الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria

Ministry of Higher Education
And Scientific Research
TAHRI Mohammed Bechar University
Faculty of Technology
Department of Civil Engineering and Hydraulic

وزارة التعليم العالي
والبحث العلمي
جامعة طاهري محمد بشار
كلية التكنولوجيا
قسم الهندسة المدنية والري

Filière : Génie civil

Spécialité : Structures

Polycopié de la matière M111

Complément de programmation : Cours et Travaux pratiques

Réalisé par : Dr. ZAOUI Djelloul

Expertisé par :

- **Dr. BENKHEIRA Souad** — **MCB à UTMBechar**
- **Dr. BEN AYAD Slimane** — **MCA à UTMBechar**

Année Universitaire 2024-2025

Préface

Le présent polycopié intitulé «**Complément de programmation : Cours et Travaux pratiques**» ; est destiné aux étudiants de la première année Master en Génie Civil de spécialité «STRUCTURES» au premier semestre en matière «Complément de programmation» codifiée M111 ; ayant selon le canevas harmonisé un volume horaire de 45 heures, 4 pour crédits et un coefficient égal à 2.

Après une actualisation des connaissances préalables ; le support intègre des notions approfondies en matière de programmation. Les travaux pratiques inclus sont ciblés aux applications liées à la spécialité «STRUCTURES» de résistances des matériaux, de calcul des structures, de béton armé et de la charpente métallique.

Le polycopié est structuré en quatre chapitres :

- Le premier chapitre regroupe des rappels sur les bases de programmation par le langage MATLAB, les commandes de base, les opérations arithmétiques et les variables ;
- Le deuxième chapitre est réservé à la programmation modulaire par le langage MATLAB ; l'utilisation des fichiers SCRIPT et les fichiers USER-FUNCTION et à la manipulation des modes d'entrée / sortie ainsi que aux programmes et sous-programmes ;
- La programmation structurée par le langage MATLAB fait l'objet du troisième chapitre ainsi que les opérateurs relationnels et logiques ;
- Des exemples d'application liés à la spécialité «STRUCTURES» ; de résistances des matériaux, de calcul des structures, de béton armé et de la charpente métallique sont abordés en détails dans le dernier chapitre.

Pour éclaircir les étapes de traitements ; les rappels des calculs théoriques des travaux pratiques sont insérés en annexes.

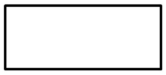
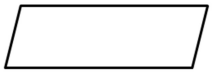
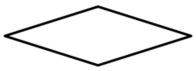
Table des matières

Liste des figures

Liste des tableaux

Liste des mise en forme graphiques et textuelles

Mise en forme graphiques

Emploi	Forme	Désignation
Organigramme	Ellipse 	Début et Fin
	Rectangle 	Déclaration des données
	Parallélépipède 	Actions
	Losange 	Testes
Schémas		Programme Principal
		Sous-Programme
		Fonction - Utilisateur

Mise en forme de texte

Gras sur fond gris : Pour les instructions des codes MATLAB ; exemple **>> 1+2**

Chapitre I :	Rappels sur les bases de programmation par le langage MATLAB

I.1. Introduction

Des rappels sur les éléments de base du langage MATLAB seront abordés dans ce chapitre ; à savoir : le lancement du programme, les opérations arithmétiques, les variables, les commandes de base ainsi d'autres.

I.2. Lancement

Pour lancer le programme, l'utilisateur doit ; cliquer sur l'icone correspondant placé sur le bureau ou taper le mot **MATLAB** dans la fenêtre de commandes. Après l'apparition du logo, le symbole >> clignote dans l'espace de travail afin d'introduire une ou des instructions.

Les données des instructions introduites sont groupées dans la fenêtre « ***Données*** » et leur historique est listé dans la fenêtre « ***Historique*** ». (**Figure I.1)**

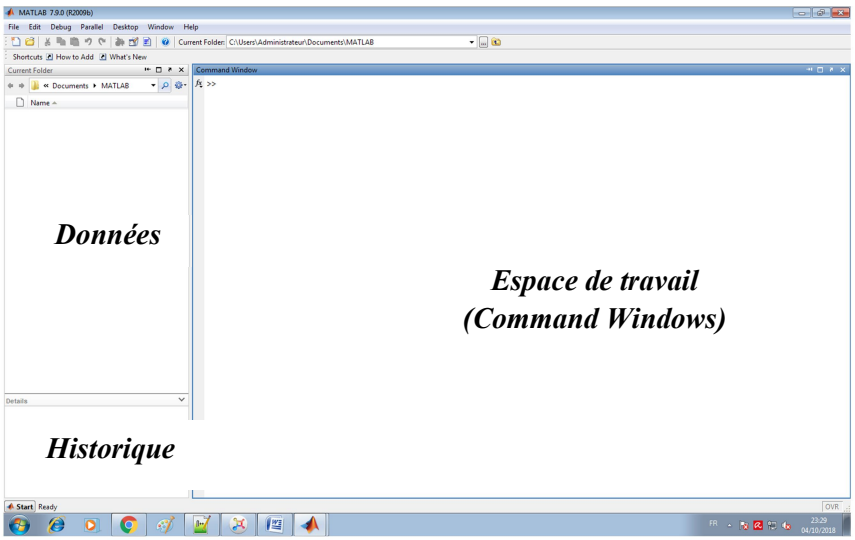


Figure I.1 : L'environnement du programme MATLAB.

I.3. Opérations arithmétiques

Dans l'espace de travail, les opérations arithmétiques entre les scalaires sont introduites à l'aide des symboles suivants : **+** pour l'opération d'addition, **-** pour la soustraction, **/** pour la division, ***** pour la multiplication et **^** pour l'élévation à la puissance. Le résultat est affecté à la variable **ans** (answer). Par exemple :

```
>> 1+2
ans =
   3
```

```
>> 1-2
ans =
   -1
```

```
>> 1*2
ans =
   2
```

```
>> 1/2
ans =
   0.5000
```

```
>> 2^3
ans =
   8
```

I.4. Variables

Pour affecter une valeur numérique à une variable on tapent directement son expression suivie du caractère **=** , la classe minuscule et majuscule de la variable est importante dans MATLAB car la variable **a** est différente de la variable **A** :

```
>> a = 1.25
a =
   1.2500
```

```
>> b = 3
b =
   3
```

```
>> c = a*b
c =
   3.7500
```

MATLAB prédéfinit les constantes suivantes **pi** : 3.1416 et **eps** : 2.2204e-016 ; l'utilisateur est tenu de les employer comme des constantes prédéfinis et non pas dans les expressions de ces propres variables.

