Cours 1: variables, types simples, flot d'exécution, conditionnelles

David Delahaye

David.Delahaye@lirmm.fr

Polytech' Montpellier

MAT3/MI3 2016-2017

Plan du cours

Trois parties

1. Généralités sur l'algorithmique
2. Objets de base : variables et types
3. Organisation d'un programme (séquences, conditionnelles)

Pourquoi programmer ?

Automatisation de tâches répétitives

- Analyse de signaux reçus par un sonar de sous-marin ;
- Calculs de trajectoire ;
- Test de résistance d'un matériau soumis à des vibrations aléatoires ;
- Etc.

Plusieurs autres raisons

- Un ingénieur doit donner une réponse vérifiable dans un délai court à un problème posé, incorporant souvent de nombreuses variables/valeurs ;
- Les données d'un problème peuvent varier dans le temps ;
- Il n'existe pas toujours de programme tout fait qui fait que l'on veut.

Algorithmique

Algorithme

Description systématique d'un procédé de calcul (d'une méthode) pour obtenir un résultat déterminé à partir de données initiales.
Algorithme = (pseudo-)code + données.

Problèmes

Un algorithme répond à un problème donné.
Deux types de problèmes :

- Problèmes de décision (résultat « oui » ou « non ») ;
- Problèmes de calcul (résultat entier, flottant, etc.).

Caractéristiques d'un algorithme

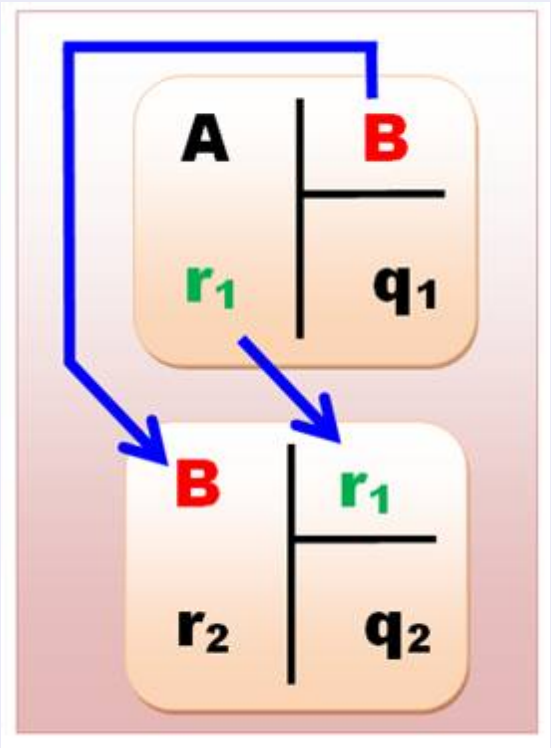
Un algorithme doit :

- S'arrêter (ne pas boucler) ;
- Fournir un résultat correct par rapport au problème donné.

Exemple d'algorithme

Algorithme d'Euclide

- Algorithme permettant de calculer le pgcd de deux entiers.
- Décrit dans le livre VII des Éléments d'Euclide (300 av. J.-C.).
- Il s'agit de divisions en cascade. Les résultats de l'une servent à poser la suivante :
  - Le diviseur B devient le dividende ;
  - Le reste r_1 devient le diviseur.
- Arrêt lorsque le reste de la division est nul.
- Le reste trouvé avant le reste nul est le pgcd.



Conception d'un programme

Algorithme vs. programme

- Programme = traduction d'un algorithme dans un langage ;
- La syntaxe doit être respectée (contrairement au pseudo-code) ;
- Le langage doit permettre d'écrire cet algorithme (Turing-complet).

Étapes

1. Identification et spécification du problème ;
2. Modélisation du problème (organigramme, UML) ;
3. Subdivision du problème en tâches simples et indépendantes ;
4. Planification des algorithmes en pseudo-code ;
5. Traduction en langage de programmation (Matlab) ;
6. Commenter ce code au fur et à mesure ;
7. Tester le programme par petits bouts et traquer les erreurs ;
8. Utiliser.

Outil utilisé : Matlab ?

