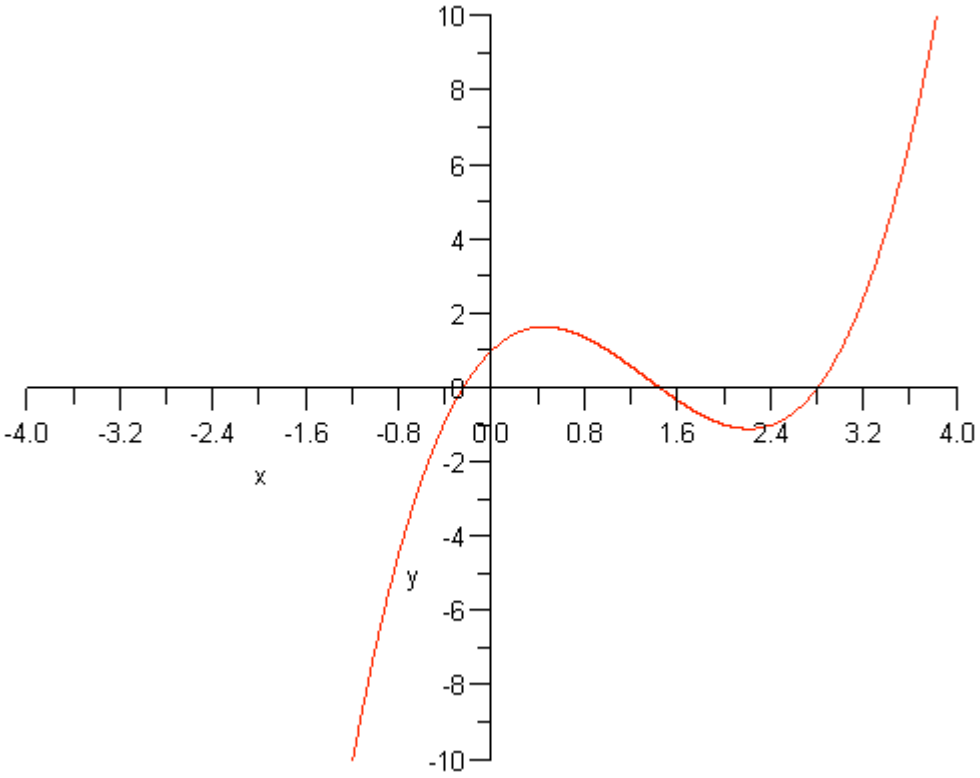
ÉTUDE DE FONCTIONS POLYNOMIALES AVEC LES MAPLETS DANS MAPLE 10



Guide du concepteur

Par Simon Laurent

TABLE DES MATIÈRES

PRÉFACES

Le but du programme était de rendre l'interface de Maple 10 moins froide, plus conviviale. Je voulais me servir de la puissance de Maple, mais changer son apparence un peu plate (entrer à la file des lignes de code).

Cela m'a été possible grâce aux Maplets (contraction de Maple et applet). Les Maplets sont des outils intégrés à Maple 10 qui permettent, justement, à l'utilisateur d'intégrer des interfaces graphiques beaucoup plus belles et conviviales. En demandant à Maple d'inclure les bibliothèques Maplets, un utilisateur averti pourrait écrire le code du Maplet à même la feuille de travail Maple. Comme je ne connaissais pas le langage, j'ai utilisé (et en ferai mention dans ce guide) l'assitant Maplet Builder, également inclus dans Maple 10. Je vous ferai part des limites de cet assistant et de la mise en forme (assez pauvre) de Maple en général.

