Projet MuPAD - Documentation

Killian POULET-ALLIGAND Nicolas LAUNAY Sébastien LEMAIRE–HEY
Etienne Caillosse

Avril 2013

Chapitre 1

Organisation

Dans la racine du projet se trouvent cette aide, ainsi que le diaporama de la soutenance. On y trouvera aussi le document de l'Université de Nantes qui a servi de support au projet mais qui n'est plus disponible en ligne.

Le code MuPAD à proprement parler est situé dans le dossier src/. Dans celui-ci, on trouvera un dossier procedures/ qui contient les procédures du programme. À la racine du dossier src/ se trouvent les fichiers contenant les applications scientifiques du projet. Un fichier suffixé d'un D indique l'utilisation de la méthode des différences finies, tandis qu'un T indique l'usage de la méthode de Tir.

Concernant le dossier src/procedures/, les fichiers sont en double. L'un au format .mn, accessible directement avec MuPAD. L'autre, au format .mu, contient les sources des procédures au format texte, afin de permettre l'import de celles-ci dans les différents autres fichiers de MuPAD.

Chapitre 2

Code

2.1 Généralités

À chaque fin de fichier de procédure, on trouvera une ligne comme celle-ci :

```
write(Text, NOTEBOOKPATH."Fichier.mu", Procedure)
```

Cette ligne permet de créer le fichier Fichier.mu, qui contient le code de la procédure "Procedure", au format texte brut. Ces fichiers peuvent alors être importé via la ligne suivante (que l'on trouve en début des fichiers qui se servent des procédures) :

```
read(NOTEBOOKPATH.pathname("procedures")."Fichier.mu");
```

La procédure "Procedure" est alors utilisable directement depuis le fichier où la ligne a été exécuté.

2.2 Procédures

Dans cette section sont indiqués les paramètres, les spécificités de chaque procédure du projet.

2.2.1 RungeKutta

Cette procédure permet la résolution d'équations différentielles à l'ordre 2 par la méthode de Runge-Kutta à l'ordre 4.

Le code de cette procédure est disponible dans le fichier src/procedures/RK.mn

Elle s'appelle ainsi :

```
RungeKutta(A, B, C, D, x0, x1, y0, z0, nbre_points, variable)
```

Les paramètres sont les suivants :
- A, B, C et D sont les paramètres de l'équation différentielles : $A.y''(x) + B.y'(x) + C.y(x) + D = 0$;
- $x0$ et $x1$ sont les bornes de l'intervalle de travail. $x0$ doit bien sûr être inférieur à $x1$;
- $y0$ est l'image de $x0$ par la fonction recherchée ;
- $z0$ est l'image de $x0$ par la DÉRIVÉE de la fonction recherchée ;
- $nbre_points$ indique le nombre de points que la procédure doit calculer ;
- variable est la variable dont dépendent les paramètres A, B, C et D. Ainsi, la syntaxe est simplifié dans l'entrée des paramètres, le $fp :: unapply$ étant exécuté par la procédure.

Cette procédure retourne la liste des points de la fonction solution, qui peuvent directement être tracés avec $plot :: PointList2d()$

2.2.2 Tir

Cette procédure permet la résolution des équations différentielles à l'ordre 2 avec conditions aux limites.

Le code de cette procédure est disponible dans le fichier src/procedures/Tir.mn.

Elle s'appelle ainsi :

```
Tir(A, B, C, D, x0, x1, y0, y1, nbre_points_sortie, precision, variable)
```

Les paramètres sont les suivants :
- A, B, C et D sont les paramètres de l'équation différentielles : $A.y''(x) + B.y'(x) + C.y(x) + D = 0$;
- $x0$ et $x1$ sont les bornes de l'intervalle de travail. $x0$ doit bien sûr être inférieur à $x1$;
- $y0$ est l'image de $x0$ par la fonction recherchée ;
- $y1$ est l'image de $x1$ par la fonction recherchée ;
- $nbre_points$ indique le nombre de points que la procédure devra calculer ;
- *variable* est la variable dont dépendent les paramètres A, B, C et D.

Cette procédure retourne la dérivée de la fonction en $x0$, ainsi que la liste des points de la droite solution, traçable directement avec $plot :: PointList2d()$. Attention, il n'est pas possible de résoudre des équations d'ordre 1 : si A est nul, il se produira une erreur (division par 0).

2.2.3 Differenciation

Cette procédure permet la résolution des équations différentielles à l'ordre 2 avec conditions aux limites.

Le code de cette procédure est disponible dans le fichier src/procedures/Differenciation.mn.

Elle s'appelle ainsi :

```
Differenciation(A, B, C, D, x0, x1, y0, y1, r, variable)
```

Les paramètres sont les suivants :
- A, B, C et D sont les paramètres de l'équation différentielles : $A.y''(x) + B.y'(x) + C.y(x) + D = 0$;
- $x0$ et $x1$ sont les bornes de l'intervalle de travail. $x0$ doit bien sûr être inférieur à $x1$;
- $y0$ est l'image de $x0$ par la fonction recherchée ;
- $y1$ est l'image de $x1$ par la fonction recherchée ;
- r indique le nombre de points que la procédure devra calculer ;
- *variable* est la variable dont dépendent les paramètres A, B, C et D.

Cette procédure retourne la liste des points de la droite solution, traçable directement avec $plot :: PointList2d()$. La résolution d'équations d'ordre 1 est à priori possible, mais peux retourner des résultats étranges.

2.2.4 Cramer

Cette procédure permet la résolution des systèmes d'équations différentielles du type : $A.X = D$, où A est une matrice carrée de taille n, X une matrice colonne de taille n, contenant les inconnues, et D une autre matrice colonne de taille n. La procédure utilise la méthode de Cramer.

Le code de cette procédure est disponible dans le fichier src/procedures/Differenciation.mn.

Elle s'appelle ainsi :

```
Cramer := proc(A, D)
```

Les paramètres sont les suivants :
- A est un objet du type Matrix. Il s'agit d'une matrice carrée de taille n, dont les coefficients sont connus.
- D est un objet du type Matrix. Il s'agit d'une matrice colonne de taille n, dont les coefficients sont connus.

Cette procédure retourne la liste des coefficients de la matrice solution.