Variables Scilab/Matlab

Objet de base :

dans Scilab, l'objet de base est la matrice bidimensionnelle de nombre réels ou complexe. Le scalaire est un cas particulier (matrice de taille 1x1). Pas de problème pour créer des tableaux multidimensionnels à plus de deux indices.

Les tableaux en ligne sont couramment appelés liste ou vecteur ligne
les tableaux en colonne sont appelés vecteurs.
les tableaux bidimensionnels sont des matrices (attention pour les opérateurs) . Le stockage est réalisé en colonne.
Les variables et tableaux sont en ***déclaration implicite***

Indice :

les indices d'un tableau sont dans l'ordre ***ligne/colonne***.
l'indiçage permet de désigner un élément particulier : *Tab1(2,3)* ou un sous ensemble du tableau : *Tab1(4:7 , 1)*
indice pour le dernier élément : *Tab (5 : $, 1 : 3);* (*Tab (5 : end , 1 : 3)* en Matlab)
syntaxe particulière pour désigner une ligne ou une colonne entière *Tab1(: , 10)* désigne les éléments de la 10ème colonne. La forme (ligne, colonne ou tableau) est conservée.

Éléments d'un tableau :

un tableau est constitué d'éléments auxquels une valeur est affectée ex *Tab1(3,4) = 10*;
remarquer que la dimension des tableaux est "extensible" automatiquement.
Affection de plusieurs valeurs structurées en tableau *Tab1 = [1 2 var1 ; 2 4 5+3*%i]* ;
Tableau vide *Tab1 = [ ]*
Affectation de valeurs incrémentales *Temps = [0 : .2 : 10]*; (ligne)

Tableaux particuliers:

tableau formé de "1 " *Tab1 = ones (10, 20)* ; *Tab2 = ones (Temps)*(en SCI seulement)
tableau formé de "0" *Tab3 = zeros (4 , 5);*
matrice diagonale *Mat1 = eye(5) ; Mat2 = eye(3 , 4) ;*

Concaténation de tableau :

Tab3 = [Tab1 Tab2] (les dimensions doivent être compatibles)
Tab3D (: , : ,1) = Tab1 ; Tab3D (: , : ,2) = Tab2 // tableau à 3 dimensions (mêmes dimensions pour *Tab1* et *Tab2*)

Dimension d'un tableau:

les tableaux étant de type dynamique; à tout moment, il est possible de connaître la taille d'un vecteur ou d'un tableau par la fonction *length* . La fonction *size* retourne la liste des valeurs des dimensions.
[nb_ligne , nb_colonne] = size(mon_tableau) ;

Transposition :

Tab2 = Tab1 ' ; permutation des lignes et des colonnes

Opérateurs :

pas de problème avec + et - sauf dimensions à respecter (addition scalaire autorisée)
attention: * représente la multiplication matricielle !!!
de même *A^2* représente *A*A* au sens matriciel (A matrice carrée obligatoirement)

Opérateurs point à point :

permettent de faire les calculs directs sur tous les éléments d'un ou plusieurs tableaux sans boucle. [Très puissant]

exemples : élévation au carré de tous les éléments d'un tableau *A.*A*

division des éléments de A par ceux de B (même dim) *A./B*

Opérateurs logiques:

Les valeurs logiques sont 0 et 1 (notée F et T à l'affichage)

Il est possible de multiplier un nombre réel ou complexe par un logique (multiplication par 0 ou 1).

Les tests usuels sont étendus au type matrice *A==%pi* retourne une matrice de 0/1 de même dimension que A (les valeurs sont considérées comme identiques si l'écart est inférieur à l'epsilon de calcul %eps défini pour l'application)

B = A . (A>10)* retourne les valeurs supérieurs à 10 de *A* , les autres valeurs étant à 0 .

Fonctions usuelles :

le logiciel dispose de très nombreuses fonctions intégrées

Syntaxe spécifique aux fonctions retournant plusieurs paramètres:

[par1 , par2] = my_function(param1, param2 ..)

Les fonctions sont généralement étendues au type matriciel lorsque ceci a un sens.

Tabsin = sin (Tab1) calcule directement le sinus de chaque terme du tableau.

l'ajout de nouvelles fonctions est facile.

