l'expression :
  $$\frac{2\cdot x-1}{3\cdot x}-\frac{x-3}{3\cdot x+3}=\frac{x^2+4\cdot x-1}{3x\cdot(x+1)}$$
  L'utilisation du discriminant est toujours nécessaire ici, mais on augmente la difficulté en cherchant des racines nécessitant une simplification du radical et du quotient. L'ensemble des solutions est : $\mathcal{S}=\{-2-\sqrt{5}\,;2+\sqrt{5}\}$

Le code ci-dessous m'a permis d'obtenir les équations des trois premiers exercices ci-dessus.

```
1 mmax:3;
2 for a:1 thru mmax do
3 for b:-mmax thru mmax do
```

```
for c:-mmax thru mmax do
for d:-mmax thru mmax do
for e:-mmax thru mmax do
for f:-mmax thru mmax do
for g:-mmax thru mmax do
for h:-mmax thru mmax do block(
  if (a*d=b*c) or (e*h=f*g) then go(fin),
  m:(a*x+b)/(c*x+d)+(e*x+f)/(g*x+h),
  n:factor(m),
  p:num(n),
  q:expand(p),
  if (ratcoef(q,x,2)=0) then go(fin),
  display([m,n,p]),
  fin
);
```

On remarquera la condition "`if (ratcoef(q,x,2)=0) then go(fin)`" qui oblige le numérateur des expressions traitées à posséder un terme de degré 2.

Pour obtenir les exercices 4 et 5, j'ai modifié légèrement le code pour que le numérateur ait deux solutions et qu'elles ne soit pas imaginaire en rajoutant la structure conditionnelle suivante :

```
p:solve(n), if (length(p)#2) or (imagpart(part(p,1,2))#0) then go(fin),
```

4. Repérage - Seconde :

Dans cet exemple, nous souhaitons déterminer les coordonnées des sommets d'un triangle équilatéral dans le plan muni d'un repère orthonormé.

Pour cela, nous allons demander à WxMaxima d'étudier des centaines d'exemples et nous ne retiendrons que des situations dont la complexité doit être liée à un apprentissage.

Exercice 1

Dans le plan muni d'un repère $(O\,;I\,;J)$ orthonormé, on considère les trois points A, B, C de coordonnées :

$A(3\,;-1)$; $B(-3\,;1)$; $C(-\sqrt{3}\,;-3\cdot\sqrt{3})$

Justifier que le triangle ABC est un triangle équilatéral.

Exercice 2

Dans le plan muni d'un repère $(O\,;I\,;J)$ orthonormé, on considère les trois points A, B, C de coordonnées :

$A\left(\frac{3}{2}\,;2\right)$; $B\left(\frac{1}{2}\,;0\right)$; $C\left(1-\sqrt{3}\,;\frac{\sqrt{3}}{2}+1\right)$

Justifier que le triangle ABC est un triangle équilatéral.

Pour le second exercice, l'apparition des quotiens le destine à des élèves ayant déjà un bon niveau en algèbre.

Voici le code ayant permis d'obtenir les coordonnées des sommets des triangles équilatérals

```
mmax:2;
pas:1/2;
for a:-mmax step pas thru mmax do
for b:-mmax step pas thru mmax do
for c:-mmax step pas thru mmax do
for d:-mmax step pas thru mmax do block(
  if (a=c) or (b=d) then go(loop),
  w:solve((x-a)^2+(y-b)^2=(x-c)^2+(y-d)^2,y),
  g:part(w,1,2),
  z:solve(ev((x-a)^2+(y-b)^2=(a-c)^2+(b-d)^2,y=g)),
  e:part(z,1,2),
  f:ev(g,x=e),
  if (denom(e)=1) then
    display([a,b,c,d,e,f])),
  loop
);
```

Les trois sommets du triangle auront pour coordonnées `(a;b)`, `(c;d)`, `(e;f)`.

5. Cercle et droites - Première :

Ici, nous utiliserons toujours la puissance de calcul de WxMaxima : à un cercle donné, nous allons calculer les points d'intersection de plus de 400 droites avec ce cercle.
Nous ne retiendrons que les droites dont les coordonnées des points d'intersection avec le cercle sont des nombres décimaux possédant un chiffre dans leur partie décimale.

Exercice 1

Dans le plan muni d'un repère $(O\,;\,I\,;\,J)$ orthonormé, on considère le cercle $\mathscr{C}$ admettant le segment $[AB]$ pour diamètre où les points A et B ont pour coordonnées :

$A(3\,;6)$; $B(6\,;3)$

et la droite (d) d'équation réduite : $y=x-2$

Déterminer les coordonnées des points d'intersection du cercle $\mathscr{C}$ et de la droite (d).

Voici le code utilisé :

```
mmax:2;
pas:1/2;
m:expand((x-3)*(x-6)+(y-3)*(y-1));
for a:-mmax thru mmax step pas do
for b:-mmax thru mmax step pas do block(
  n:ev(m,y=a*x+b),
  p:solve(n),
  q:part(p,1,2),
  if (imagpart(q)#0) or (ceiling(q*10)#q*10) then go(fin),
  display([y=a*x+b,p]),
  fin
);
```

La condition `(imagpart(q)#0) or (ceiling(q*10)#q*10)` permet d'éliminer :

- les solutions de l'équation ayant une partie imaginaires *(la droite et le cercle ne s'interceptent pas)*
- et celle qui ont une partie décimale possédant plus d'un chiffre.

