Prise en main de *Xcas*

Ce document est rédigé à partir des manuels d'accompagnement de Xcas (R. De Graeve) et du Tutoriel de calcul formel (R. De Graeve, B. Parisse et B. Ycart), qu'on trouve dans l'*Aide* du logiciel.

Que voit-on au démarrage ?

On obtient au démarrage l'ouverture d'une session avec de haut en bas :

- une barre de menu ;
- juste en dessous, un onglet Unnamed, qui sera remplacé par un nom de fichier, celui qui sera donné lors de la sauvegarde ;
- une barre de boutons :
  - un bouton ? pour l'Aide ;
  - un bouton Save pour sauvegarder la session, de couleur rouge si la session a été modifiée depuis la dernière sauvegarde, de couleur verte sinon ;
  - un bouton affichant le mode de fonctionnement du logiciel et permettant de le modifier ;
  - le bouton STOP permettant d'arrêter un calcul trop long ;
  - un bouton Kbd pour faire apparaître ou disparaître le clavier virtuel.

Entrer des commandes

L'exécution d'une ligne se fait simplement par la touche de validation. Si l'on ne souhaite pas afficher le résultat, on termine la ligne de commande par :; et l'on valide. On peut éditer plusieurs commandes à la file avant leur exécution à condition de les séparer par un point-virgule.

Entrer les commandes suivantes et noter ce qu'elles permettent d'obtenir :

- ax + b
- 2 x + b
- 1/3+1/4
- sqrt(2)^5
- solve(a*x^2+b*x+c,x)
- 50 !
- graphe(x^2-3)
- point(1,2)
- int(x^2-1,x,-1,1)
- desolve(y'=y,y)
- a:=3%5;a+12
- plotfunc(x+y^2,[x=-5..5,y=-2..2],xstep=0.5,ystep=0.1, affichage=vert+rempli)
- e
- evalf(e,5)

Toutes les commandes sont gardées en mémoire. On peut remonter dans l'historique de la session pour modifier des commandes antérieures. Faire des modification dans les commandes précédentes et observer. Par exemple, modifier la commande ci-dessus de résolution d'une équation en solve(a*x+b*x+c,x)…

Aide sur une commande

Si l'on ne connaît pas la commande, on peut :
- chercher dans les menus par thèmes une fonction
- ou ouvrir le manuel de calcul formel ou un autre manuel, dans le menu Aide
- ou chercher un mot-clé, dans le menu Aide
- ou chercher dans l'index des commandes (menu Aide ou bouton ?)

Si l'on connaît la commande et qu'on cherche à connaître les arguments :
- on utilise le bouton ?, suivi du nom de la commande, qu'on valide,
- ou, dans une ligne de commande, on saisit la commande (par exemple, fonction) et on utilise le bouton ?…

Interrompre un calcul

Cliquer sur le bouton STOP en haut et à droite. Cette commande est importante car la complexité d'un calcul peut vite devenir effrayante (même pour un ordinateur).

Configuration

Le mode de fonctionnement du logiciel est affiché dans un bouton en haut de la session (au milieu) : nombre de chiffres significatifs, unité d'angle, mode réel ou complexe, mode exact ou approché… On peut modifier ces éléments en cliquant sur ce bouton ou par le menu Cfg.

Gérer sa session de travail

xcas permet d'ouvrir plusieurs sessions de travail. Chacune d'elles est située dans un onglet et peut être sauvegardée sur le disque dur. Au lancement une session Unnamed est ouverte. Chaque session de travail contient des niveaux d'entrée numérotés de types divers : une ligne de commande, un éditeur de programmes, un tableur (avec ou sans graphique), un éditeur d'équations, un écran de géométrie (2-d ou 3-d) ou un groupe constitué de plusieurs de ces niveaux…

Les objets du calcul formel

Les nombres peuvent être exacts ou approchés. Les nombres exacts sont les constantes prédéfinies, les entiers, les fractions d'entiers et, plus généralement, toute expression ne contenant que des entiers et des constantes, comme sqrt(2)*e^(i*pi/3). Les nombres approchés sont notés avec la notation scientifique standard : partie entière suivie du point de séparation et partie fractionnaire (éventuellement suivie de e et d'un exposant). Par exemple, 2 est un entier exact, 2.0 est la version approchée du même entier ; 1/2 est un rationnel, 0.5 est la version approchée du même rationnel.

On passe d'une valeur exacte à une valeur approchée par evalf, on transforme une valeur approchée en un rationnel exact par exact. Les calculs sont effectués en mode exact si tous les nombres qui interviennent sont exacts. Ils sont effectués en mode approché si l'un des

nombres est approché. Ainsi 1.5+1 renvoie un nombre approché alors que 3/2+1 renvoie un nombre exact.

Entrer les commandes suivantes et noter ce qu'elles permettent d'obtenir :
- evalf(sqrt(2))
- sqrt(2)-evalf(sqrt(2))
- exact(evalf(sqrt(2)))*10^9
- exact(evalf(sqrt(2)*10^9))

Pour les nombres réels approchés, la précision par défaut est proche de 14 chiffres significatifs. Elle peut être changée, en donnant le nombre de décimales désiré comme second argument de evalf.

