TABLEAUX DANS UN VECTEUR UNIQUE

par Charles Hubert

Introduction

Dans certains calculs comme l'intégration numérique d'un système différentiel ou la résolution d'un système d'équations numériques, on rencontre la difficulté suivante.

Supposons par exemple qu'il s'agit d'équations différentielles ; les équations numériques s'examinent de manière analogue. Quand on programme les équations on préfère désigner les variables d'état par des symboles significatifs comme

```
   position  vitesse  temps
```

en étant bavard.

Quand on intègre numériquement les équations il faut au contraire que toutes les variables soient rassemblées dans un seul vecteur, nécessairement désigné par un seul symbole.

On présente ici un jeu de fonctions qui concilient ces deux points de vue en rassemblant les variables dans un tableau unique.

Gestion du vecteur de tableaux

On préparera le travail de l'exemple cité dans l'introduction en exécutant l'instruction

```
   1 VarEtat 'position 3 vitesse 3 temps'
```

ou

```
   1 VarEtat 'position' 3 'vitesse' 3 'temps'
```

si on veut que "position" et "vitesse" soient des vecteurs de dimension 3 et que "temps" soit un scalaire. Cette fonction "VarEtat" crée trois variables :

```
   ⍳position ⍳vitesse ⍳temps   ⍝ ⎕io=0
     0 1 2     3 4 5    6
```

et six fonctions :

```
    ∇  δ_x←position                  ∇  Dposition δ_x
[1]    δ_x←etat[;⍳position]     [1]    Detat[;⍳position]←δ_x
    ∇                                ∇
    ∇  δ_x←vitesse                   ∇  Dvitesse δ_x
[1]    δ_x←etat[;⍳vitesse]      [1]    Detat[;⍳vitesse]←δ_x
    ∇                                ∇
    ∇  δ_x←temps                     ∇  Dtemps δ_x
[1]    δ_x←etat[;⍳temps]        [1]    Detat[;⍳temps]←δ_x
    ∇                                ∇
```

Le vecteur d'état complet est censé être dans la variable "etat" sous forme de matrice, on verra plus loin pourquoi, et sa dérivée dans la matrice "Detat" de même structure :

```
  ⍴etat ⋄ ⍴Detat
1 7
1 7
```

Après exécution de VarEtat, ces deux variables sont pleines de zéros. Ensuite la vitesse, par exemple, sera dans les colonnes "⍳vitesse" (3 4 5) de "etat".

La fonction "vitesse", sans argument, donnera la valeur de la vitesse en sélectionnant ces colonnes. Mais on ne pourra pas lui affecter une valeur ; c'est normal car physiquement on ne peut changer une variable d'état qu'en agissant sur sa dérivée. La fonction "Dvitesse" affectera la valeur de son argument à la dérivée de la vitesse en affectant les mêmes colonnes de "Detat". Mais on ne pourra pas lui demander sa valeur ; cela n'est pas gênant car on peut écrire une instruction comme

```
  Dvitesse dvitesse ← ...
```

si on veut réutiliser cette valeur ultérieurement.

Par convention, les colonnes correspondant à une variable d'état s'obtiennent par le préfixe "⍳" et sa dérivée par le préfixe "D".

En donnant aux trois variables d'état précédentes des noms plus courts analogues à ceux qu'on peut trouver dans un traité de mécanique, l'intégration se prépare par l'instruction

```
      1 VarEtat 'r 3 v 3 t'
```

et les équations de la chute d'un corps dans l'air se programment

```
    ∇  Detat←Chute etat
[1]    Detat←etat
[2]    Dt 1
[3]    Dr v
[4]    Dv 0 0 ¯9.81 -[1]v×[0]k×(+/v*2)*0.5
    ∇
```

La ligne `[1]` crée le vecteur d'état dérivé de même structure que le vecteur d'état. Les autres lignes traduisent les équations

$[2]\ t' = 1 \qquad [3]\ \vec{r}' = \vec{v} \qquad [4]\ \vec{v}' = \vec{g} - \vec{v}\ k\ \|\vec{v}\|$

L'initialisation se fera par

```
      1 VarEtat 'r 3 v 3 t'
```

Si on veut une structure des variables plus compliquée, on peut initialiser par une instruction comme

```
     1 VarEtat 'r' n 3 'v' n 3 't'
```

ou comme

```
     1 VarEtat 'xa' (1+2×n) 'yb' (n*2) 't'
```

Les variables "etat" et "Detat" sont des matrices parce que les algorithmes de résolution peuvent traiter plusieurs situations en même temps. Ces différentes situations sont dans autant de lignes différentes de ces matrices ; c'est le choix déterminé par l'argument gauche "1" de VarEtat. En fin d'intégration, les états successifs sont aussi dans les lignes succcessives de la matrice "etat".