Pourquoi utiliser Matlab

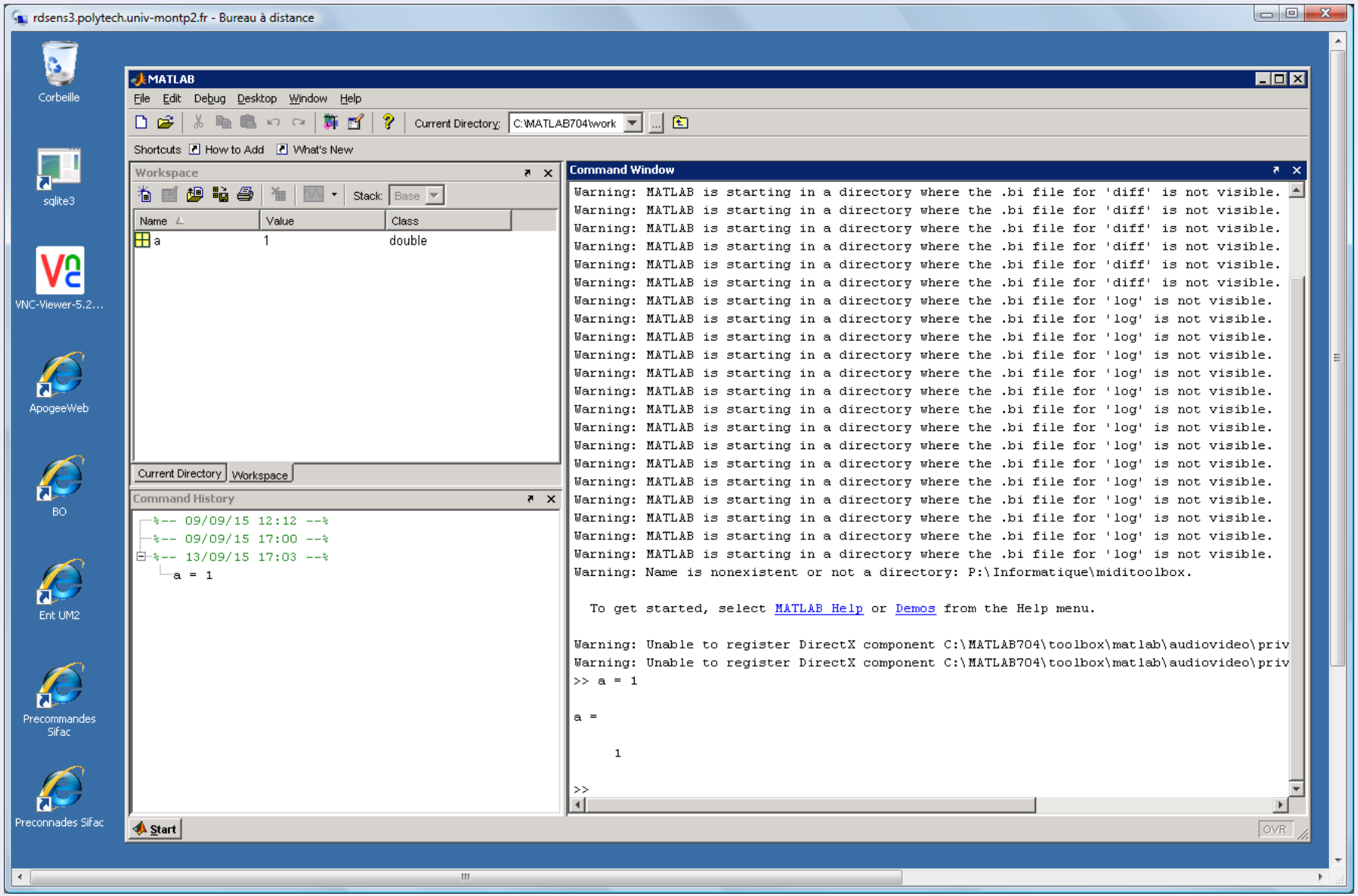
- C'est un environnement de programmation facile à utiliser :
  - Plusieurs fonctions prédéfinies pour analyser et représenter des données : on peut faire des choses élaborées avec très peu de code.
- C'est un outil adapté à la résolution de problèmes numériques rencontrés par les ingénieurs :
  - Il contient des outils d'optimisation et de résolution d'équations ;
  - Il contient des modules spécialisés d'analyse du signal et de l'image.
- Il est disponible dans vos salles de TP et en entreprise.
- Scilab est un équivalent gratuit proposé par Inria.