6. Approche du nombre dérivé - Première :

Dans ce paragraphe, nous allons utiliser la capacité de WxMaxima à calculer avec des nombres décimaux à virgule flottante possédant un grand nombre de chiffres dans la partie décimale. Nous allons observer la convergence des taux d'accroissement lors du calcul d'un nombre dérivé.

WxMaxima va nous aider à formater une chaîne de caractères que nous intégrerons dans un document LaTeX afin de produire un document qui illustrera notre cours sur le nombre dérivé en classe de première.

Voici le code utilisé :

```
kill(all);
m:x^2-x+1;
mm:ev(m,x=1);
for i:0 thru 10 do (
h:10^(-i),
xx:1-h,
xxx:ev(m,x=xx),
yyy:xxx-mm,
xxxx:yyy/(-h),
printf(true,
  "~f&\\dfrac{f(~f)-f(1)}{~f-1}=\\dfrac{~f-~f}{~f}=\\dfrac{~f}{~f}=~f\\cr~%n",
  h,xx,xx,xxx,mm,-h,yyy,-h,xxxx)
);
```

La chaîne de caractère produite par WxMaxima est copier/coller dans un document LaTeX pour produire le tableau ci-dessous :

$\ell.0$	$h=1$	$\dfrac{f(0)-f(1)}{0-1}$	$=\dfrac{1-1}{-1}$	$=\dfrac{0}{-1}$	$=0$
$\ell.1$	$h=0{,}1$	$\dfrac{f(0{,}9)-f(1)}{0{,}9-1}$	$=\dfrac{0{,}91-1}{-0{,}1}$	$=\dfrac{-0{,}09}{-0{,}1}$	$=0{,}9$
$\ell.2$	$h=0{,}01$	$\dfrac{f(0{,}99)-f(1)}{0{,}99-1}$	$=\dfrac{0{,}9901-1}{-0{,}01}$	$=\dfrac{-0{,}0099}{-0{,}01}$	$=0{,}99$
$\ell.3$	$h=0{,}001$	$\dfrac{f(0{,}999)-f(1)}{0{,}999-1}$	$=\dfrac{0{,}999001-1}{-0{,}001}$	$=\dfrac{-0{,}000999}{-0{,}001}$	$=0{,}999$
$\ell.4$	$h=0{,}0001$	$\dfrac{f(0{,}9999)-f(1)}{0{,}9999-1}$	$=\dfrac{0{,}99990001-1}{-0{,}0001}$	$=\dfrac{-0{,}00009999}{-0{,}0001}$	$=0{,}9999$
$\ell.5$	$h=0{,}00001$	$\dfrac{f(0{,}99999)-f(1)}{0{,}99999-1}$	$=\dfrac{0{,}9999900001-1}{-0{,}00001}$	$=\dfrac{-0{,}0000099999}{-0{,}00001}$	$=0{,}99999$
$\ell.6$	$h=0{,}000001$	$\dfrac{f(0{,}999999)-f(1)}{0{,}999999-1}$	$=\dfrac{0{,}999999000001-1}{-0{,}000001}$	$=\dfrac{-0{,}000000999999}{-0{,}000001}$	$=0{,}999999$
$\ell.7$	$h=0{,}0000001$	$\dfrac{f(0{,}9999999)-f(1)}{0{,}9999999-1}$	$=\dfrac{0{,}99999990000001-1}{-0{,}0000001}$	$=\dfrac{-0{,}00000009999999}{-0{,}0000001}$	$=0{,}9999999$
$\ell.8$	$h=0{,}00000001$	$\dfrac{f(0{,}99999999)-f(1)}{0{,}99999999-1}$	$=\dfrac{0{,}99999999-1}{-0{,}00000001}$	$=\dfrac{-0{,}0000000099999999}{-0{,}00000001}$	$=0{,}99999999$
$\ell.9$	$h=0{,}000000001$	$\dfrac{f(0{,}999999999)-f(1)}{0{,}999999999-1}$	$=\dfrac{0{,}999999999-1}{-0{,}000000001}$	$=\dfrac{-0{,}000000000999999999}{-0{,}000000001}$	$=0{,}999999999$
$\ell.10$	$h=0{,}0000000001$	$\dfrac{f(0{,}9999999999)-f(1)}{0{,}9999999999-1}$	$=\dfrac{0{,}9999999998999999-1}{-0{,}0000000001}$	$=\dfrac{-0{,}000000000999999999}{-0{,}0000000001}$	$=0{,}9999999999$

On remarquera une erreur de WxMaxima sur les valeurs arrondies à la ligne $\ell.8$:

$$f(0{,}99999999)=0{,}9999999900000001$$

WxMaxima va nous permettre d'augmenter le nombre maximal de décimales utilisées dans

l'écriture des nombres en virgule flottante.