Pour changer le format d'affichage des valeurs numériques ; l'instruction **format long** affiche 14 chiffres après la virgule et l'instruction **format short** affiche 4 chiffres après la virgule (affichage utilisé par défaut) :

```
>> format long
>> pi
ans =
  3.141592653589793
```

```
>> format short
>> pi
ans =
   3.1415
```

Parmi les fonctions incorporées dans MATLAB les plus utilisées : Exponentielle, Logarithmique, Trigonométrique, Hyperbolique et d'autres ; sont citées ci-après :

exp(x)	Exponentielle de **x**	**sin(x)**	Sinus de **x**
log(x)	Logarithme népérien de **x**	**cos(x)**	Cosinus de **x**
log10(x)	Logarithme en base **10** de **x**	**tan(x)**	Tangente de **x**
x^n	Puissance **n** de **x**	**asin(x)**	Sinus inverse de **x**
sqrt(x)	Racine carrée de **x**	**sinh(x)**	Sinus hyperbolique de **x**
abs(x)	Valeur absolue de **x**	**asinh(x)**	Sinus hyperbolique inverse de **x**
sign(x)	**-1** si **x<0**, **0** si **x=0** et **1** si **x>0**	**atanh(x)**	Tangente hyperbolique inverse de **x**

I.5. Commandes de base

Les commandes de base incorporées dans MATLAB sont :

who	Pour afficher les noms des variables introduites ;
whos	Pour afficher les informations des variables introduites ;
clc	Pour effacer l'écran de la fenêtre de commande, tout en gardant l'ensemble des variables ;
clear all	Pour supprimer toutes les variables introduites ;
close all	Pour fermer toutes les fenêtres graphiques créées.

I.6. Vecteurs

L'utilisateur peut définir un vecteur **Vx** en donnant la liste de ses éléments :

```
>> Vx = [0.5 1.2 -3.75 5.82 -0.375]
Vx =
   0.5000    1.2000   -3.7500    5.8200   -0.3750
```

Il a ainsi défini un **(1,5)** vecteur, c.à.d. un vecteur ligne contenant **5** éléments.

Par contre **(5,1)** est un vecteur colonne :

```
>> Vy = [0.5; 1.2; -3.75; 5.82; -0.375]
Vy =
   0.5000
   1.2000
  -3.7500
   5.8200
  -0.3750
```

On peut aussi définir un vecteur sous forme d'une suite arithmétique : par exemple de **2** à **4** avec un pas de **0.5** :

```
>> Vz = 2:0.5:4
Vz =
   2.0000    2.5000    3.0000    3.5000    4.0000
```

L'opération **transpose(Vz)** ou **Vz'** permet d'obtenir la transposée du vecteur **Vz** (en colonne pour un vecteur ligne et l'inverse pour un vecteur colonne) :

```
>> Vz'
ans =
   2.0000
   2.5000
   3.0000
   3.5000
   4.0000
```

Le produit du vecteur **Vz** par le scalaire **w = -2.5** est calculé par l'instruction **Vz*w (**la même syntaxe pour l'addition **Vz+w**, la soustraction **Vz-w** et la division **Vz/w**) :

```
>> w = -2.5
w =
  -2.5000
>> Vq = w*Vz
Vq =
  -5.0000   -6.2500   -7.5000   -8.7500  -10.0000
```

Les opérateurs d'addition **+**, de soustraction **-** entre deux ou plusieurs vecteurs nécessitent qu'ils soient du même type (ligne ou colonne) et de la même taille. Leurs syntaxes sont identiques aux celles des opérations entre les scalaires :

```
>>  Vs = Vx+Vz
Vs =
   2.5000    3.7000   -0.7500    9.3200    3.6250
```

Par contre les opérateurs de multiplication *****, de division **/**, et d'élévation à la puissance **^** qui sont définies par la syntaxe terme à terme ; le produit des deux vecteurs **x** et **z** est calculé par l'instruction :

```
>>  Vm = Vx.*Vz
Vm =
   1.0000    3.0000  -11.2500   20.3700   -1.5000
```

Si on veut extraire la valeur du **nième** élément du vecteur **Vm**, on utilisera l'indexation **Vm(nième)**:

```
>> Vm(3)
ans =
 -11.2500
```

Il existe aussi des commandes qui sont propre aux vecteurs ; pour le vecteur **Vx** on peut tirer :

sum(Vx)	La somme des éléments.	**mean(Vx)**	La moyenne des éléments.
prod(Vx)	Le produit des éléments.	**sort(Vx)**	Éléments arrangés par ordre croissant.
max(Vx)	L'élément le plus grand.	**fliplr(Vx)**	Éléments arrangés par ordre inverse.
min(Vx)	L'élément le plus petit.	**length(Vx)**	Le nombre des éléments.

I.7. Matrices

Une matrice est créée de façon similaire à un vecteur avec la commande **[]**. On définit la matrice **MA** :

$$MA = \begin{bmatrix} 1 & 2 & 5 \\ 4 & 3 & 6 \end{bmatrix}$$

```
>> MA=[1  2   5 ; 4  3   6]
MA =
   1    2    5
   4    3    6
```

La matrice **MA** est composée de **l** lignes et **c** colonnes. Si on souhaite connaitre la valeur de **l** et **c**, on utilise la commande **size(MA)** :

```
>> [l c]=size(MA)
l =
    2
c =
    3
```

Les éléments de la matrice **MA** peuvent être indexés sous la forme suivante :

MA(2,3)	Scalaire positionné à ligne **2** et la colonne **3**.
MA(:,3)	Tous les éléments de la colonne **3** sous forme d'un vecteur.
MA(2,:)	Tous les éléments de la ligne **2** sous forme d'un vecteur.

```
>> MA(2,3)
ans =
    6
```

```
>> MA(:,3)
ans =
    5
    6
```

```
>> MA(2,:)
ans =
    4    3    6
```

Les matrices particulières souvent employées sont :

La matrice zéro : on peut créer une matrice (**l** × **c**) de **0** en utilisant l'instruction **zeros(l,c)**. Par exemple :

```
>> MA1= zeros (2,3)
```

Crée la matrice de zéro a deux lignes et trois colonnes :

```
MA1 =
    0    0    0
    0    0    0
```

La matrice de un : on peut créer une matrice (**l** × **c**) de **1** en utilisant l'instruction **ones(l,c)**. Par exemple :

```
>> MA2=ones (2,3)
```

Crée la matrice de un a deux lignes et trois colonnes :

```
MA2 =
    1    1    1
    1    1    1
```

Les opérations matricielles traitées par MATLAB sont illustrées dans les parties suivantes :

