—————— Outils Informatiques — Mias 1 — TP 6 ——————

Résolution de problèmes sous MuPAD

Ce dernière séance de travaux pratiques porte entièrement sur le logiciel MuPAD. Elle est composée (comme les séances précédentes) de deux parties portant respectivement sur l'étude des polynômes et la résolution de systèmes linéaires d'une part, et la résolution de systèmes d'équations différentielles d'autre part.

Première partie : polynômes et systèmes linéaires

Cette première partie commence donc par quelques manipulations sur les polynômes, et se termine par un exercice sur la résolution de systèmes linéaires.

Exercice 1-1 : manipulation des polynômes

1. Dans un nouveau carnet de notes, après quelques lignes de texte indiquant à quoi il correspond, définissez les polynomes P_1 et P_2 correspondant respectivement à $X^4 + X^2 + 1$ et $X^3 - 1$.
2. Faites manuellement la division de P_1 par P_2. Vérifiez votre résultat en utilisant la commande `divide(P1, P2);` sous MuPAD. Utilisez ensuite les commandes `divide(P1, P2, Quo);` et `divide(P1, P2, Rem);` pour affecter à deux variables `Q` et `R` respectivement le quotient et le reste de cette division.
3. Pour obtenir un affichage plus lisible, tapez par exemple la commande `print("Quotient : ", Q , "; Reste : ", R);`.
4. Vérifiez l'exactitude des résultats obtenus, en recalculant la somme entre le produit `P2 * Q` et `R`. Utilisez pour cela la commande `expand(P2 * Q + R);`, la fonction `expand` effectuant le développement de l'expression passée en paramètre.
5. Factorisez les deux polynômes P_1 et P_2 à l'aide de la commande `factor`. Quel est le plus grand diviseur commun (PGCD) de ces deux polynômes ?
6. Vérifiez votre réponse, avec la commande `gcd`. Chercher aussi le PPCM en utilisant la commande `lcm`. Afficher les résultats obtenus de façon lisible !

Exercice 1-2 : caractérisation d'un polynôme

Un polynôme peut être caractérisé l'ensemble de ses coefficients. On peut aussi vouloir connaître son degré, le nombre de ses termes, son terme constant, etc. Ces informations sont obtenues sous MuPAD en utilisant respectivement les commandes `coeff`, `degree`, `nterms` et `ground`. Essayez ces commandes avec le polynôme $P = 7X^4 + 5X^3 + X$.

On peut aussi chercher, pour un polynôme, le terme de plus grande puissance, son coefficient ou bien le monôme associé. Avec MuPAD, il suffit pour cela d'utiliser respectivement les commandes `lterm`, `lcoeff` et `lmonomial`. Pour obtenir ces informations pour n'importe quel monôme d'un polynôme, il suffit d'utiliser les commandes `nthterm`, `nthcoeff` et `nthmonomial`, en passant comme paramètre après le polynôme le numéro d'ordre (suivant les puissances décroissantes) du terme en question. Ainsi, la commande `nthterm(P, 3);` donne, pour le polynôme P précédemment défini, le terme numéro 3 (c'est-à-dire X). Enfin, la commande `tcoeff` donne le coefficient de la plus petite puissance, et la commande `coeff(P, X, 3);` donne le coefficient du terme X^3. Essayer toutes ces commandes sur le polynôme P précédemment défini.

Exercice 1-3 : division suivant les puissances croissantes

Soient les deux polynômes $P_1 = 8X^4 + 6X^3 + 2X^2$ et $P_2 = 4X^3 - 2X$. On va tout d'abord les normaliser en les divisant par leur monôme de plus faible puissance, et appeler Q_1 et Q_2 les quotients respectifs de ces divisions.

La position du monôme de plus faible puissance est donnée par le nombre de termes du polynôme. Ainsi, pour P1, utilisez la commande `n1:=nterms(P1):`. Ensuite, on divise P_1 par `m1` où `m1:= nthmonomial(P1, n1):`. Le quotient Q_1 est alors obtenu par la commande `Q1:= divide(P1, m1, Quo);`. Calculez de la même façon Q_2 et affichez proprement les deux quotients trouvés.

La division selon les puissances croissantes de la variable de Q_1 par Q_2 permet d'obtenir le développement limité au voisinage de 0 de la fraction rationnelle Q_1/Q_2. Écrivez le petit programme suivant, en tapant simultanément `Shift` et `Entrée` pour changer de ligne :

```
DP := 0:
for k from 1 to 3 do
  n1 := nterms(Q1):   m  := nthmonomial(Q1, n1):
  DP := DP + m:       Q1 := expand(Q1 - m * Q2):
  print("k = ", k, ", DP = ", DP, ", Q1 = ", Q1);
end_for;
```

Le développement limité de P_1/P_2 est alors obtenu par la commande `DP:= expand(DP * m1 / m2);`. Vérifiez le résultat fourni avec la commande `series(P1 / P2, X);`.