Pour cela :

- on indique la précision avec laquelle WxMaxima va calculer sur les "*big floats*" :

```
fpprec:100;
```

- on modifie la ligne 11 pour lui demander d'insérer les valeurs en "*big float*" :

```
"~f&\dfrac{f(~h)-f(1)}{~h-1}&\dfrac{~h-~h}{~h}&\dfrac{~h}{~h}&~h\cr~%",
```

Voici maintenant le tableau de convergence des taux d'accroissement correctement rempli :

$\ell.0$	$h = 1{,}0$	$\frac{f(0)-f(1)}{0-1}$	$= \frac{1-1}{-1}$	$= \frac{0}{-1}$	$= 0$
$\ell.1$	$h = 0{,}1$	$\frac{f(0{,}9)-f(1)}{0{,}9-1}$	$= \frac{0{,}91-1}{-0{,}1}$	$= \frac{-0{,}09}{-0{,}1}$	$= 0{,}9$
$\ell.2$	$h = 0{,}01$	$\frac{f(0{,}99)-f(1)}{0{,}99-1}$	$= \frac{0{,}9901-1}{-0{,}01}$	$= \frac{-0{,}0099}{-0{,}01}$	$= 0{,}99$
$\ell.3$	$h = 0{,}001$	$\frac{f(0{,}999)-f(1)}{0{,}999-1}$	$= \frac{0{,}999001-1}{-0{,}001}$	$= \frac{-0{,}000999}{-0{,}001}$	$= 0{,}999$
$\ell.4$	$h = 0{,}0001$	$\frac{f(0{,}9999)-f(1)}{0{,}9999-1}$	$= \frac{0{,}99990001-1}{-0{,}0001}$	$= \frac{-0{,}00009999}{-0{,}0001}$	$= 0{,}9999$
$\ell.5$	$h = 0{,}00001$	$\frac{f(0{,}99999)-f(1)}{0{,}99999-1}$	$= \frac{0{,}9999900001-1}{-0{,}00001}$	$= \frac{-0{,}0000099999}{-0{,}00001}$	$= 0{,}99999$
$\ell.6$	$h = 0{,}000001$	$\frac{f(0{,}999999)-f(1)}{0{,}999999-1}$	$= \frac{0{,}999999000001-1}{-0{,}000001}$	$= \frac{-0{,}000000999999}{-0{,}000001}$	$= 0{,}999999$
$\ell.7$	$h = 0{,}0000001$	$\frac{f(0{,}9999999)-f(1)}{0{,}9999999-1}$	$= \frac{0{,}99999990000001-1}{-0{,}0000001}$	$= \frac{-0{,}00000009999999}{-0{,}0000001}$	$= 0{,}9999999$
$\ell.8$	$h = 0{,}00000001$	$\frac{f(0{,}99999999)-f(1)}{0{,}99999999-1}$	$= \frac{0{,}9999999900000001-1}{-0{,}00000001}$	$= \frac{-0{,}0000000099999999}{-0{,}00000001}$	$= 0{,}99999999$
$\ell.9$	$h = 0{,}000000001$	$\frac{f(0{,}999999999)-f(1)}{0{,}999999999-1}$	$= \frac{0{,}999999999000000001-1}{-0{,}000000001}$	$= \frac{-0{,}000000000999999999}{-0{,}000000001}$	$= 0{,}999999999$
$\ell.10$	$h = 0{,}0000000001$	$\frac{f(0{,}9999999999)-f(1)}{0{,}9999999999-1}$	$= \frac{0{,}99999999990000000001-1}{-0{,}0000000001}$	$= \frac{-0{,}00000000009999999999}{-0{,}0000000001}$	$= 0{,}9999999999$

- WxMaxima n'est pas limité sur le nombre de décimales utilisées. Nous aurions pu continuer à afficher d'autres résultats. Seul la taille de la page à limiter le tableau.
- Voici un petit code permettant d'illustrer l'usage des nombres décimaux habituel dans WxMaxima et l'usage des "*big floats*" :

```
1 fpprec:1000;
2 fpprintprec:0;
3 set_display('none);
4 float(1/7);
5 bfloat(1/7);
6 bfloat(%pi);
```

GnuPlot

E. Présentation :

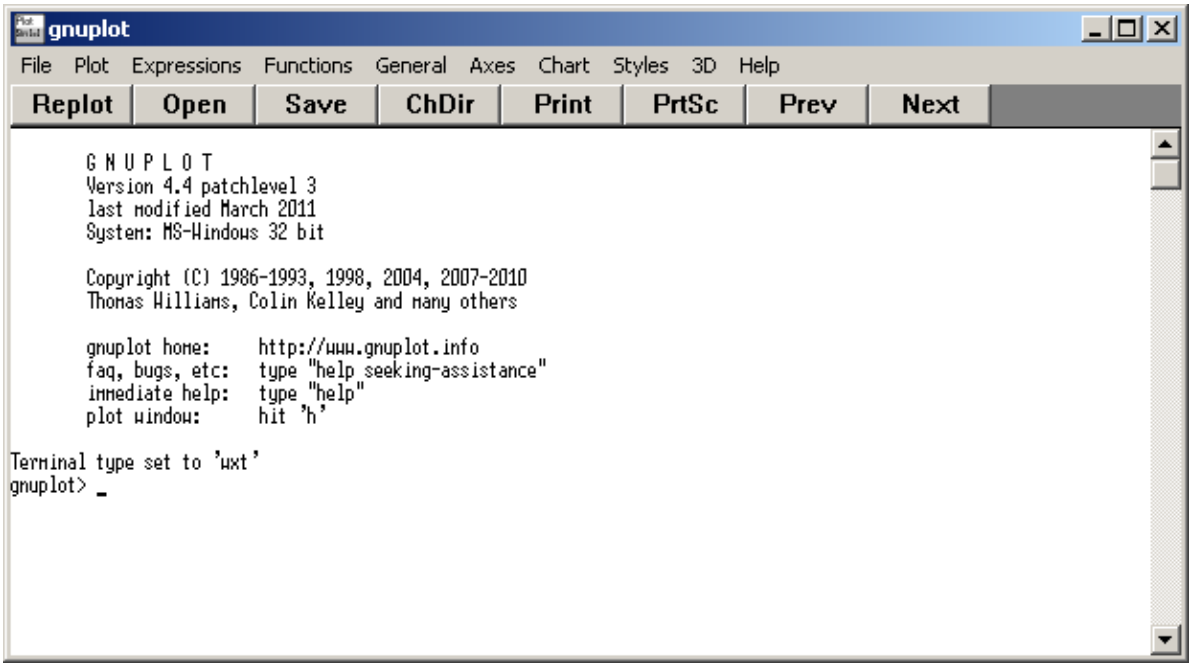
Il est possible de tracer des courbes dans Maxima à l'aide des commandes `plot2d` et `plot3D`. Maxima sous-traite au logiciel GnuPlot la production des représentations graphiques.