A. Opérations matricielles : addition et soustraction

Les opérations d'addition et de soustraction sont possibles uniquement sur des matrices de taille identique. Ce sont des opérations termes à termes, similaires aux opérations entre des scalaires et des vecteurs. Par exemple le traitement de l'expression matricielle (I.1) est :

$$MA = \begin{bmatrix} 1 & 5 \\ -3 & 2 \end{bmatrix} \text{ et } MB = \begin{bmatrix} 4 & 2 \\ 0 & -4 \end{bmatrix};$$
$$MP = MA + MB = \begin{bmatrix} 1+4 & 5+2 \\ -3+0 & 2-4 \end{bmatrix} = \begin{bmatrix} 5 & 7 \\ -3 & -2 \end{bmatrix} \; ; \quad (I.1)$$
$$MM = MA - MB = \begin{bmatrix} 1-4 & 5-2 \\ -3-0 & 2+4 \end{bmatrix} = \begin{bmatrix} -3 & 3 \\ -3 & 6 \end{bmatrix}$$

Sous MATLAB, la syntaxe a les mêmes notations que pour les scalaires et les vecteurs :

```
>> MA = [1 5 ; -3 2]
MA =
   1   5
  -3   2
```

```
>> MB = [4 2 ; 0 -4]
MB =
   4   2
   0  -4
```

```
>> MP = MA+MB
MP =
   5   7
  -3  -2
```

```
>> MM = MA-MB
MM =
  -3   3
  -3   6
```

B. Opérations matricielles : produit

Il existe deux types de multiplication *terme à terme* ou celle dite *matricielle* :

- La multiplication terme à terme de l'expression matricielle (I.2) est notée de cette façon : **MA.*MB**

$$MA.*MB = \begin{bmatrix} 1 & 5 \\ -3 & 2 \end{bmatrix}.* \begin{bmatrix} 4 & 2 \\ 0 & -4 \end{bmatrix} = \begin{bmatrix} (1 \times 4) & (5 \times 2) \\ (-3 \times 0) & (2 \times -4) \end{bmatrix} = \begin{bmatrix} 4 & 10 \\ 0 & -8 \end{bmatrix} \quad (I.2)$$

```
>> MA = [1 5 ; -3 2]
MA =
   1   5
  -3   2
```

```
>> MB = [4 2 ; 0 -4]
MB =
   4   2
   0  -4
```

```
>> MPR1 = MA.*MB
MPR1 =
   4  10
   0  -8
```

- Le produit matriciel ; c.à.d. le produit non-commutatif on utilise simplement la syntaxe **MA*MB** :

$$MA * MB = \begin{bmatrix} 1 & 5 \\ -3 & 2 \end{bmatrix} * \begin{bmatrix} 4 & 2 \\ 0 & -4 \end{bmatrix}$$
$$= \begin{bmatrix} (1 \times 4) + (5 \times 0) & (1 \times 2) + (5 \times -4) \\ (-3 \times 4) + (2 \times 0) & (-3 \times 2) + (2 \times -4) \end{bmatrix} = \begin{bmatrix} 4 & -18 \\ -12 & -14 \end{bmatrix} \quad (I.3)$$

```
>> MA = [1 5 ; -3 2]
MA =
   1   5
  -3   2
```

```
>> MB = [4 2 ; 0 -4]
MB =
   4   2
   0  -4
```

```
>> MPR2 = MA*MB
MPR2 =
   4  -18
 -12  -14
```

C. Opérations matricielles : division et inverse

La division est évaluée à partir de l'inverse selon l'expression matricielle (I.4):

$$MA / MB = MA \times MB^{-1} \quad (I.4)$$

Elle nécessite donc que la matrice ***MB*** soit inversible et que les dimensions de ***MA*** et ***MB*** soient compatibles, sous MATLAB la division est implantée des deux façons suivantes :

```
>> MA = [1 5 ; -3 2]
MA =
   1    5
  -3    2
```

```
>> MB = [4 2 ; 0 -4]
MB =
   4    2
   0   -4
```

```
>> MM1 = MA/MB
MM1 =
   0.2500   -1.1250
  -0.7500   -0.8750
```

Ou par le produit de la matrice **MA** par l'inverse de la matrice **MB** :

```
>> MA = [1 5 ; -3 2]
MA =
   1    5
  -3    2
```

```
>> MB = [4 2 ; 0 -4]
MB =
   4    2
   0   -4
```

```
>> MM2 = MA*inv(MB)
MM2 =
   0.2500   -1.1250
  -0.7500   -0.8750
```

I.8. Résolution des systèmes linéaires

Le formalisme matriciel est adopté pour résoudre un système d'équations linéaires. Par exemple, on cherche à résoudre le système de l'expression (I.5) :

$$\begin{cases} 8x_1 + x_2 + 6x_3 = -3.5 \\ 3x_1 + 5x_2 + 7x_3 = -8.5 \\ 4x_1 + 9x_2 + 2x_3 = 4.5 \end{cases} \quad \text{(I.5)}$$

Mathématiquement ; c'est le produit d'une matrice notée $\boldsymbol{MM}$ (de taille 3×3) par un vecteur colonne des valeurs inconnues noté $\boldsymbol{VX}$ (de taille 3) qui est égal à un vecteur colonne des valeurs connues noté $\boldsymbol{VC}$ (de taille 3) ; tel que $\boldsymbol{MM \times VX = VC}$

Pour obtenir la solution du système linéaire ; il faut inverser la matrice $\boldsymbol{MM}$ et calculer le produit $\boldsymbol{MM^{-1} \times VC}$ formulé par l'expression (I.6) :

$$\boldsymbol{VX = MM^{-1} \times VC} \quad \text{après résolution ; on aura } \boldsymbol{VX} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad \text{(I.6)}$$

Alors ; sous MATLAB :

On déclare la matrice $\boldsymbol{MM}$ à partir des coefficients du système :

$$\boldsymbol{MM} = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

On déclare le vecteur $\boldsymbol{VC}$ à partir des coefficients du système :

$$\boldsymbol{VC} = \begin{pmatrix} -3.5 \\ -8.5 \\ 4.5 \end{pmatrix}$$

On cherche le vecteur VX à partir du produit de l'inverse de la matrice MM par le vecteur VC, les instructions correspondantes sont :

```
>> MM = [8 1 6; 3 5 7; 4 9 2]
MM =
     8     1     6
     3     5     7
     4     9     2
```

```
>> VC = [-3.5; -8.5; 4.5]
VC =
   -3.5
   -8.5
    4.5
```

```
>> VX = inv(MM)*VC
VX =
   1.000
   0.500
  -2.000
```

On peut alors vérifier que :

```
>> (MM(1,1)*VX(1))+(MM(1,2)*VX(2))+(MM(1,3)*VX(3))
ans =
   -3.5000
```