Interface de Matlab

TP

- Ouvrir Matlab ;
- Se familiariser avec les différentes fenêtres ;
- Taper sa première commande : `a = 1` (ne pas oublier <return>) ;
- Regarder l'évolution de l'espace de travail (« workspace ») ;
- Afficher la liste des commandes (`help`) ;
- Afficher l'aide pour la fonction valeur absolue (`abs`) ;
- Afficher la liste des variables (utiliser l'aide) ;
- Quitter Matlab.

Aperçu de l'interface



Objets de base : variables et types

Algorithme

- Programme = suite d'instructions manipulant des informations ;
- Informations stockées dans des variables ;
- Variables initialisées avec les données du programme ;
- Résultat = valeurs de l'ensemble des variables après exécution.

Variable

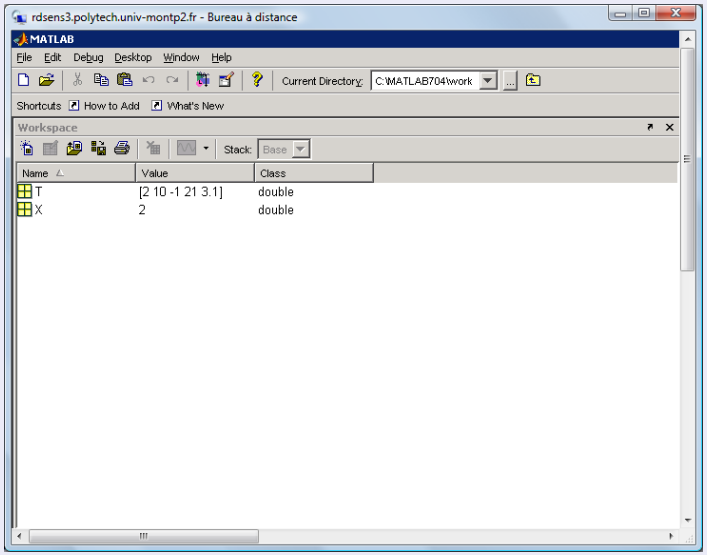
- Emplacement mémoire pour stocker des informations ;
- Identifiée par un nom ;
- Taille variable (suivant son type, c'est-à-dire ce que l'on stocke) ;
- Exemples : un entier = 1 case, un tableau = plusieurs cases.

Variables

Utilisation et définition

- Affectation (donne une valeur à une variable) :

```
>> X = 2
X =
     2
>> T = [2 10 -1 21 3.1]
T =
    2.0000   10.0000   -1.0000   21.0000    3.1000
```



Variables (suite)

Rôle du « ; »

- Permet d'enchaîner les instructions :

```
>> X = 2; Y = 3; % deux instructions et aucun affichage
>> who % liste synthétique des variables (juste les noms)
Your variables are:
X  Y
```

Élimination

- Commande `clear` :

```
>> clear X % élimine la variable X de la mémoire
>> X
??? Undefined function or variable 'X'.
>> clear % élimine toutes les variables de la mémoire
>> T
??? Undefined function or variable 'T'.
```

Types

Types des expressions

- Chaque variable a un type qui indique comment décoder l'information située dans l'emplacement et comment la manipuler au moyen d'opérations prédéfinies ;
- De manière générale, chaque expression (nous verrons plus en détail ce que c'est exactement plus tard) a un type ;
- Exemples de types : nombres entiers, nombres réels, booléens, chaînes de caractères, etc.

Opérations sur les types

- Nombres : +, -, *, / ;
- Chaînes de caractères : concaténation, sous-chaîne, etc. ;
- Etc.

Types (suite)

Connaître le type des variables

- Utilisation de la commande `whos` :

```
>> X = 2;
>> T = [2 10 -1 21 3.1];
>> whos
  Name        Size                    Bytes  Class
  T           1x5                        40  double array
  X           1x1                         8  double array
Grand total is 6 elements using 48 bytes
```

- Remarques :
  - Tout est matrice (une seule valeur sera considéré de taille 1×1) ;
  - Par défaut, Matlab stocke les entiers comme des nombres réels.
  - Pour vraiment stocker des entiers (ici sur 8 bits), faire :

```
>> X = int8(2);
```

Variables et types

TP

- Définir la variable `X` comme l'entier 2 sur 16 bits ;
- Définir la variable `Y` comme le flottant 3,5 en double précision ;
- Additionner `X` et `Y` (qu'observe-t-on ?) ;
- Refaire la même addition mais en obtenant un résultat flottant ;
- Afficher le plus grand entier représentable sur 32 bits en signé ;
- Même question sur 64 bits en non signé ;
- Définir la variable `Z` comme le nombre complexe $2 + 3i$.