STRUCTURE DU PROGRAMME

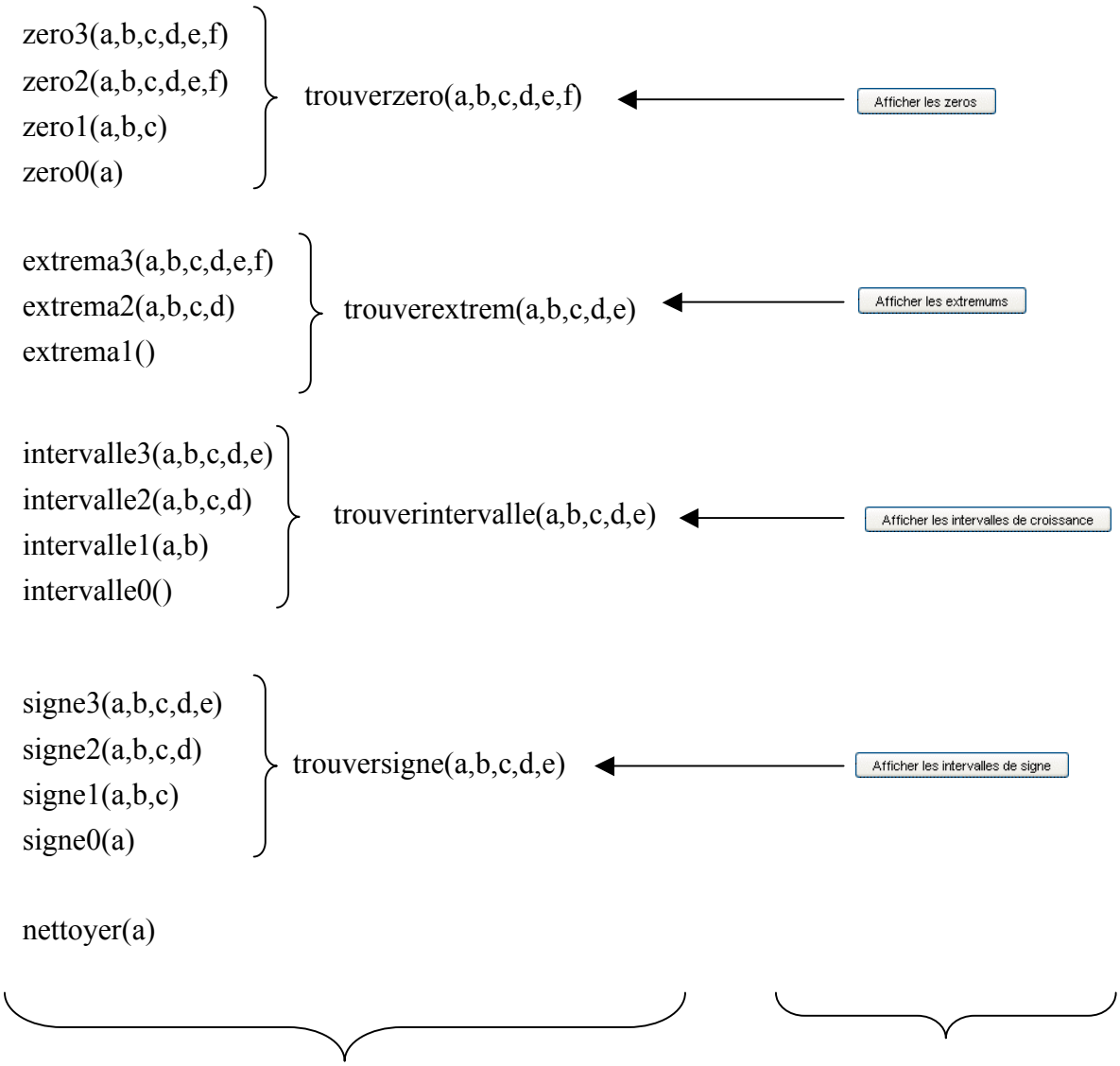
Depuis Maple 10, ouvrir le fichier **Fonction.mw**. C'est la feuille de travail Maple 10 qui contient tout le code du programme. Tel que mentionné dans le guide de l'utilisateur, en analysant ce code, on peut le séparer en deux parties : une première partie dans laquelle sont définies une multitude de procédures et une deuxième partie constituée d'un bloc d'exécution assez volumineux contenant le code du Maplet (notez que le bloc d'exécution commence avec la commande with(Maplets[Elements]);