I.9. Opérations sur les polynômes

Sous MATLAB un polynôme est défini par un vecteur des coefficients; Si **P** est un polynôme d'ordre 3 par exemple : alors $P(x) = a\,x^3 + b\,x^2 + c\,x + d$

D'où : **P(1) = a**, **P(2) = b**, **P(3) = c** et **P(4) = d** ; alors **P = [a b c d]**

Exemples : **P = [1 0 -2]** défini le polynôme : $x^2 - 2$ et **P = [2 0 0 0]** défini le polynôme : $2x^3$

Le polynôme **P** ayant l'expression : $P(x) = x^3 - 6\,x^2 + 11\,x - 6$, son vecteur caractéristique est le suivant :

```
>> P = [1 -6 11 -6]
P =
     1    -6    11    -6
```

Pour évaluer le polynôme ***P(x)*** pour le scalaire **X1** on calcule le scalaire **Y1 = P(X1)** par la syntaxe **polyval(P, X1)** :

```
>> X1 = 1
X1 =
     1
>> Y1 = polyval(P, X1)
Y1 =
     0
```

La même syntaxe est employée pour évaluer le polynôme ***P(x)*** pour le vecteur **VX**; alors le vecteur correspondant **VY** est évalué par l'instruction **polyval(P, VX)** :

```
>> VX = 0:0.5:4
VX =
         0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000
>> VY = polyval(P, VX)
VY =
  -6.0000   -1.8750         0    0.3750         0   -0.3750         0    1.8750    6.0000
```

Les racines du polynôme ***P(x)*** sont évaluées par la syntaxe **roots(P)** :

```
>> VR = roots(P)
VR =
   3.0000
   2.0000
   1.0000
```

I.10. Représentations graphiques

Sous MATLAB la représentation graphique basique employée pour créer un graphique a deux dimensions avec deux axes linéaires est introduite par l'instruction **plot (VX,VY)** dont les arguments **VX** (valeurs en abscisses) et **VY** (valeurs en ordonnées) sont des vecteurs de la même longueur. La commande plot permet aussi de représenter plusieurs couples de vecteurs dans un même graphique. L'instruction introduite est :

```
plot(VX1,VY1, VX2,VY2,VX3,VY3, …, VXN,VYN)
```

L'instruction peut être cependant spécifiée en ajoutant des arguments optionnels :

```
plot (VX, VY, 'LineSpec', 'PropertyName', 'PropertyValue')
```

LineSpec spécifie les styles et les couleurs des lignes ainsi que les types des marqueurs, qui peuvent être combinés par conséquence comme indiqué sur le tableau I.1 :

Tableau I.1 : Styles, couleurs et marqueurs de la commande plot.

Style	**LineSpec**	Couleur	**LineSpec**	Marqueur	**LineSpec**
Solide	**-**	Rouge	**r**	Cercle	**o**
Traitillé	**--**	Vert	**g**	Astérisque	*****
Pointillé	**:**	Bleu	**b**	Carré	**s**
Point-trait	**-.**	Jaune	**y**	Croix	**x**

PropertyName et **PropertyValue** permettent de désigner l'épaisseur des lignes, la taille des marqueurs ainsi que la couleur des contours. Le paramétrage des lignes et des marqueurs est montré sur le tableau I.2 :

Tableau I.2 : Paramétrage des lignes et des marqueurs.

PropertyName	**Description**	**PropertyValue**
LineWidth	Epaisseur de la ligne	scalaire (par défaut = 0.5)
Markersize	Taille du marqueur	scalaire (par défaut = 10)
MarkerEdgeColor	Couleur de la bordure du marqueur	r, g, b, … (voir tableau I.1)
MarkerFaceColor	Couleur du marqueur	r, g, b, … (voir tableau I.1)

Pour mettre en place une légende pour chaque graphique avec **N** nombre de couples de vecteurs **(VX,VY)** :

legend('Légende1',…,' LégendeN')

Pour manipuler l'emplacement de la légende, MATLAB offre les possibilités basiques regroupées dans le tableau I.3 :

Tableau I.3 : Paramétrage de la localisation des légendes.

Position	Location spécifiée	Description
1	NorthEast	En haut à droite
2	NorthWest	En haut à gauche
3	SouthWest	En bas à droite
4	SouthEast	En bas à gauche

Pour ajouter un titre sous forme d'une chaîne de caractères **'Titre du graphique'** à la représentation graphique, l'instruction correspondante est ;

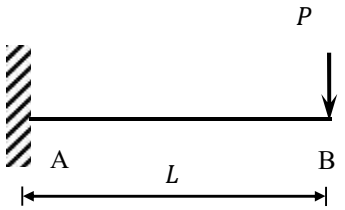
title('Titre du graphique')

L'instruction employée pour affecter des titres (chaînes de caractères) aux axes : **xlabel** pour l'abscisse et **ylabel** pour l'ordonnée est :

```
xlabel('Titre axe (x)')
ylabel('Titre axe (y)')
```

- **Exemple de représentation graphique**

On s'intéresse à tracer les graphes de l'effort tranchant (Equation I.7) et du moment fléchissant (Equation I.8) de la console schématisée sur la figure I.2, de longueur L = 2m soumise à une force ponctuelle P = 5N. Les instructions nécessaires sont listées et commentées ligne par ligne :



$$T(x) = P \quad \text{(I.7)}$$

$$M(x) = Px - PL \quad \text{(I.8)}$$

Figure I.2 : Exemple choisi pour la représentation graphique.

>> clc, clear all, close all

Effacer l'écran de la fenêtre de commande, supprimer toutes les variables introduites précédemment et fermer toutes les fenêtres graphiques crées précédemment.

>> L = 2

Introduire la longueur L = 2. L'exécution de cette ligne d'instruction affiche :

L =

2

>> P = 5

Introduire la force ponctuelle P = 5. L'exécution de cette ligne d'instruction affiche :

P =

5

>> VX = 0:L/4:L

Créer le vecteur des positions x de 0 à L avec un pas de L/4. L'exécution de cette ligne d'instruction affiche :

VX =

0 0.5000 1.0000 1.5000 2.0000

>> PY = [0]

Vecteur caractéristique d'un polynôme à termes nuls, employé pour tracer la ligne horizontale. L'exécution de cette ligne d'instruction affiche :

PY =

0

>> PT = [P]

Vecteur caractéristique du polynôme $T(x)$ de l'effort tranchant. L'exécution de cette ligne d'instruction affiche :

PT =

5

>> PM = [P -PL]

Vecteur caractéristique du polynôme $M(x)$ du moment fléchissant. L'exécution de cette ligne d'instruction affiche :