Les booléens

Expressions logiques

- Type `logical` : c'est le type des expressions logiques ;
- Exemples : `X > Y`, `X ~= Y`, `(X < 10) & (isDiag(M))` ;
- Deux valeurs de ce type : *vrai* et *faux* ;
- Ces valeurs sont codées comme 1 et 0.

Opérations sur les booléens

- Ce sont des opérations qui retournent un booléen en résultat ;
- Opérations logiques : `&`, `|`, `~`, `xor`, `all`, `any` ;
- Opérations relationnelles : `==`, `~=`, `<`, `>`, `<=`, `>=` ;
- Opérations de test : `isinteger`, `ischar`, etc. ;
- Opération de conversion : `logical`.

Modélisation d'un problème avec les booléens

TP

- Test de conformité thermique d'un produit (contrôle de production) : On prend 3 échantillons (3 mesures de conductivité) du produit à évaluer, avec en provision un échantillon supplémentaire.
- On calcule :
  - Pour chaque échantillon i, B_i, à savoir la valeur absolue de l'écart relatif entre la conductivité thermique mesurée λ_{mes_i} et la conductivité thermique modélisée λ_{mod_i} :

$$B_i = \left| \frac{\lambda_{\mathrm{mes}_i} - \lambda_{\mathrm{mod}_i}}{\lambda_{\mathrm{mod}_i}} \right|$$

  - Le rapport indicateur S, qui est la moyenne arithmétique des écarts relatifs observés :

$$S = \frac{\sum_{i=1}^{3} B_i}{3}$$

Modélisation d'un problème avec les booléens

TP

- Le test de conformité thermique est déclaré satisfaisant lorsque :
  - L'indicateur $S \leq 0,03$ et aucune valeur des bornes $B_i > 0,06$;
  - Ou l'indicateur $S \leq 0,03$ et si une seule valeur des bornes $B_i > 0,06$ mais pour l'échantillon supplémentaire mesure $B_4 \leq 0,06$.
- Formuler le test de conformité comme une expression logique ;
- Tester cette expression dans plusieurs configurations.

Modélisation d'un problème avec les booléens

Solution

```
>> B1 = input('B1= ');
>> B2 = input('B2= ');
>> B3 = input('B3= ');
>> B4 = input('B4= ');
>> M = [B1 B2 B3];
>> S = sum(M) / 3;
>> T = [(B1 > 0.06) (B2 > 0.06) (B3 > 0.06)];
>> RES = ((S <= 0.03) & sum(T) == 0) |
         ((S <= 0.03) & sum(T) == 1 & B4 <= 0.06;
>> disp(RES);
```

Expressions

Instructions

- Instruction = ordre donné à la machine ;
- Programme = suite d'instructions ;
- Les instructions manipulent des informations appelées expressions.

Expressions

- Nombres : entiers (`42`, `-1`), flottants (`3.1416`, `-1.6180`) ;
- Booléens : `0`, `1` ;
- Caractères : `'a'`, `'$'` ;
- Chaînes de caractères : `'12345'`, `'Paul Hiteque'` ;
- Matrices : `[1 2 3 4 5]`, `[1 2 4; 5 6 7; 8 9 10]` ;
- Variables : `X`, `Y1` ;
- Opérateurs : `+`, `=` ;
- Fonctions : `intmax`, `all(T)`.

Organisation d'un programme

Programme

- Programme = suite d'instructions ;
- Le « ; » permet d'enchaîner les instructions ;
- Les instructions sont exécutées séquentiellement ;
- Certaines instructions peuvent casser cette séquence (itératives).

Instructions vues jusqu'à présent

- Affectation : `X = exp` ;
- Exécution de l'affectation :
  - L'expression `exp` est évaluée et donne la valeur v ;
  - La valeur v est ensuite stockée dans la variable `X`.
- Une affectation est donc utilisée pour mémoriser le résultat d'un calcul.