Entrer les commandes suivantes et noter ce qu'elles permettent d'obtenir :
- evalf(sqrt(2),50)
- evalf(pi,100)

La lettre i est réservée au nombre complexe de carré –1 et ne peut être réaffectée ; en particulier on ne peut pas l'utiliser comme indice de boucle.

Entrer les commandes suivantes et noter ce qu'elles permettent d'obtenir :
- (1+2*i)^2
- (1+2*i)/(1-2*i)
- e^(i*pi/3)

Les variables

On dit qu'une variable est formelle si elle ne contient aucune valeur : toutes les variables sont formelles tant qu'elles n'ont pas été affectées (à une valeur).

L'affectation est notée :=. Au début de la session, la variable a est formelle, elle devient affectée après l'instruction a:=3, elle sera alors remplacée par 3 dans tous les calculs suivants ; ainsi a+1 renverra 4.

xcas conserve tout le contenu de la session. Si l'on veut que la variable a, après l'avoir affectée, soit à nouveau une variable formelle, il faut la "vider" par purge(a).

Dans toute la suite, les variables utilisées sont supposées avoir été purgées avant chaque suite de commandes.

Il ne faut pas confondre :
- le signe := qui désigne l'affectation,
- le signe == qui désigne une égalité booléenne : c'est une opération binaire qui retourne 1 ou 0 (1 si Vrai et 0 si Faux),
- le signe = utilisé pour définir une équation.

Saisir les instructions suivantes et expliquer ce qu'elles renvoient successivement :
- a==b
- a:=b

- a==b

On peut faire certaines hypothèses sur une variable avec la commande assume, par exemple assume(a>2). Une hypothèse est une forme spéciale d'affectation, elle efface une éventuelle valeur précédemment affectée à la variable. Lors d'un calcul, la variable n'est pas remplacée, mais l'hypothèse sera utilisée dans la mesure du possible, par exemple abs(a) renverra a si l'on a fait l'hypothèse a>2.

Saisir successivement les instructions suivantes et expliquer ce qu'elles renvoient :
- sqrt(a^2)
- assume(a<0)
- sqrt(a^2)
- assume(n,integer)
- sin(n*pi)

La fonction subst permet de remplacer une variable dans une expression par un nombre ou une autre expression, sans affecter cette variable.

Saisir successivement les instructions suivantes et expliquer ce qu'elles renvoient :
- subst(a^2+1,a=1)
- subst(a^2+1,a=sqrt(b-1)
- a^2+1

Les expressions

Une expression est une combinaison de nombres et de variables reliés entre eux par des opérations : par exemple x^2+2*x+c.

Lorsqu'on valide une commande, xcas remplace les variables par leur valeur, si elles en ont une, et exécute les opérations.

Saisir dans une ligne de commande et expliquer ce qui est obtenu :

(a-2)*x^2+a*x+1:;a:=2:;(a-2)*x^2+a*x+1

Certaines opérations de simplification sont exécutées automatiquement lors d'une évaluation.

Développer et simplifier

A part les simplifications évoquées ci-dessus, xcas ne fait pas de simplification systématique. Il y a deux raisons à cela. La première est que les simplifications non triviales sont parfois coûteuses en temps et le choix d'en faire ou non est laissé à l'utilisateur ; la deuxième est qu'il y a, en général, plusieurs manières de simplifier une même expression, selon l'usage que l'on veut en faire.

Les principales commandes pour transformer une expression sont les suivantes :
- expand développe une expression en tenant compte uniquement de la distributivité de la multiplication sur l'addition et du développement des puissances entières ;
- normal et ratnormal : écrivent une fraction rationnelle (rapport de deux polynômes) sous forme de fraction irréductible développée ;

- factor écrit une fraction sous forme irréductible factorisée ;
- simplify essaie de se ramener à des variables algébriquement indépendantes avant d'appliquer normal : ceci est plus coûteux en temps et "aveugle" (on ne contrôle pas les réécritures intermédiaires) ; les simplifications faisant intervenir des extensions algébriques (par exemple des racines carrées) nécessitent parfois deux appels et/ou des hypothèses (assume) pour enlever des valeurs absolues avant d'obtenir la simplification souhaitée ;
- tsimplify essaie de se ramener à des variables algébriquement indépendantes, mais sans appliquer normal ensuite.

Les fonctions

De nombreuses fonctions sont déjà définies dans xcas, en particulier les fonctions classiques. La plupart d'entre elles ont pour argument une expression (et non une fonction). Pour créer une nouvelle fonction, il faut la déclarer à l'aide d'une expression contenant la variable.

Par exemple l'expression $x^2 - 1$ est définie par x^2-1. Pour la transformer en la fonction f qui à x associe $x^2 - 1$, trois possibilités :
- f(x):= x^2-1
- f:=x->x^2-1
- f:=unapply(x^2-1,x)

Si f est une fonction d'une variable et E est une expression, $f(E)$ est une autre expression.