PM =

5 -10

>> VY = polyval(PY,VX)

Vecteur VY ayant les valeurs nulles de l'ordonnée **y** le long de la console. L'exécution de cette ligne d'instruction affiche :

VY =

0 0 0 0 0

>> VT = polyval(PT,VX)

Vecteur VT ayant les valeurs de l'effort tranchant le long de la console. L'exécution de cette ligne d'instruction affiche :

VT =

5 5 5 5 5

>> VM = polyval(PM,VX)

Vecteur VM ayant les valeurs du moment fléchissant le long de la console. L'exécution de cette ligne d'instruction affiche :

VM =

-10.0000 -7.5000 -5.0000 -2.5000 0

>> figure

Créer une fenêtre graphique vierge. L'exécution de cette ligne d'instruction affiche la figure I.3 :

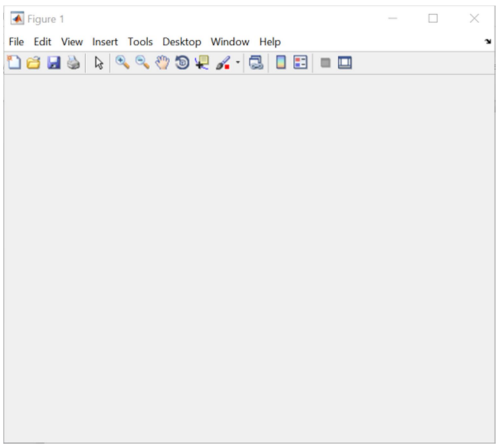


Figure I.3 : Fenêtre graphique vierge.

>> plot(VX,VY,VX,VT,'-b*',VX,VM,'--gs','LineWidth',2)

Tracer les couples de vecteurs : **(VX,VY)** relatif à la ligne horizontale, **(VX,VT)** relatif à l'effort tranchant en trait continu de couleur bleue d'un marqueur astérisque et **(VX,VM)** relatif au moment fléchissant en trait discontinu de couleur verte d'un marqueur en forme carrée. L'ensemble des lignes ont une épaisseur doublée. L'exécution de cette ligne d'instruction affiche la figure I.4 :

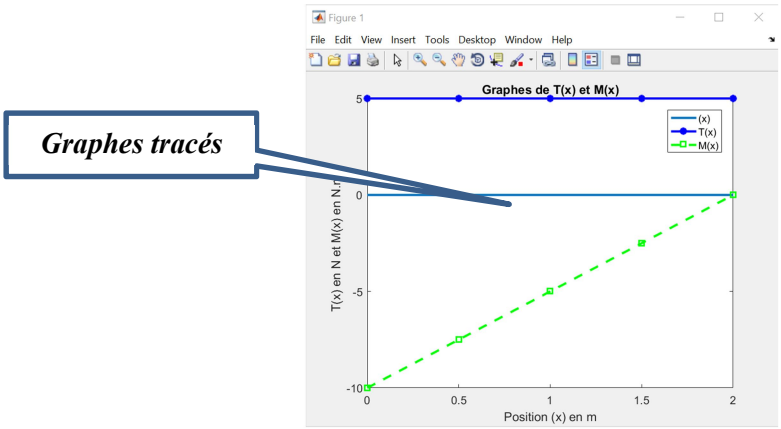


Figure I.4 : Plot des graphes T(x) et M(x).

>> legend({'(x)','T(x)','M(x)'},'Location','NorthEast')

Créer une légende ayant les indications **(x)**, **T(x)** et **M(x)**. La légende est positionnée au Nord-Est de la fenêtre graphique. L'exécution de cette ligne d'instruction affiche la figure I.5 :

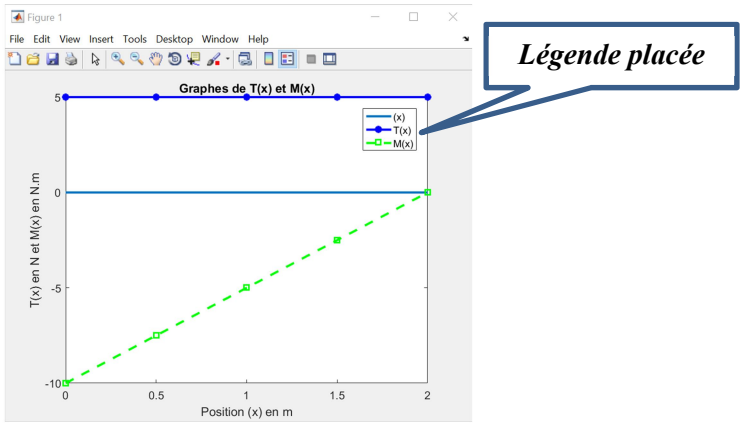


Figure I.5 : Mise en place de la légende.

>> title('Graphes de T(x) et M(x)')

Créer un titre de la fenêtre graphique formé de la chaîne de caractères **'Graphes de T(x) et M(x)'**. L'exécution de cette ligne d'instruction affiche la figure I.6 :

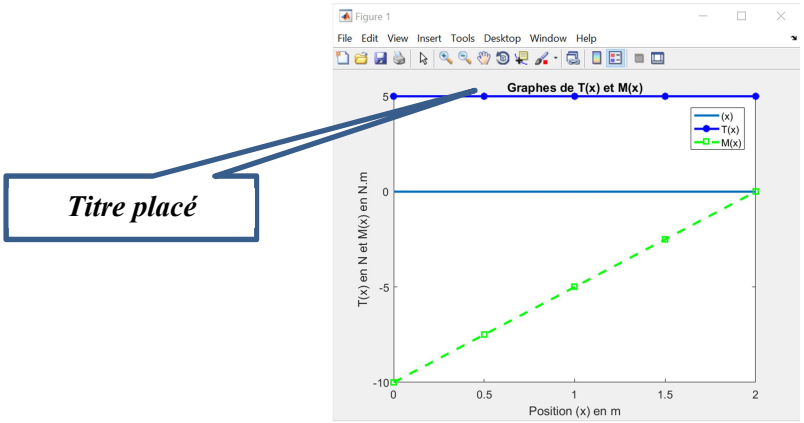


Figure I.6 : Mise en place d'un titre au graphique.