Lorsqu'on exécute le Maplet, la fenêtre graphique apparait. L'utilisateur peut ainsi manipuler le programme, lequel appelle différentes procédures définies plus haut dans la feuille de travail. Un Maplet peut être autonome s'il n'a pas de calculs complexes à effectuer (par exemple, un Maplet qui fait seulement afficher un graphique ou une équation). Dans le cas présent, le Maplet doit résoudre des équations complexes, afficher des graphique, etc.

Par ailleurs, le programme permet à l'utilisateur de trouver quatre informations : les zéros, les extremums, les intervalles de croissance et les intervalles de signe. De plus, pour chacune des informations, le programme supporte les degrés 0, 1, 2 et 3. Il y a donc ici environ 16 procédures différentes définies dans la feuille de travail qui permettent de trouver l'ensemble des informations. Or, le programme doit savoir qu'elle procédure appeler selon les paramètre définies par l'utilisateur. Il y a donc 4 autres procédures qui permettent de choisir qu'elle procédure appeler.

Le schéma de la page suivante illustre bien la situation…

Structure du programme :

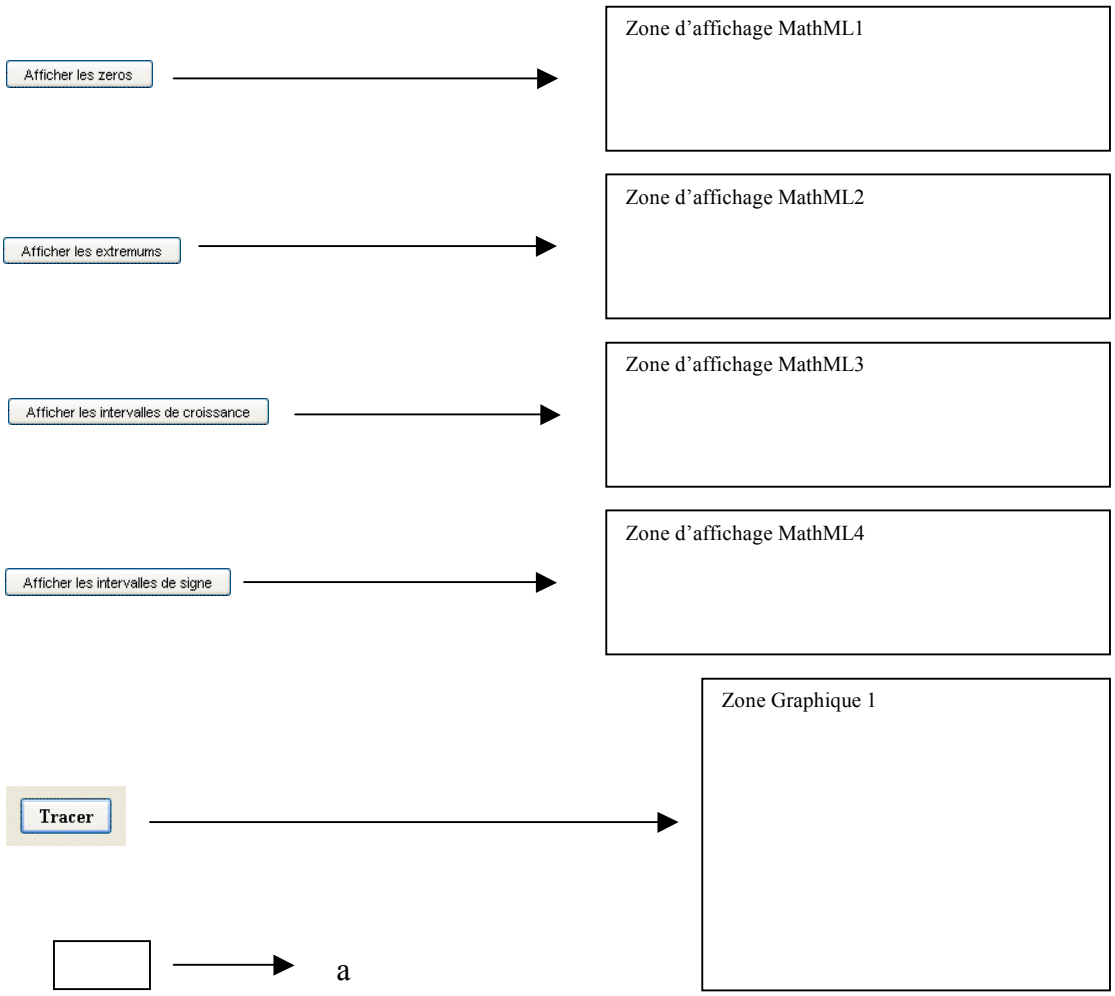


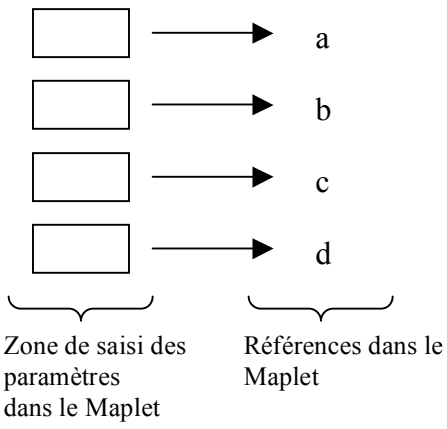
Procédures définies dans la feuille de travail Maple 10

Boutons définis dans le Maplet qui appellent ces procédures.

Une partie de la difficulté vient du fait que chaque bouton a *une* cible précise et *une* commande précise. On ne peut pas assigner à un bouton plusieurs cibles (par exemple une ou plusieurs zones d'affichage MathML et une ou plusieurs zones graphique) ou plusieurs commandes (par exemple appeler trouverzero() et trouversigne() avec un seul bouton).