Exercice 1-4 : recherche des racines d'un polynôme

On veut factoriser le polynôme $P = 7X^4 - 64X^3 + 114X^2 - 64X + 7$. Pour cela, on peut utiliser la commande `factor` de la librairie standard, déjà vue au niveau du premier exercice, et la commande `sqrfree` de la bibliothèque `polylib`, dont l'appel s'écrit `polylib::sqrfree`. Essayez ces deux commandes et expliquez la différence qui existe entre les deux résultats obtenus.

On veut maintenant chercher d'une autre façon les racines de ce polynôme. Pour cela, on va tester quelques commandes définies dans les bibliothèques `polylib` et `numeric`.

Dans la bibliothèque `polylib`, la commande `realroots` permet de localiser toutes les racines réelles, en les situant sur des intervalles dont la largeur est fixée par l'utilisateur. Par exemple, la commande `polylib::realroots(P, 0.1);` situe, dans des intervalles de largeur 0,1 maximum, les racines réelles de P. Testez cette commande. Pour avoir le résultat en virgule flottante, utilisez la commande `float(%);`.

Dans la bibliothèque `numeric` existent aussi plusieurs commandes permettant de localiser les racines du polynôme P. La commande `numeric::realroots(P, X = 0..8, 0.1);` a le même effet que la commande précédente (on remarquera qu'on doit préciser l'intervalle global sur lequel les racines seront recherchées). La commande `polyroots` calcule toutes les racines (réelles ou complexes) du polynôme en question. Essayez ces deux commandes.

On peut aussi utiliser le solveur de la librairie standard ou celui de la bibliothèque `numeric`. Testez ces deux solveurs en utilisant respectivement les commandes `solve(P, X)` et `numeric::solve(P)`, puis refaites ce même exercice pour le polynôme `expand((X - sqrt(2))^3 * (X - 1.414) * (X - 1.415))`. Que constatez-vous ?

Exercice 1-5 : résolution de systèmes linéaires

Soit le système linéaire à deux équations et deux inconnues suivant :

$$\left\{ \begin{array}{rcl} 10^{-10}x + y & = & 1 \\ x + y & = & 2 \end{array} \right.$$

Ce système peut être résolu par la commande `linsolve` de la bibliothèque `numeric`, si on lui donne comme paramètre la liste des équations et celle des inconnues (les inconnues doivent donc être écrites sous la forme `[x, y]`). Pour avoir une solution exacte, il suffit d'ajouter le paramètre `Symbolic`.

On s'intéresse maintenant aux systèmes matriciels, c'est-à-dire les systèmes pouvant s'écrire sous la forme $AX = B$, X et B étant deux vecteurs de taille n et A une matrice $n \times n$. Nous allons considérer un exemple pour $n = 4$, où A et B valent respectivement :

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}, \quad B = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$$

La commande `matlinsolve` de la bibliothèque `linalg`, avec pour paramètres la matrice A du système et le vecteur résultat B, permet la résolution de ces systèmes matriciels linéaires. On rappelle qu'une matrice se définit grace au constructeur `matrix`, avec comme paramètre la liste de ses lignes, ces lignes étant elle-même des listes.

Résolvez le système défini plus haut, puis essayez de modifier légèrement le vecteur résultat (prenez par exemple le vecteur $(32.1{,}22.9{,}33.1{,}30.9)$) et comparez les solutions. Pour mieux comprendre ce phénomène, définissez le vecteur B sous la forme $(32 + t{,}23 - t{,}33 + t{,}31 - t)$, et cherchez la solution du système associé.

Résolvez le système correspondant à la matrice A et au vecteur résultat $(30{+}15/7{,}20{+}\exp(1){,}30{+}\pi{,}30{+}\sqrt{2})$. Utilisez la commande `float(%);` pour obtenir la forme décimale des vecteurs résultat et solution. Quelle est la solution lorsque le vecteur résultat vaut $(32.143{,}22.718{,}33.142{,}31.414)$? Qu'en pensez-vous ?

Seconde partie : équations différentielles

Exercice 2-1 : mise en évidence d'une équation raide

Dans cet exercice, on se propose de résoudre l'équation différentielle $y'(t) = 5y(t) - 7/3e^{-2t}$, avec comme condition initiale $y(0) = 1/3$.

Commençons par retrouver la solution exacte de cette équation, à l'aide des commandes suivantes :

```
EqDiff := y'(t) = 5 * y(t) - 7/3 * exp(-2 * t):
CInit  := y(0) = 1/3:
PbDiff := ode({EqDiff, CInit}, y(t)):
sol    := solve(PbDiff);
```

On peut aussi récupérer cette même solution sous forme d'une fonction, en tapant la commande `fsol := fp::unapply(op(sol));`.

La résolution de cette équation avec des méthodes numériques mène à des solutions approchées, dont on va montrer dans notre case la dégradation progressive, même lorsqu'on utilise de très bonnes méthodes numériques. Pour cela, commencer par définir l'équation différentielle sous la forme d'une procédure : `eqDiff := proc(t, Y) begin [5 * Y[1] - 7 / 3 * exp(-2 * t)] end_proc`.