>> xlabel('Position (x)')

Créer un titre de l'axe horizontal formé de la chaîne de caractères **'Position (x) en m'**. L'exécution de cette ligne d'instruction affiche la figure I.7 :

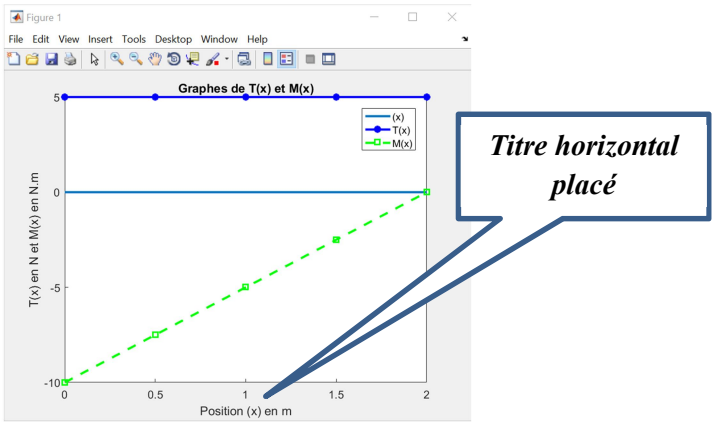


Figure I.7 : Mise en place d'un titre à l'axe horizontal.

>> ylabel('T(x) en N et M(x) en N.m')

Créer un titre de l'axe vertical formé de la chaîne de caractères **'T(x) en N et M(x) en N.m'**. L'exécution de cette ligne d'instruction affiche la figure I.8 :

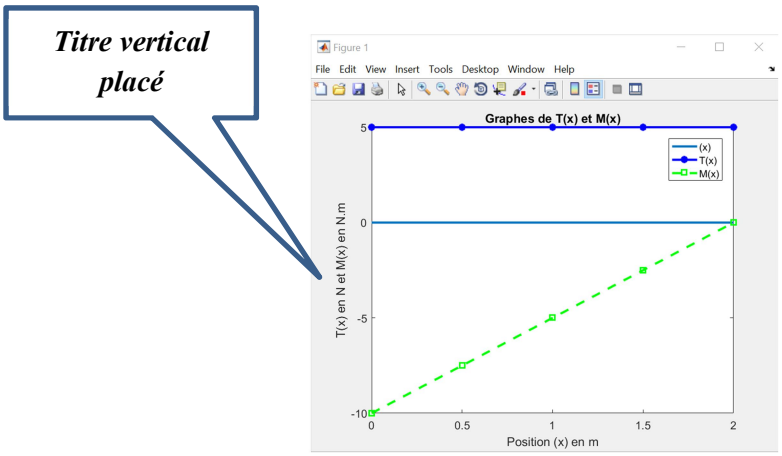


Figure I.8 : Mise en place d'un titre à l'axe vertical.

Chapitre II :	Programmation modulaire par le langage MATLAB
II.1. II.2. II.3.	Introduction Eléments de base Eléments avancés

II.1. Introduction

Le présent chapitre est réservé à la programmation modulaire par le langage MATLAB ; à savoir les éléments de base : fichiers SCRIPT et les fichiers USER-FUNCTION. La manipulation de certains éléments avancés : les modes d'entrée / sortie et les sous-programmes est aussi abordée.

II.2. Eléments de base

En complément de l'éditeur des commandes (Command Windows) ; le langage MATLAB permet à l'utilisateur l'emploi d'un éditeur des fichiers, ces fichiers peuvent être ordinaires ou dédiés à une fonction.

II.2.1. Fichiers SCRIPT

Un **m-file** est un fichier contenant une suite d'instructions du langage MATLAB. Pour créer un **m-file**, utilisez le menu **File → new → Blank M-File**. Une fois le M-file est créé et sauvegardé sous le nom **Fichier1**, les instructions sont introduites dans des lignes numérotées et le bouton **Run** (triangle vert) est utilisé pour lancer le calcul.

Autrement ; l'utilisateur peut aller à la fenêtre de commande et taper le nom du fichier sans l'extension ***.m** ; c.à.d. uniquement le mot **Fichier1** (Figure II.1).

```
>> Fichier1
```

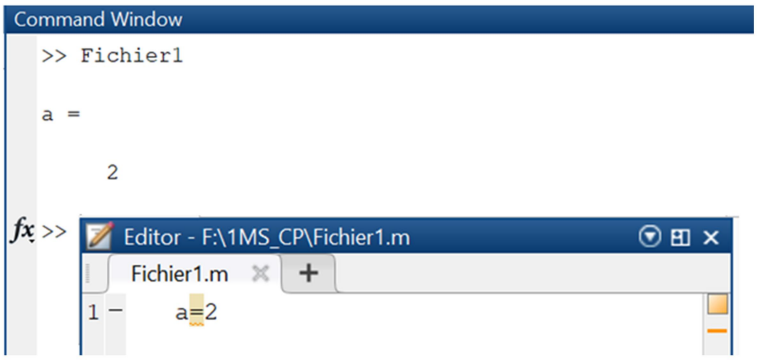


Figure II.1 : Exécution du fichier (Fichier1.m).

II.2.2. Fichiers USER-FUNCTION

Pour des certains raisons d'organiser le traitement, d'éviter la répétions, …, etc. ; l'utilisateur est souvent obligé de créer ses propres fonctions (**USER FUNCTIONS**), qui sont écrites sous la forme suivante :

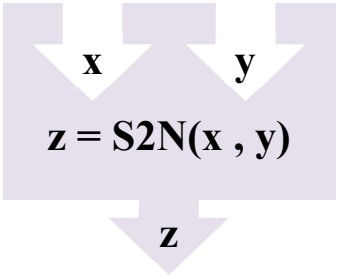
Y = NF (X)

Sortie = nom de la fonction (Entrée)

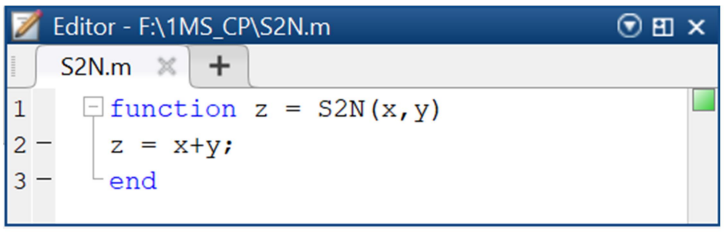
Pour effectuer le calcul on tape le nom de la fonction y compris les arguments d'entrée ; c.à.d. les valeurs des variables pour lesquelles on veut faire le calcul.

Exemple : la fonction **S2N** de la figure II.2 renvois la somme des deux nombres **x** et **y** (**z = x+y**) ; c.à.d.

z = S2N(x,y)



(a)



(b)

Figure II.2 : Fonction définie par l'utilisateur S2N [(a) Schéma et (b) Code].