GnuPlot est un logiciel de tracer de courbes en ligne de commande donc asser difficile à utiliser directement. WxMaxima sera notre interface pour produire ces représentations graphiques. En sortie, il permet de créer des fichiers en différents formats : *jpg*, *png*, *pdf*, *ps*...

GnuPlot est complet et assez complexe à utiliser. Nous nous limiterons à l'étude des fonctions de base de ce logiciel afin de pouvoir représenter des courbes en 2 ou 3 dimensions ou des diagrammes *(comme la loi de probabilité)*.
Nous verrons comment en sortie enregistrer l'image en "*.jpg*" ou '*.pdf*'.

Dans la version Windows de Maxima, le logiciel GnuPlot se trouve à l'emplacement :

`C:\ProgramFiles\Maxima-5.25.0\gnuplot\wgnuplot.exe`



Les menus de cette fenêtre permettent d'accéder aux commandes de GnuPlot, mais nous allons, dans ce tutorial, utiliser principalement la ligne de commande, repérable par le préfixe :

`gnuplot>`.

Il est également possibl d'écrire les paramètres saisis dans GnuPlot dans un fichie externe et de charger ce dernier dans GnuPlot.

L'exercice ci-dessous en est le premier exemple :

1. Créer un dossier à l'emplacement `C:\gnuplotExemple`.
2. Créer un fichier texte nommé "`premier.txt`" contenant la ligne suivante :

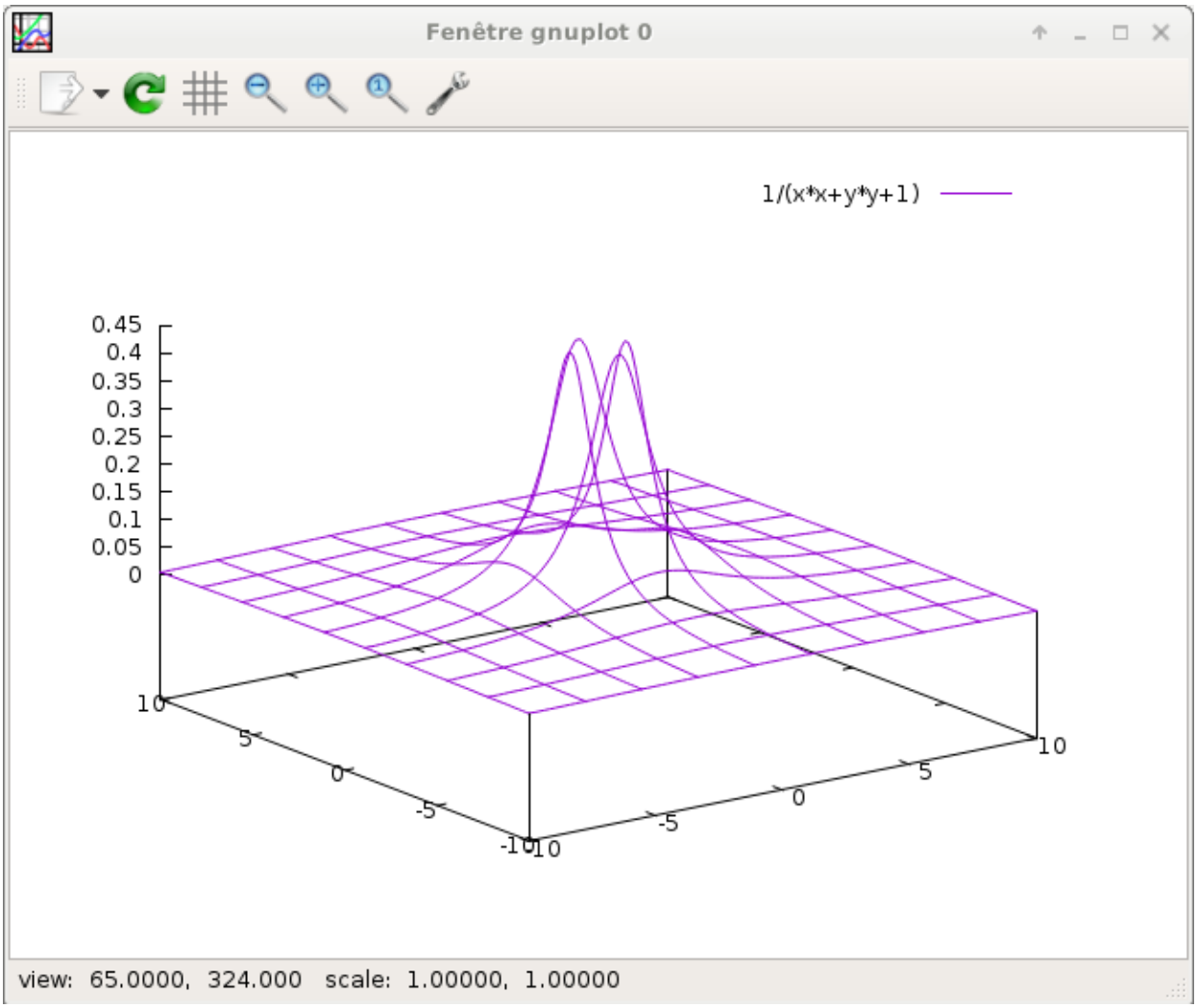
```
splot 1/(x*x+y*y+1)
```

3. Maintenant dans Gnuplot :
   a. On demande à GnuPlot de changer de répertoire courant :

```
cd 'C:\gnuplotExemple'
```