Saisir les instructions suivantes et expliquer ce qu'on obtient :
- E:=x^2-1
- subst(E,x=2)
- E(2)

Définir, par exemple, la fonction g qui à x associe $5\,x^2 + 3\,x - 2$, puis saisir chacune des instructions suivantes et expliquer ce qu'on obtient :
- diff(g(x))
- int(g(x))
- taylor(sin(x),x,3)

On peut ajouter et multiplier des fonctions, par exemple f:=sin*exp. Pour composer des fonctions, on utilise l'opérateur @.

xcas dispose aussi de fonctions qui génèrent des nombres aléatoires, par exemple :
- Si n désigne un entier relatif rand(n) ou alea(n) ou hasard(n) renvoient, au hasard et de façon équiprobable, un nombre entier de $[0 ; n[$ si $n > 0$ (ou de $]n ; 0]$ si $n < 0$).
- Si a et b désignent des nombres, rand(a,b) ou alea(a..b) ou hasard(a..b) renvoient, au hasard et de façon équiprobable, un nombre décimal appartenant à $[a ; b[$.

Listes, séquences

xcas distingue :
- les listes (entre crochets) : saisir, dans trois lignes de commandes successives, 1$5, puis [k^2 $ (k=-2..2)], puis makelist(x->x^2,-2,2) et décrire ce qu'on obtient ;

- les séquences (entre parenthèses) : saisir dans une ligne de commande, seq(k^2,k=-2..2).

Ainsi pour fabriquer une liste ou une séquence, on utilise des commandes d'itération comme $ ou seq (qui itèrent une expression) ou makelist (qui définit une liste à l'aide d'une fonction).

Les listes peuvent contenir des listes (c'est le cas des matrices), alors que les séquences sont plates (un élément d'une séquence ne peut pas être une séquence). Il suffit de mettre une séquence entre crochets pour en faire une liste. Le nombre d'éléments d'une liste est donné par size (ou nops).

La séquence vide est notée NULL, la liste vide []. Pour ajouter un élément à une séquence il suffit d'écrire la séquence et l'élément séparés par une virgule. Pour ajouter un élément à une liste on utilise append. On accède à un élément d'une liste ou d'une séquence grâce à son indice mis entre crochets, le premier élément étant d'indice 0.

A titre d'exercice, en une seule ligne de commande, obtenir une liste composée :
- de 11 nombres entiers, choisis aléatoirement et de façon équiprobable, parmi les entiers compris entre 0 (compris) et 5 (exclu) ;
- de 11 nombres décimaux, choisis aléatoirement et de façon équiprobable, dans [0 ; 5[.

Utiliser ensuite les fonctions sum et product pour faire respectivement les somme et produit des éléments d'une des listes précédentes.

Représentations graphiques, objets géométriques

Chaque commande graphique crée un objet graphique, qui est traduit en une image de réponse dans une fenêtre graphique. Saisir, par exemple, droite(y=3x-1). A droite de cette fenêtre, se trouve un pavé de boutons : explorer les commandes qui correspondent aux boutons (agrandissement, réduction, déplacement vers le bas, vers le haut…)

On peut effectuer les représentations graphiques les plus courantes grâce au menu Graphe, qui propose des boîtes de dialogue qui se chargent de créer une ligne de commande et de la valider pour afficher le graphique souhaité. En utilisant ce menu, représenter graphiquement la fonction qui à x associe $\frac{2x+1}{x^2+1}$.

On peut aussi créer une figure grâce au menu Geo : dans ce menu, Nouvelle figure (2d ou 3d), puis utiliser le menu Geo pour créer des objets géométriques.

Saisir les instructions suivantes et décrire ce qu'elles permettent d'obtenir :
- E:=(2*x+1)/(x^2+1)
- plot(E)
- plot(E,x=-2..2,color=red)
- couleur(vert)
- tangent(plot(E),0)
- plot([sin(x),x,x-x^3/6],x=-2..2,couleur=[rouge,bleu,vert])

Noter que les objets graphiques 2d sont aussi affichés dans une fenêtre appelée DispG (Display Graphics) que vous pouvez faire apparaître par le menu Cfg->Montrer->DispG ou

avec la commande DispG. La différence est que les graphiques successifs sont tracés individuellement dans chaque fenêtre de réponse, alors qu'ils sont superposés dans la fenêtre DispG. On peut effacer la fenêtre DispG par la commande ClrGraph.

Dans une autre fenêtre graphique 2d :

- A:=point(1,2)
- B:=point(i)
- droite(A,B)
- cercle(A,3)

Dans une fenêtre graphique 3d :

- plotfunc(x^2-y^2,[x,y])
- plotfunc(x+y^2,[x=-5..5,y=-2..2],xstep=0.5,ystep=0.1, affichage=vert+rempli)

On obtient ainsi une surface en dimension 3. Pour modifier le point de vue, on peut utiliser la souris en dehors du cube de visualisation ou les boutons du pavé situé à droite de la fenêtre graphique.

Poursuivre, dans la même fenêtre 3d :

- plan(z=x+y)
- plan(z=x+y,couleur=rempli+rouge)
- affichage(line_width_3)
- droite(x=y,z=y,couleur=bleu)
- A:=point(1,2,4)
- B:=point(2,-4,1)
- droite(A,B)