Donc, il suffit de taper **S2N(10,5)** dans la fenêtre de commandes pour avoir comme résultats la somme des entrées **10** et **5** :

```
>> S2N(10,5)
ans =
   15
```

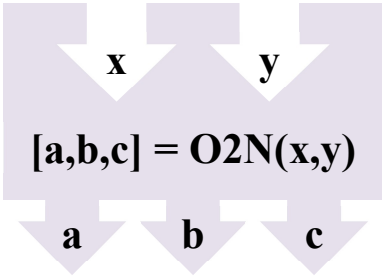
Une fonction définie par l'utilisateur peut renvoyer plusieurs données : dans ce cas la liste des noms de variables ou ces données seront rangées et introduites entre crochets dans l'ordre imposé par le programmeur :

[sortie1, sortie2, ...] = nom de la fonction (entrée1, entrée2, ...)

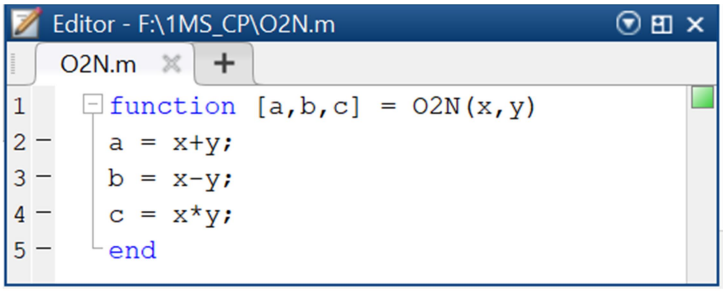
Exemple: la fonction **O2N** de la figure II.3 renvois les trois opérations des deux nombres **x** et **y** introduits dans l'ordre ; c.à.d. :

[a,b,c] = O2N(x,y)

$a = O2N_1(x,y) = x+y$, $b = O2N_2(x,y) = x-y$ et $c = O2N_3(x,y) = x \times y$



(a)



(b)

Figure II.3 : Fonction définie par l'utilisateur O2N [(a) Schéma et (b) Code].

Donc il suffit de taper **O2N(10,5)** dans la fenêtre de commandes pour avoir directement la somme, la différence et le produit de **10** et **5** affectées respectivement aux **sortie1**, **sortie2** et **sortie3** notées **e**, **f** et **g** :

```
>> [e,f,g] = O2N(10,5)
e =
   15
f =
    5
g =
   50
```

II.3. Eléments avancés

Les points abordés dans cette partie du chapitre offrent à l'utilisateur des fonctionnalités avancées pour la manipulation des entrées (inputs) et des sorties (outputs) ainsi qu'à la gestion des programmes.

II.3.1. Manipulation des entrées / sorties

Les commandes suivantes concernent la source des entrées et des sorties par une simple lecture / écriture par Workspace ou par des importations / exportation à partir des fichiers externes (souvent de format Texte et Excel).

A. Entrées / Sorties par Workspace

L'utilisateur peut dans l'exemple ci-dessous saisir un nombre que le script pourra alors utiliser. La commande **input** permet de demander à l'utilisateur fournir des données sous forme d'un dialogue. Par exemple :

```
N = input('Donnez un nombre pour calculer sa racine carre :');
```

La chaîne de caractères écrite entre guillemets **'Donnez un nombre pour calculer sa racine carre :'** est affichée et le code MATLAB attend que l'utilisateur tape une donnée au clavier, ensuite la valeur numérique de la réponse au dialogue sera affecter à la variable **N**.

```
M = sqrt(N);
```

Affecter à la variable **M** la racine carrée de la variable introduite **N**.

```
CC1 = 'Le nombre entre est :';
CC2 = num2str(N);
```

L'instruction **num2str(N)** permet de transformer la valeur numérique du variable **N** à une chaîne de caractères qui peut être afficher par l'instruction d'affichage **disp**.

```
CC12 = [CC1 CC2]
```

L'instruction entre crochets **CC12 = [CC1 CC2]** permet de regrouper dans l'ordre les deux chaînes de caractères **CC1** et **CC2** et les transformées à une seule chaîne de caractères nommée **CC12**.

```
CC3 = 'Sa racine caree est :';
CC4 = num2str(M);
```

Identique à la précédente ; la ligne d'instruction **num2str(M)** permet de transformer la valeur numérique du variable **M** à une chaîne de caractères qui peut être affiché par l'instruction d'affichage **disp**.

```
CC34 = [CC3 CC4]
```

Identique à la précédente ; la ligne d'instruction entre crochets **CC34 = [CC3 CC4]** permet de regrouper dans l'ordre les deux chaînes de caractères **CC3** et **CC4** et les transformées à une seule chaîne de caractères nommée **CC34**.

```
disp([CC12])
disp([CC34])
disp('Fin')
```

Ces trois lignes permettent l'affichage des chaînes de caractères **CC12**, **CC34** et celle formée par l'instruction **'Fin'**.

L'exécution des lignes précédentes et la saisie du nombre 4 permettent d'avoir l'affichage suivant :

```
Donnez un nombre pour calculer sa racine carre : 4
Le nombre entre est : 4
Sa racine caree est : 2
Fin
```

B. Entrées / Sorties par fichier Texte

Pour importer des données à partir des fichiers texte d'extension **.txt** et **.dat** constitués des nombres et / ou des textes ayant le même nombre et le même ordre sur chaque ligne ; la commande à utiliser est **textread**.

```
[sortie1, Sortie2, …, SoriteN] = textread('Fichier à lire', 'Format de lecture')
```

Soit par exemple le fichier **Fichier1.txt** ; dont les données à importer sont organiser en trois colonnes : le première contiens des nombre entiers, le deuxième inclut des réels et la troisième colonne comporte des chaînes de caractères (Figure II.4).

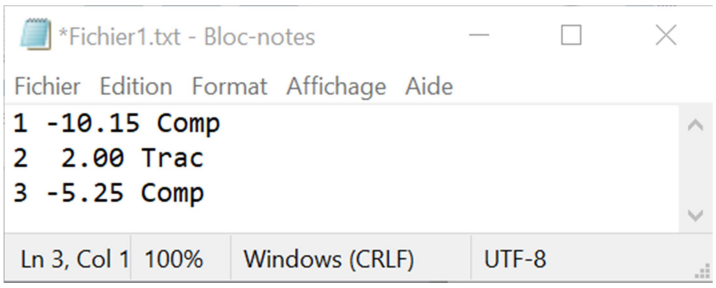


Figure II.4 : Données à importer depuis un fichier Texte.

Les instructions de lecture sont :

```
FLec = 'Fichier1.txt';
```

Le nom du fichier à lire est affecté à la variable d'identification **FLec**.

```
FFormat = '%d %f %s';
```

Le format de lecture (affecté à la variable **FFormat**) est composé de trois constituants relatifs aux trois colonnes à importer. Les caractères **%d**, **%f** et **%s** corresponds respectivement aux types de données : nombres entiers, des réels et des chaînes de caractères.

```
[V1, V2, V3] = textread(FLec,FFormat)
```

Les trois colonnes de données lues sont stockées respectivement dans les vecteurs **V1**, **V2** et **V3**. En MATLAB les correspondances entre les constituantes à lire et celles stockées sont regroupées dans le tableau II.1 :

Tableau II.1 : Arguments de l'instruction Textread.