Pour résoudre au mieux notre problème, nous allons utiliser de quatre manières différentes la fonction `odesolve` du module `numeric`, qui permet de résoudre des équations différentielles ordinaires, et définir de cette façon quatre fonctions. Tout d'abord, utilisez la à l'ordre 4 et avec un pas fixe de 0.01 (`sol4 := t ->numeric::odesolve(0..t, eqDiff, [1/3], RK4, Stepsize = 0.01)`). Ensuite, ajustez automatiquement le pas avec la commande `sol4b := t ->numeric::odesolve(0..t, eqDiff, [1/3], RK4)`. On peut aussi utiliser l'ordre 8 (ordre par défaut avec le pas fixe) et un pas de 0.01 (`sol8 := t ->numeric::odesolve(0..t, eqDiff, [1/3], Stepsize = 0.01)`). Enfin, utilisez cette commande à l'ordre 8 et avec un pas ajusté automatiquement (`sol8b := t ->numeric::odesolve(0..t, eqDiff, [1/3])`).

On va maintenant calculer les valeurs de ces différentes fonctions pour t valant 1, 2, 3, 4 et 5. On calculera ensuite l'erreur pour chacune de ces valeurs. Si on tape `v := 1:`, la solution exacte en ce point est obtenue par la commande `s := fsol(v);`, les valeurs des quatre solutions approchées sont données par `sol4(v); sol4b(v); sol8(v); sol8b(v);`, et l'erreur pour ces valeurs sont :

```
er4  := 100 * (1 - sol4(v)[1] / s):
er4b := 100 * (1 - sol4b(v)[1] / s):
er8  := 100 * (1 - sol8(v)[1] / s):
er8b := 100 * (1 - sol8b(v)[1] / s):
```

Pour afficher proprement ces résultats, utilisez la commande `print(Unquoted, "Erreurs relatives en %", float(er4), float(er4b), float(er8), float(er8b));`. Refaire ce même travail pour t valant 2, 3, 4 et 5 (vous pouvez utiliser une boucle `for`).

Pour comprendre cette dégradation de la qualité des solutions, recherchez la solution exacte de notre même équation différentielle, mais avec la condition initiale `CInitap := y(0)= 1/3 + eps`. On obtient `1/3` $\mathtt{e}^{-2t}$ `+ eps` $\mathtt{e}^{5t}$, le terme `eps` $\mathtt{e}^{5t}$ s'ajoutant à chaque fois à la solution approchée. Regarder son évolution pour `eps` valant 10^{-10}, en tapant `float(eps * exp(5 * k)) $ k = 1..5;`.

Exercice 2-2 : étude d'un accéléromètre

Soit l'accéléromètre présenté par la figure 1, où f est le frottement visqueux de l'amortisseur, r est la raideur du ressort ($r = 1$ kg/s^2), m est la masse de l'objet centré sur O ($m = 1$ kg), x est la position de cet objet. Cet objet subit une accélération horizontale γ ($\gamma = 1$ m/s^2). On note k la constante $f/(2m)$.

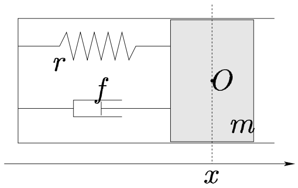

FIG. 1 – *Schéma d'un accéleromètre.*

L'équation du mouvement $mx''(t) = \gamma m - rx(t) - fx'(t)$ s'écrit donc sous la forme simplifiée $x''(t) = 1 - x(t) - 2kx'(t)$. Donner la solution exacte de cette équation pour les conditions initiales $x(0) = 0, x'(0) = 0$. Cette solution est elle définie pour $k = 1$?

On va résoudre cette équation pour trois valeurs de k, valant respectivement 1, 0.15 et 0.05. Calculez la solution de l'équation pour chacune de ces valeurs, et rangez la dans un vecteur `sol` à 3 éléments, en utilisant une boucle `for`.

Affichez ensuite les courbes des trois solutions sur un même graphique. Pour cela, définissez trois variables correspondant à l'appel de la commande `plot::Function2d` pour ces fonctions, sur l'intervalle $[0,40]$, et tapez la commande `plot(Title = "Accéléromètre", Courbes[k] $ k = 1..3);`. Changez les couleurs des courbes obtenues.

Pour obtenir aussi le diagramme de phase de ces fonctions, on doit définir la dérivée des fonctions (`g[i]:= diff(sol[i])`, pour `i = 1..3`) et la courbe dont les coordonnées sont données par la valeur de la solution et la valeur de sa dérivée (`G[i] := plot::Curve2d([sol[i], g[i]], t = 0..40, Grid = [200])`). La visualisation des courbes est enfin obtenue par :

```
plot(Title ="Accéléromètre - Diagramme de Phase",
     Scaling = Constrained,  G[k] \$ k = 1..3);
```