C'est essentiellement pourquoi j'ai inséré dans le programme un bouton pour tracer et quatre autres pour faire afficher les quatre informations désirées. Enfin, notons que ces informations sont affichées dans quatre zones d'affichage MathML.





Note : Les éléments qui peuvent êtres affichés dans la zone MathML sont malheureusement assez restreint. Outre les nombres, je n'ai pu faire afficher que des listes. En étudiant les procédures de la feuille de travail, on s'aperçoit que la mise en forme est faite grâce à des listes, ce qui explique l'affichage très peu extraordinaire observé dans le Maplet. Je crois que c'est parce que le MathML conserve les propriétés (dans Maple) des objets affichés et c'est pourquoi il ne supporte que quelques objets. On aurait souhaité qu'il puisse afficher au moins les matrices, ce qui aurait rendu la mise en forme plus souple.

LES PROCÉDURES DANS MAPLE 10

Le langage de Maple est semblable à de nombreux autres langages de programmation. Et comme dans tous ces autres langages de programmation, il possède des particularités au niveau de sa syntaxe et de son vocabulaire. Si on veut créer une fonction dans la feuille de travail que l'on veut appeler plus tard dans cette même feuille, on définie dans Maple des *procédures*.

Je conseille très fortement à un quelconque nouvel utilisateur de parcourir les fiches du Maple Help concernant la programmation (accessible depuis le menu « Help »).

Regardons ensemble la procédure nettoyer(a); que l'on retrouve dans la feuille de travail. J'utilise la procédure nettoyer pour éliminer les composantes réelle ou imaginaire d'un nombre lorsque celles-ci sont très petites ou nulle. J'ordonne à Maple de calculer les zéros avec 100 décimales de précision. À partir de ce nombre, j'estime qu'un nombre qui vaut moins de $1\cdot10^{-11}$ vaut en réalité 0 et qu'il devrait être traité de telle sorte.