   b. On exécute le fichier en lançant la commande :

```
load 'premier.txt'
```



⚠ Le premier chargement du fichier peut être long, il faut patienter.

F. Courbes en deux dimensions :

1. Quelques commandes :

Exercice 2

Ouvrez une nouvelle session de GnuPlot, pour repartir avec les paramètres par défaut de ce logiciel et pour chacune des questions, saissisez la commande et laissez un commentaire sur l'effet de celle-ci sur la représentation graphique :

1. `plot sin(x)`

.....

2. `set xrange[0:5]; replot;`

.....

3. `set yrange[-2:2]; replot;`

.....

4. `set autoscale y; replot;`

.....

5. `set xtics 0.5; replot;`

.....

6. `set grid; replot;`

.....

7. `set grid linetype 1 linecolor rgb "cyan"; replot;`

.....

8. `unset grid; replot;`

.....

9. `reset; replot;`

.....

10. `plot [0:2] [-2:2] sin(x), 0.5*cos(x); replot;`

.....

11. `plot [0:5] [-1:1] sin(x) with lines linecolor rgb "pink", 0.5*cos(x) with boxe`
    `replot`

.....

12. `plot [0:2] x**2 title 'Fonction carre', log(x) title 'Fonction logarithme'`

.....

13. `plot for [i=0:5:2] sin(x*i) title sprintf("sin(%dx)",i)`

14. `set samples 5; replot;`

15. `set samples 300; replot;`

Voici quelques commandes rencontrées précédemment :

- `plot` : permet d'effectuer le tracé d'une courbe :

  ```
  plot sin(x);
  ```

  ou d'effectuer le tracé de plusieurs fonctions :

  ```
  plot sin(x), 0.5*cos(x), x**2;
  ```

  On remarque que l'opérateur `**` est l'opérateur puissance.

  Il est possible de définir automatiquement les bornes d'affichages sur les abscisses :

  ```
  plot [0:3] sin(x);
  ```

  et en même temps l'axe des ordonnées :

  ```
  plot [0:3] [-1:1] sin(x);
  ```

  On peut définir le style du tracé de la courbe :

  ```
  plot sin(x) with boxes linecolor rgb "pink";
  ```

  On peut imposer la légende de la courbe :

  ```
  plot sin(x) title "Sinus";
  ```

  Il est également possible de tracer plusieurs courbes en faisant varier le paramètre :

  ```
  plot for [i=-3:3] i*sin(x) title "Sinus";
  ```

- `replot` : redessine le graphique avec les nouveaux paramètres ;
- `reset` : affecte les valeurs par défaut aux différents paramètres de GnuPlot ;
- `set` : permet d'affecter de valeurs aux paramètres d'affichage. `unset` permet de ré-initialiser un paramètre avec sa valeur par défaut.

Voici quelques paramètres rencontrés précédemment :

- `sample` : nombres de points utilisés pour le tracé des courbes ;
- `xrange` et `yrange` : permet de choisir les bornes d'affichages sur les abscisses et sur les ordonnées ;
- `autoscale y` : GnuPlot choisira les bornes de l'axe des ordonnées pour un affichage optimal de la courbe ;
- `xtics` et `ytics` : définit l'espace séparant les graduations respectivements sur les axes des

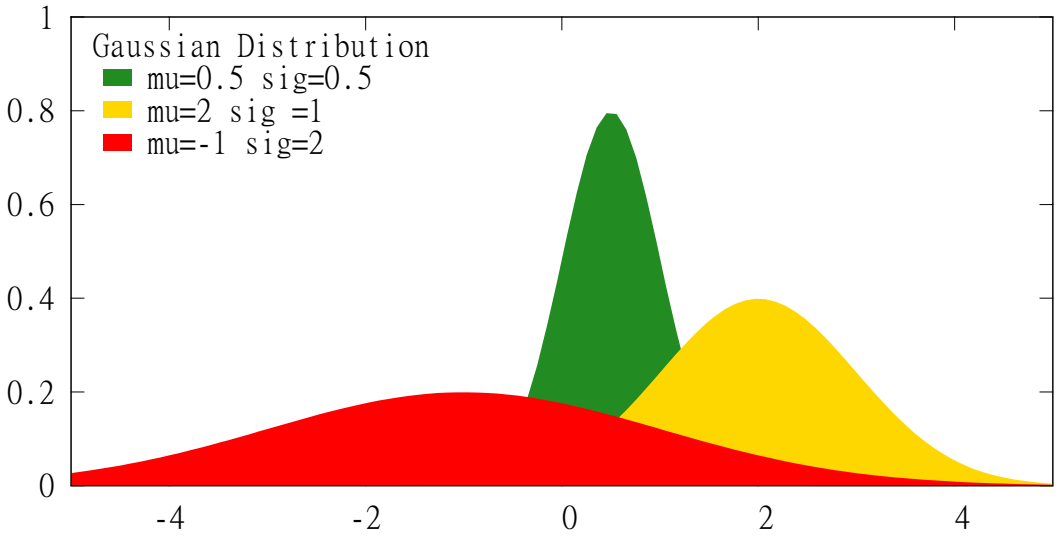
abscisses et des ordonnées ;