Conditionnelle

Nature, syntaxe, et exécution

- La conditionnelle est une instruction (et non une expression) ;
- Elle manipule cependant une expression booléenne (la condition) ;
- Syntaxe :

```
if cond
    inst1
else
    inst2
end
```

- Exécution :
  - La condition (expression) `cond` est évaluée et donne la valeur v ;
  - Si la valeur v est égale à `1` (vrai), alors l'instruction (ou le bloc d'instructions) `inst1` est exécuté ;
  - Sinon, si la valeur v est égale à `0` (faux), alors l'instruction (ou le bloc d'instructions) `inst2` est exécuté.

Conditionnelle

Exemple

- Tester si un nombre est positif :

```
>> X = input('X =  ');
   if (X >= 0)
       disp(strcat(num2str(X), ' est positif'))
   else
       disp(strcat(num2str(X), ' est négatif'))
   end
X = 2
2 est positif
```

Conditionnelle

« Dangling else »

- Pas d'alternative :

```
if cond
    inst
end
```

- Exécution :
  - La condition (expression) `cond` est évaluée et donne la valeur v ;
  - Si la valeur v est égale à 1 (vrai), alors l'instruction (ou le bloc d'instructions) `inst` est exécuté ;
  - Sinon, on ne fait rien.

Conditionnelle

Exemple

- Passer un nombre en valeur absolue :

```
>> X = input('X = ');
   if (X < 0)
       X = - X;
   end;
   disp(num2str(X));
X = -2
2
```

Conditionnelle

Conditionnelles en cascade

- Enchaînement de conditionnelles :

```
if cond1
    inst1
elseif cond2
    inst2
...
elseif condn
    instn
else
    inste
end
```

Conditionnelle

Conditionnelles en cascade

- Exécution :
  - La condition (expression) `cond1` est évaluée et donne la valeur v_1 ;
  - Si la valeur v_1 est égale à 1 (vrai), alors l'instruction (ou le bloc d'instructions) `inst1` est exécuté ;
  - Sinon on évalue la conditionnelle :
    `if cond2 inst2 ... elseif condn instn else inste end.`

Conditionnelle

Exemple

- Tester si un nombre est dans 3 intervalles :

```
>> X = input('X = ');
   if (X < -1)
       disp(strcat(num2str(X), ...
            ' est strictement inférieur à -1'))
   elseif (X <= 1)
       disp(strcat(num2str(X), ...
            ' est compris entre -1 et 1'))
   else
       disp(strcat(num2str(X), ...
            ' est strictement supérieur à 1'))
   end;
X = 0.5
0.5 est compris entre -1 et 1
```

Conditionnelle

TP

- Demander la saisie de 3 variables entières et afficher le maximum de ces 3 variables ;
- Demander la saisie d'une variable entière représentant une année et tester si cette année est bissextile. On sait qu'une année divisible par 4 est bissextile, sauf si elle est divisible par 100, cependant les années divisibles par 400 sont également bissextiles.
  - Donner une solution en utilisant les conditionnelles ;
  - Donner une solution en utilisant uniquement les expressions booléennes.

Conditionnelle

Solution

- Calcul du maximum :

```
>> X = input('X = ');
   Y = input('Y = ');
   Z = input('Z = ');
   MAX = X;
   if (Y > MAX)
       MAX = Y;
   end;
   if (Z > MAX)
       MAX = Z;
   end;
   disp(['Maximum = ' num2str(MAX)]);
```

Conditionnelle

Solution (suite)

- Calcul du maximum :

```
X = 3
Y = 7
Z = -1
Maximum = 7
```

Conditionnelle

Solution

- Année bissextile (conditionnelles) :

```
>> A = input('Année = ');
   if (mod(A, 4) == 0)
       if (mod(A, 100) ~= 0)
           RES = true;
       elseif (mod(A, 400) == 0)
           RES = true;
       else
           RES = false;
       end;
   else
       RES = false;
   end;
```

Conditionnelle

Solution (suite)

- Année bissextile (conditionnelles) :

```
>> ...
   if (RES)
       disp([num2str(A) ' est bissextile']);
   else
       disp([num2str(A) ' n''est pas bissextile']);
   end;
A = 2000
2000 est bissextile
```

Conditionnelle

Solution

- Année bissextile (expressions booléennes) :

```
>> A = input('Année = ');
   RES = ((mod(A, 4) == 0) & (mod(A, 100) ~= 0)) | ...
         (mod(A, 400) == 0);
   if (RES)
       disp([num2str(A) ' est bissextile']);
   else
       disp([num2str(A) ' n''est pas bissextile']);
   end;
A = 2000
2000 est bissextile
```