On peut préférer que les valeurs de chaque variable de ces différentes situations soient dans une seule case, alors "etat" et "Detat" seront des vecteurs dont la profondeur "2" sera réservée à cet usage ; ce choix sera fait par la valeur "0" de l'argument gauche de VarEtat. Les équations s'écriront alors

```
    ∇  Detat←Chute etat
[1]    Detat←etat
[2]    Dt 1
[3]    Dr v
[4]    Dv 0 0 ¯9.81 -v×k×(+/v*2)*0.5
    ∇
```

et l'initialisation se fera par

```
      0 VarEtat 'r 3 v 3 t'
```

La résolution d'un système numérique se fera en initialisant par une fonction "VarIncon" et les inconnues seront rassemblées dans une variable "incon".

Tous les symboles locaux de ces fonctions sont préfixés par "δ_" afin de réduire le risque de masquer des variables globales homonymes. Le caractère "δ" est dessiné "Δsouligné" dans les polices de caractères APL traditionnelles.

Les fonctions

On peut les trouver dans le fichier APL*PLUS

EQUAT.SF

La composante 1 est une table des matières, les autres composantes sont les `⎕vr` des fonctions de ce fichier.

La fonction VarEtat est listée dans l'article

INTEGRATION DES SYSTEMES DIFFERENTIELS PAR RUNGE-KUTTA

La fonction VarIncon est listée dans l'article

RESOLUTION DES EQUATIONS NUMERIQUES PAR NEWTON

Ces fonctions appellent "IndVec" et "ValVec".

```
     ∇  δ_z←IndVec δ_x;δ_d;δ_n;δ_p
[1]     δ_x←(δ_x≠' ')⊂δ_x←,' ',⍕δ_x
[2]     δ_p←⊂(↑δ_x)~'.' ◇ δ_x←1↓δ_x
[3]     δ_x←(+\~(↑¨δ_x)∊'0123456789')⊂δ_x
[4]     δ_n←1↓¨⍎¨⍕¨'0',¨1↓¨δ_x ◇ δ_x←↑¨δ_x
[5]     δ_z←δ_nρ¨2 ◇ (∊δ_z)←⍳δ_d←ρ∊δ_z
[6]     ⍎'(',(1↓∊' ',¨δ_p,¨δ_x),')←δ_z'
[7]     δ_z← δ_p δ_x δ_d
     ∇

     ∇  δ_z←δ_a ValVec δ_x;⎕io;δ_f;δ_i;δ_l;δ_n;δ_s;δ_t;δ_v
[1]     ⎕io←0
[2]     δ_a←,⍕δ_a ◇ δ_i←'←'∊δ_a ◇ δ_a←δ_a~'←'
[3]     (δ_f δ_v)←2↑(δ_a≠' ')⊂δ_a ◇ δ_v←';',δ_v,';'
[4]     (δ_s δ_t)←1↓¨(δ_v=';')⊂δ_v ◇ δ_v←δ_v~';'
[5]     ((0=↑¨0ρ¨∊δ_x)/∊δ_x)←' '
[6]     δ_x←(δ_x≠' ')⊂δ_x←,⍕δ_x ◇ δ_l←1↑δ_x ◇ δ_n←1↓δ_x
[7]     δ_z←(ρδ_n)ρ⊂δ_i⊃(' δ_x←@' ' δ_x←&[%]')(' @ δ_x' ' &[%]←δ_x')
[8]     (('@'=∊δ_z)/∊δ_z)←(⊂δ_f~'.'),¨δ_n
[9]     (('&'=∊δ_z)/∊δ_z)←⊂δ_v
[10]    (('%'=∊δ_z)/∊δ_z)←(⊂δ_s),¨(⊂δ_l~¨'.'),¨δ_n,¨⊂δ_t
[11]    δ_z←⎕fx¨⊃¨∊¨¨δ_z
     ∇
```