Opérateurs de traitement de données:

fonctions *sum, min, max , mean, variance , stdev* appliquées directement à un tableau.

maxi , mini permettent de rtrouver l'indice du maxi : *[rmax, cmax] = maxi(A)*

Saisie d'une valeur :

x = input ("valeur du seuil') retourne la valeur saisie dans la variable x

Structure de test:

structure if :

```
if test_logique1 then
    a = 3 ;
elseif test_logique2
    a = 10 ;
end
```

structure select :

```
select expr,
  case expr1 then instructions1,
  case expr2 then instructions2,
  ...
  [else instructions],
end
```

Structures de boucle :

```
for i = 1 : valeur_fin
    a(i) = ....   // calcul itératif
end
```

```
while condition do ..
    instructions
. [else
    instructions]
end ,
```

Ajout de fonctions

Une nouvelle fonction se crée avec le mot-clef *function* . La syntaxe est la suivante :

```
function [param1 , param2] = nom_fonction( param3 , param3)
    contenu de la fonction
endfunction
```

Graphiques Scilab/Matlab

plot :

fonction principale de tracé classique ; permet de tracer automatiquement une fonction (liste de valeurs) selon les valeurs abscisse

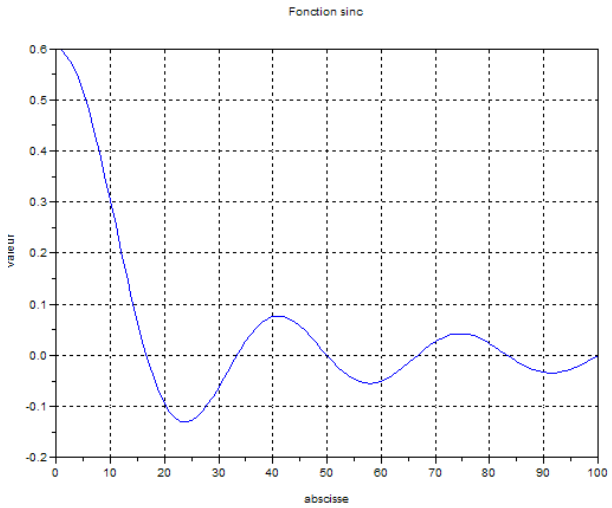
plot(liste_abscisse , valeurs_fonction)

exemple:

*T = [0.1 : .1 :10]; Ma_Fonction = sin(0.3*2*%pi*T)./(%pi*T) ;* //fonction de type *sinu/u*

plot (Ma_Fonction) // tracé en fonction de l'indice de liste des valeurs

plot (T , Ma_Fonction) //tracé avec abscisse



enrichissement graphique:

xgrid affiche la grille xy

xtitle permet de mettre un titre sur le graphique et les axes 2D et 3D

xtitle('Calcul Final' , 'Abcisse en mm' , 'Amplitude') ;

clf(n) efface le contenu de la fenêtre *n* .

scf(n) permet d'affecter la fenêtre *n* à un tracé (la fenêtre est créée si elle n'existe pas) à utiliser avant le *plot*)

xdel(n) détruit la n-ième fenêtre graphique ouverte

mise à l'échelle du graphique de la fenêtre courante :

a = gca() ; a.data_bounds=[min_x , min_y ; max_x, max_y];

paramètres de la fonction plot :

plot (abscisses , ordonnées , 'r-o') // couleur rouge avec interpolation et points

autres couleur : *b,y ,g* ... marqueur + *o* ...(voir LineSpec)

voir également *plot2d* qui offre des possibilités plus étendues .

Toolbox SIVP

lecture image :

im1=imread('nom_image.ext')
lecture du fichier image dénommé et transfert du contenu dans la structure image *im1*
supporte une grande variété de type de fichier image (bmp, png , ...), mais pas les images *indexées* (table de gris ou couleur sur 8 bits par exemple en format tif)
l'image est stockée sous forme de 3 plans R,V,B : *im1(: , : , 1)* désigne le plan Rouge de l'image *im1*

formats d'image

Les formats internes supportés par les fonctions de la bibliothèque sont :
- le format RVB [0:255 , 0:255 , 0:255]
- les images à niveau de gris G [0 : 255]
- les formats INT8, UINT8, INT16, UINT16 , INT32,
- le format binaire noir et blanc [0,1]
- les formats couleurs usuels NTSC , YCrCb , HSV

écriture image:

imwrite(im1, 'mon_image.png')
écriture de tout type d'image dans les formats usuels (bmp, jpg , png, tif...)

affichage image :

imshow(mon_image) la fonction est unique pour les images noir&blanc ou couleur ;
limitée à une seule fenêtre (décrit dans la notice),
les grandeurs doivent être cadrées entre [0 255]
ne fonctionne pas avec le type double (utiliser *uint8(mon_tableau)* pour faire la conversion)

écriture graphique:

fonction réduite à un simple rectangle, pas d'autre primitive
imr = rectangle(im1, [20, 30, 50, 100], [0 255 0]);
autre méthode : forcer la valeur des niveaux à 255 dans l'un des plans (RVB) pixel par pixel

Fonctions de traitement

La Toolbox comprend un certain nombre de fonctions propres au traitement d'image. Ces fonctions seront abordées lors de leur utilisation.