2. Un bel exemple :

- Voici un exemple tiré de la page :

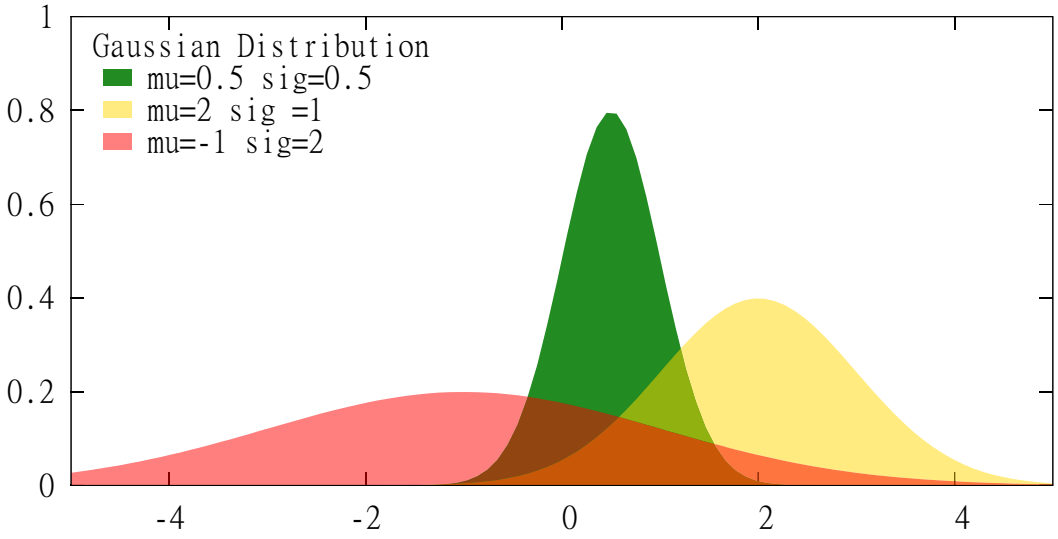
  http://gnuplot.sourceforge.net/demo/transparent.html

```
#
# Example of transparent fill areas
# Ethan A Merritt - Aug 2006
# NB:
#     Not all terminal types support transparency
#     Assumes UTF-8 support for plot titles
#
set style fill solid 1.0 noborder
set style function filledcurves y1=0
set clip two

Gauss(x,mu,sigma) = 1./(sigma*sqrt(2*pi)) * exp( -(x-mu)**2
    / (2*sigma**2) )
d1(x) = Gauss(x, 0.5, 0.5)
d2(x) = Gauss(x,  2.,  1.)
d3(x) = Gauss(x, -1.,  2.)

set xrange [-5:5]
set yrange [0:1]

unset colorbox

set key title "Gaussian Distribution"
set key top left Left reverse samplen 1

set title "Solid filled curves"
plot d1(x) fs solid 1.0 lc rgb "forest-green" title "mu =
  0.5 sig = 0.5", \
     d2(x) lc rgb "gold" title "mu =  2.0 sig = 1.0", \
     d3(x) lc rgb "red" title "mu = -1.0 sig = 2.0"
```



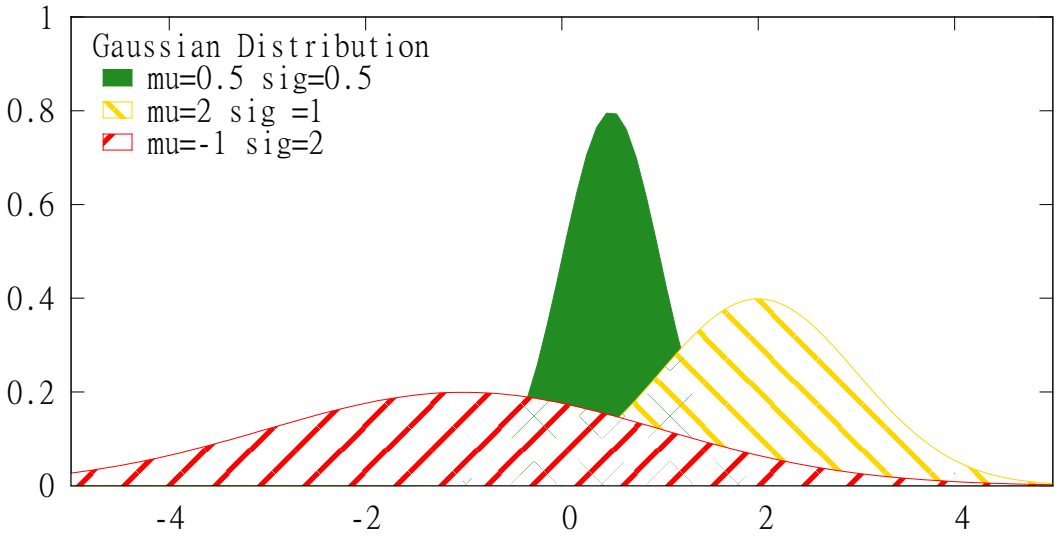
- On peut modifier l'affichage pour donner un effet de transparence aux courbes :

```
set style fill transparent solid 0.5 noborder
set title "Transparent filled curves"
replot
```



- On peut également hachurer les courbes de la figure :

```
set style fill pattern 4 bo
set title "Pattern-filled curves"
replot
```



G. Les courbes en $3D$:

1. Quelques commandes :

Exercice 3

Ouvrez une nouvelle session de GnuPlot, pour repartir avec les paramètres par défaut de ce logiciel et pour chacune des questions, saissisez la commande et laissez un commentaire sur l'effet de celle-ci sur la représentation graphique :

1. `splot x**2-y**2`

.....