Ainsi, lorsque je fais afficher un zéro, j'obtiens quelque chose du genre $0-2\cdot i$ au lieu de $0.345532457823487967862348768639578897987\cdot10^{-93}-2\cdot i$ ce qui est, vous le conviendrez, beaucoup plus clair.

```
> nettoyer:=proc(a)
```
On déclare le nom de la procédure. Entre parenthèses, les paramètres que l'utilisateur doit spécifier

```
local var;
```
On déclare le nom des variables locales. Dans le cas de variables globales, donc pour toute la feuille de travail, on aurait `global var;`

```
var:=Re(a)+Im(a)*I;
```
On définit la variable. Notez la syntaxe `:=`

```
if abs(Re(var))<0.00000000001
then var:=Im(var)*I; end if;
if abs(Im(var))<0.00000000001
then var:=Re(var); end if;
```
Ici, on réalise deux tests l'un après l'autre afin de savoir si, d'une part, la partie réelle est plus petite que $1\cdot10^{-11}$ *(en valeur absolue). Dans ce cas, on élimine la composante réelle. On refait le même test avec la partie imaginaire du nombre.*

```
return var;
```
On demande à Maple de retourner la (nouvelle) valeur de `var`

```
end proc;
```
On termine notre procédure.

Je mets un bémol à la commande `return var;`. Dans ce cas-ci, la procédure sera appeler à l'intérieur de la feuille Maple, par exemple dans une autre procédure. On veut utiliser le nombre que nettoyer nous retourne.

Si la procédure est appelé à être utiliser dans le Maplet, c'est cette commande (l'*output*) qui doit être modifié. Par exemple, si on voulait appeler nettoyer du Maplet et faire afficher le nombre obtenue dans une zone d'affichage MathML, il faudrait alors écrire : `MathML[Export](var);`. Les commandes étant nombreuses et spécifiques à chaque action, je recommande à l'utilisateur de parcourir les différentes sections du Maple Help qui seront très utiles.

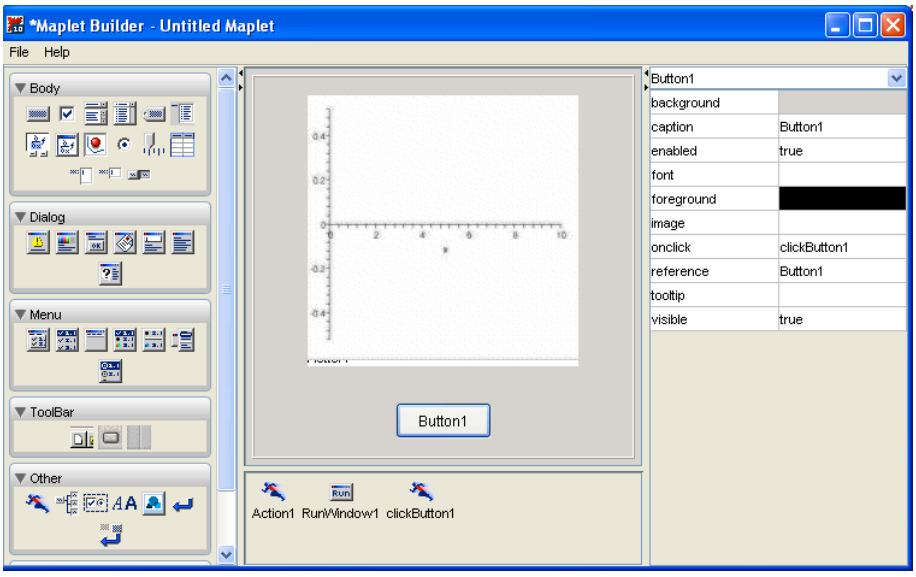
LE MAPLET BUILDER

Dans Maple, ouvrir le menu « Tools », puis sélectionner « Assistant ►» et enfin « Maplet Builder ».