2. `set isosample 30,30; replot`

.....

3. `set isosample 10,10; set view 76,27; replot;`

.....

4. `set contour; replot;`

.....

5. `set cntrparam levels 10; replot`

.....

6. `unset contour; replot;`

.....

7. `set pm3d at s; replot;`

8. `unset colorbox; replot;`

9. `set sample 10,10; set view 90,335; replot;`

10. `set pm3d hidden3d 1 depthorder; replot;`

11. `set palette gray; replot;`

12. `set palette gamma 5; replot;`

13. `set style line 1 lt rgb "black" lw 1 pt 1; replot;`

14. `unset border; replot;`

15. `unset xtics; unset ytics; unset ztics; replot;`

16. `splot x**2-y**2 notitle; replot;`

La commande `splot` permet d'effectuer le tracé de courbes en 3 dimensions.

Voici la description des paramètres rencontrés dans cet exercice :

- `isosample` permet de définir le nombre de traits utilisés sur l'axe des abscisses et des ordonnées définissant la coube.
  Si vous utilisez le module `pm3d`, il faudra utiliser le paramètre `sample`.
- `view` définit l'angle par lequel la courbe est visualisée ; la taille de la courbe est liée à la fenêtre d'affichage de GnuPlot.
  Il est possible de modifier l'angle de vue avec la souris ou avec les touches directionnelles du clavier : on peut alors voir la valeur courante de l'angle de vue en bas à gauche dans la fenêtre d'affichage de GnuPlot.

- `contour` : affiche les courbes de niveau associée à la fonction sur le plan (Oxy). La commande `cntrparam` permet de définir combien de courbes de niveau représenteront la fonction.
- Le module `pm3d` permet de colorier la surface. On l'initialise avec :

  ```
  set pm3d at s;
  ```

  La couleur des traits utilisés est définie par :

  ```
  set pm3d hidden3d 1
  ```

  L'option `depthorder` permet d'afficher la partie de courbe présent en premier plan et de cacher les parties de la courbe situées derrière.

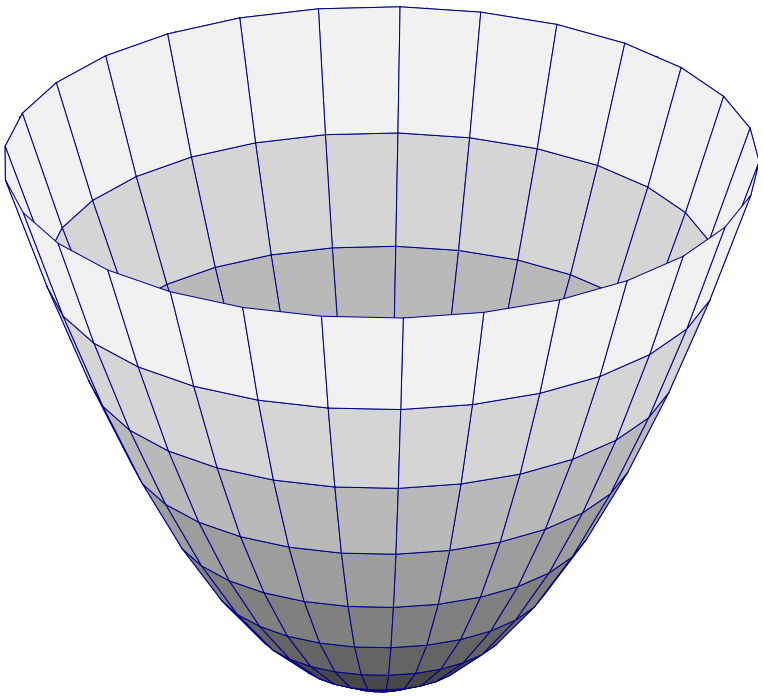
  ```
  set pm3d depthorder
  ```
- `palette` : permet de définir le mode de couleurs utilisé. On peut utiliser `gray` ou `color`. Dans le mode `gray`, on peut éclaircir ou obscursir les tons de gris utilisés à l'aide du paramètre `gamma` ;
- `set style line 1` : définie le style du trait utilisé pour tracer la courbe ;
- `unset border` : efface la boîte entourant la courbe ;
- `unset xtics; unset ytics; unset ztics;` : efface les graduations sur les axes.

2. Quelques exemples :

- Un exemple personnel :

```
set samples 30,10
unset pm3d
set pm3d at s
set pm3d hidden3d 1
set pm3d depthorder
set style line 1 lt rgb "dark-blue" lw 1 pt 1
set surf
set palette gray
set palette gamma 2
unset grid
unset colorbox
set noborder
unset xtics
unset ytics
unset ztics
```

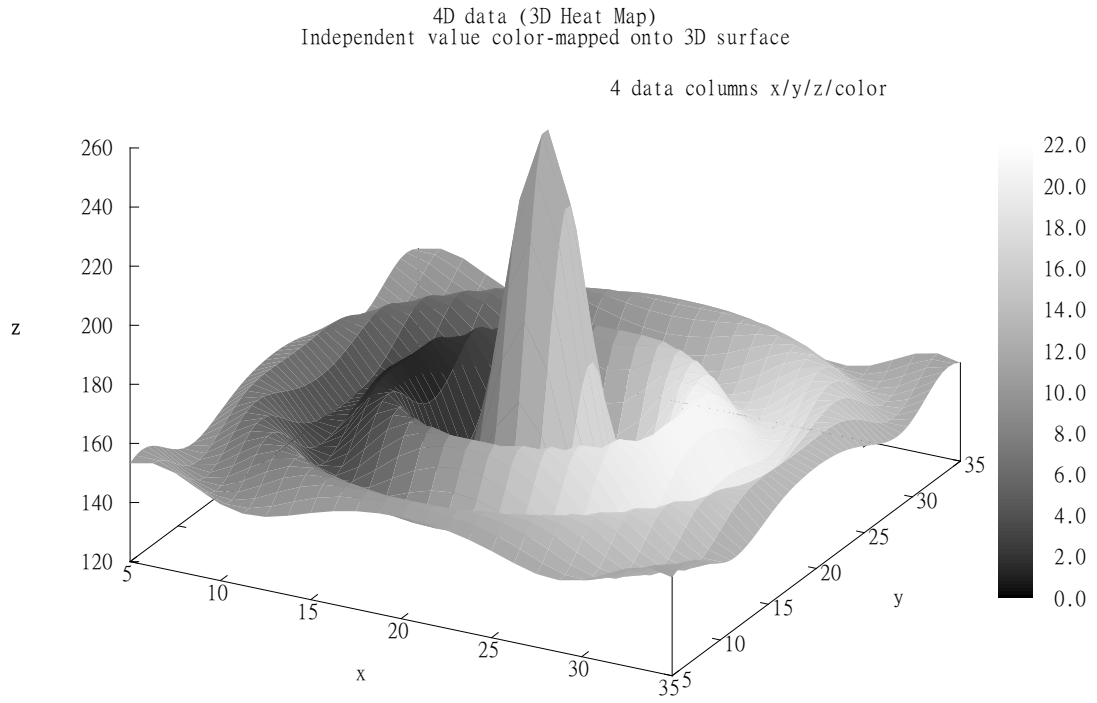
```
set xrange [-2:2]
set yrange [-2:2]
set zrange [0:4]
set urange [0:360]
set vrange[0:2]
set view 59,37
set lmargin at screen 0
set rmargin at screen 1
set bmargin at screen 0.0
set tmargin at screen 0.8
set parametric
set angles degrees
splot abs(v)*cos(u),abs(v)*sin(u),v*v with lines ls 1
```



- Un exemple issu du site http://gnuplot.sourceforge.net/demo/heatmaps.html

```
#
# Demonstrate use of 4th data column to color a 3D surface.
# Also demonstrate use of the pseudodata special file '++'.
# This plot is nice for exploring the effect of the 'l' and 'L' hotkeys.
#
set view 49, 28, 1, 1.48
```

```
set xrange [ 5 : 35 ] noreverse nowriteback
set yrange [ 5 : 35 ] noreverse nowriteback
# set zrange [ 1.0 : 3.0 ] noreverse nowriteback
set ticslevel 0
set format cb "%4.1f"
set colorbox user size .03, .6 noborder
set cbtics scale 0

set samples 25, 25
set isosamples 50, 50

set title "4D data (3D Heat Map)"\
          ."\nIndependent value color-mapped onto 3D surface"  offset 0,1
set xlabel "x" offset 3, 0, 0
set ylabel "y" offset -5, 0, 0
set zlabel "z" offset 2, 0, 0
set pm3d implicit at s

Z(x,y) = 100. * (sinc(x,y) + 1.5)
sinc(x,y) = sin(sqrt((x-20.)**2+(y-20.)**2))/sqrt((x-20.)**2+(y-20.)**2)
color(x,y) = 10. * (1.1 + sin((x-20.)/5.)*cos((y-20.)/10.))

splot '++' using 1:2:(Z($1,$2)):(color($1,$2)) with pm3d title "4 data columns x/y/z/color"
```



H. En plus :

1. Ecriture dans un fichier :

Il est possible de créer un fichier contenant le graphique. Voici quelqu'uns des formats disponibles :

- `gif` : on saisira le code suivant :

```
set terminal gif size 500,300; set output 'dessin.gif';
replot; unset output;
```

- On crée un fichier `png` à l'aide du code :

```
set terminal png size 500,300; set output 'dessin.png';
replot; unset output;
```

- Les fichiers `pdf`, présentant l'avantage d'offrir la présence de dessin vectoriel s'effectue via

```
set terminal pdf size 10cm,5cm; set output 'dessin.pdf';
replot; unset output;
```

Pour revenir à l'affichage par défaut de GnuPlot, il faut saisir :

```
set terminal wxt; replot;
```

2. Quelques commandes :

- `show all` : affiche tout le paramètrage de GnuPlot. Mais voici quelques unes de ces parties les plus intéressantes :
  - `show colornames` : affiche les couleurs disponibles ;
  - `show angles` : affiche l'unité des angles actuellement utilisée par GnuPlot ;
  - `show autoscale` : indique sur l'échelle des axes est automatique ou non ;
  - `show decimalsign` : indique le caractère utilisé pour séparer partie entière et partie décimale d'un nombre ;
- `set parametric` : passe le mode de creation des graphiques pour les courbes paramétriques.