Le Maplet Builder est un assistant qui permet de construire des Maplets sans avoir à connaitre de façon approfondie le code. De façon très simple, l'utilisateur se retrouve avec un Maplet « vierge » et différents outils à gauche. Il n'a qu'à faire glisser les éléments qu'il désire ajouter dans le Maplet au centre. Ces éléments sont automatiquement créés et mis en forme. Parmi ces éléments, on retrouve, entre autres, des boutons, des étiquettes, des glissières, des zones d'affichages graphique, des champs de saisi, des menus, des zones d'affichage MathML, des boutons poussoir, des cases à cocher, etc. Les éléments qu'on y retrouve sont comparables aux outils que l'on retrouve dans la « boîte à outils Contrôles » du programme Microsoft Excel.

Dans la colonne de droite, l'utilisateur peut aller choisir un à un chaque élément de son Maplet, selon la référence qu'il a spécifié pour chaque élément. Lorsqu'il le fait, les options de l'élément lui sont proposées, et l'utilisateur peut les changer à sa guise. On peut retrouver, par exemple, la couleur de fond, la police, le nom de référence, la taille de la zone ou l'action associée à l'élément dans ces options.

Enfin, la zone en bas du Maplet Builder, sous le Maple « vierge », renferme de façon visuelle toute les actions ou propriétés définies dans le programme. Si l'utilisateur ajoute un bouton « Quitter », l'action « Shutdown » apparaîtra dans cette zone. Si l'utilisateur définit une nouvelle police/mise en forme, elle apparaîtra également dans cette zone. Néanmoins, si l'utilisateur programme un Maplet qui tient dans une fenêtre (comme je l'ai fait), cette zone lui sera utile, mais pas autant qu'elle l'aurait été pour un utilisateur qui désire faire apparaître de nombreuses fenêtres ou menus.



Dans le menu fichier, on retrouve la commande « Run ». Cette commande fonctionne seulement pour les Maplets simples. Si votre Maplet est moindrement complexe (par exemple avec une zone d'affichage MathML), vous devez sauvegarder le Maplet, quitter le Maplet Builder et revenir à votre feuille de travail Maple. De là, vous ouvrez le Maplet (l'extension est, justement, ***.maplet) comme vous ouvririez une feuille de travail normale. Le code du Maplet (que le Maple Builder a gentiment écrit pour vous) apparaît dans la feuille. De là, vous n'avez qu'à exécuter le bloc d'exécution comme pour n'importe quelle autre commande de la feuille de travail. Le Maplet apparaîtra quelques secondes plus tard.

Enfin, lorsque vous travailler dans le Maplet Builder, sauvegardez souvent sous plusieurs nom de fichiers (selon les différentes versions). Puisque le Maplet Builder n'a pas de « Annulez » ou « Ctrl+Z ». Une erreur est donc, à priori, irréparable. Le débogage du code est aussi désagréable puisqu'il faut aller étudier le code bogué dans la feuille de travail pour déceler le problème et le régler. C'est un peu gênant quand on sait que la fonction première du Maplet Builder est, justement, de ne pas obliger les utilisateurs à s'empêtrer dans un code de programmation assez lourd.

Avant de continuer, je recommande à quiconque voudrait programmer des Maplets de compléter les tutoriels du Maplet Builder, que l'on peut retrouver depuis le menu « Help ».

COMMENT MODIFIER LE PROGRAMME

Un des buts de ce guide est de rendre le programme parfaitement ouvert aux modifications que chacun voudrait pouvoir y apporter. Comment quelqu'un doit-il se prendre pour modifier le programme ?

Voici les étapes pour créer un Maplet fonctionnel :

1. Créer la ou les procédures utilisées dans la feuille de travail (s'il y a lieu). S'assurer par la suite qu'elles sont parfaitement fonctionnelles et sans bogue.
2. Ouvrir le Maplet Builder et créer le Maplet, en associant déjà les procédures décrites dans la feuille aux différents boutons ou menus. Sauvegarder le maplet.
3. Ouvrir le fichier ***.maplet depuis Maple (et non depuis le Maplet Builder). Le code apparait dans une feuille de travail. Copier tout le code (tout le bloc d'exécution commençant par `with(Maplet[Elements]);`) à la fin de la feuille de travail où sont décrites la ou les différentes procédures.
4. Exécuter tous les blocs d'exécution de cette même feuille.

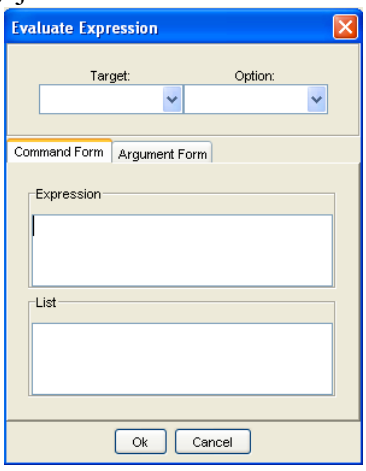
Si le Maplet est déjà créé et que vous voulez le modifier :

1. Ouvrir la feuille de travail comprenant les procédures et le bloc d'exécution du Maplet (par exemple, **Fonction.mw**).
2. Ajouter les nouvelles procédures directement sur la feuille. S'assurer que ces nouvelles procédures sont parfaitement fonctionnelles et sans bogue.
3. Copier tout le bloc d'exécution du Maplet (en bas de la feuille de travail) et **couper** (Ctrl+X).
4. Dans une nouvelle feuille de travail **coller** le bloc d'exécution de Maplet.
5. Du menu « File », sélectionner « Export as… »
6. Choisir Maplet dans le menu déroulant (celui dont l'extension est ***.maplet).
7. Dans les menus « Tools » et « Assistants », ouvrir le Maplet Builder.
8. Du Maplet Builder, ouvrir le fichier ***.maplet exporté (« File » et « Open »)
9. Modifier le Maplet à sa guise et reprendre à **l'étape 2** de la liste précédente (p. 8)

Exemple :
Nous voulons ajouter une fenêtre qui ferait afficher la valeur initiale. Dans la feuille de travail, il suffirait d'ajouter une procédure bien simple. Comme nous étudions des fonctions polynomiales, la valeur initiale est toujours la constante.

```
> valeurinit:=proc(a)
MathML[Export](a);
end proc;
```

En suivant les étapes décrites ci-haut (p. 9), j'ouvre le Maplet dans le Maplet Builder. J'ajoute une nouvelle zone d'affichage MathML. La référence de cette zone est MathMLViewer5. J'ajoute également un bouton. La référence de ce bouton est Button7. Dans le menu déroulant de la colonne de droite, je choisis Button7 et modifie les options désirées (par exemple « Caption »). À la ligne « Onclick », je choisis dans le menu déroulant « Evaluation ». Une fenêtre apparait : je m'assure que la cible est bien MathMLViewer5 et que « Option : » est bien à « Value ». Dans la zone « Expression » j'écris la commande à exécuter, ici : valeurinit(d). Je NE mets PAS de point-virgule. Il est à noter que le *d*, ici, représente la référence d'une zone de saisi correspondant à mon paramètre d. J'ai nommé chaque zone de saisi des paramètres a, b, c et d de la fonction polynomiale respectivement par a, b, c et d, afin de faciliter la construction du Maplet (voir p.5). Clic Ok.



En reprenant à **l'étape 2** de la page 8, nous avons modifié avec succès le Maplet en ajoutant une zone qui donne à l'utilisateur, en un clic de bouton, une nouvelle information, à savoir la valeur initiale.

Si une quelconque étape décrite ci-dessus n'est pas claire, je vous recommande de compléter les tutoriels fournis dans le menu « Help » du Maplet Builder.