Caractère	Entité en entrée	Entité en sortie
%d	Entier	**Double**
%f	Nombre à virgule	**Double**
%s	Chaîne de caractères	**Cell** (à transformer au type **"char"**)

Pour le présent exemple la commande **whos** permet d'identifier les types des vecteurs importés :

```
>> whos V1 V2 V3
```

```
  Name        Size                Bytes  Class
  V1          3x1                    24  double
  V2          3x1                    24  double
  V3          3x1                   360  cell
```

Alors, la variable **V3** est un vecteur de type **"cell"**, pour le transformer à un vecteur de type **"char"**, il faut utiliser la commande **cell2mat** (conversion de type **"cell"** vers le type **"char"**).

```
V3T = cell2mat(V3)
```

Ce qui est confirmé par la commande **whos** :

```
>> whos V3T
```

```
  Name        Size                Bytes  Class
  V3T         3x4                    24  char
```

Après le traitement ; les sorties sont :

```
V1 =
     1
     2
     3

V2 =
   -10.1500
     2.0000
    -5.2500

V3 =
    'Comp'
    'Trac'
    'Comp'

V3T =
Comp
Trac
Comp
```

Pareillement ; pour exporter des sorties de MATLAB vers un fichier Texte ; les étapes sont :

1) La création du fichier : **fopen** ; les paramètres de la fonction sont le nom du fichier et la permission d'écriture (soit par écrasement de l'ancien contenu ou par ajout ; si le fichier existe déjà et sa création si il n'existe pas). Les clés de permissions sont les suivantes :

- **'wt'** : l'ancien contenu est écrasé ;
- **'a'** : le nouveau contenu est ajouté à la fin du fichier.

La fonction retourne un identifiant d'écriture :

```
fid = fopen(filename, permission);
```

2) L'écriture dans les fichiers se fait par la fonction **fprintf**. Les paramètres sont l'identifiant du fichier, le format, ensuite les données :

```
count = fprintf(fid, format, A, ...)
```

L'argument du format est un texte contenant des caractères et des spécifications de conversion des données :

- En cas d'un entier (**%d**) : largeur du champ d'impression ; **%nd** spécifie qu'au moins **n** caractères seront réservés pour imprimer l'entier. Par défaut, la donnée sera cadrée à droite du champ. Le signe (**-**) avant le format signifie que la donnée sera justifié à gauche du champ (**% - nd**).
- En cas d'un flottant (**%f**) : la précision est ajoutée à la largeur du champ d'impression ; **%.mf** signifie qu'un flottant sera imprimé avec **m** chiffres après la virgule. Aussi **%n.mf** signifie que l'on réserve **n** caractères (incluant le caractère du point) pour imprimer le flottant et que 2 d'entre eux sont destinés aux chiffres après la virgule. Lorsque la précision n'est pas spécifiée, elle correspond par défaut à 4 chiffres après la virgule.
- Le retour à la ligne est géré par la syntaxe **\n**.

L'exemple ci-dessous crée le fichier **Fichier2.txt** dans lequel on export le vecteur **Vx** et les racines carrées de ses éléments :

```
Vx = 0:2:10
```

Vecteur composé des entiers allons de 0 à 10 avec un pas de 2.

```
Vy = sqrt(Vx)
```

Vecteur des racines carrées du vecteur **Vx**.

```
MM = [Vx; Vy]
```

Matrice à exporter est composée de deux lignes ; en premier les éléments du vecteur **Vx** et en deuxième les éléments du vecteur **Vy**.

fid = fopen(' Fichier2.txt','wt');

L'identifiant d'ouverture du fichier **Fichier2.txt** avec la permission d'écrasement.

fprintf(fid,'%6d %8.3f\n',MM);

La matrice **MM** est écrite selon les spécifications suivantes : Pour le vecteur **Vx** : **%6d** spécifie qu'au moins 6 caractères seront réservés pour largeur du champ d'impression. Pour le vecteur **Vy** : **%8.3f** spécifie qu'au moins 8 caractères seront réservés pour la partie entière et 3 chiffres pour la partie décimale, la syntaxe **\n** indique que le retour à la ligne est géré après ligne d'écriture.

%6d	**%8.3f**
<---->	<-------->
…...0	….0.000
…...2	….1.414
…...4	….2.000
…...6	….2.449
…...8	….2.828
….10	….3.162

fclose('all');

A la fin d'écriture le fichier crée est refermé. Le fichier crée est présenté sur la figure II.5 :

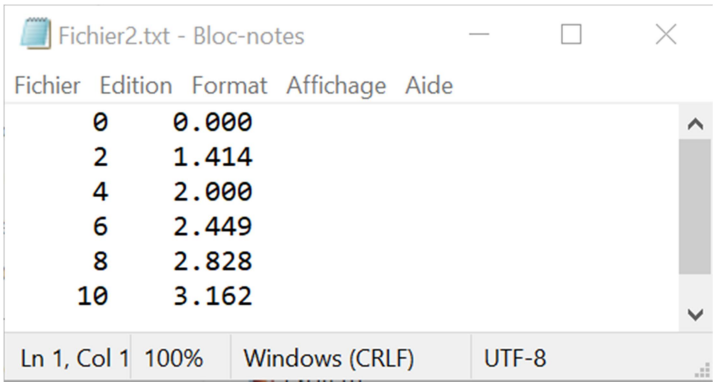


Figure II.5 : Résultats exportés vers un fichier Texte.

C. Entrées / Sorties par fichier Excel

Pour importer des données à partir d'un fichier Excel ; il faut utiliser la commande **xlsread** en précisant la feuille-Excel et les limites des données à lire. L'instruction complète est :

[DataNum, DataTxt] = xlsread('NomFichier', 'NomFeuille', 'Limite')

A la partie droite de l'instruction :

'NomFichier' : est le nom du fichier Excel à lire ;

'NomFeuille' : est le nom de la feuille-Excel à lire ;

'Limite' : sont les limites (colonnes et lignes) des cellules à lire.

A la partie gauche de l'instruction :

DataNum : est la partie numérique des données lues ;

DataTxt : est la partie non-numérique (chaînes de caractères) des données lues en format **Cell**.

Par exemple ; il est demandé de lire les données comprises verticalement entre les colonnes **A** et **C** (c.à.d. l'ensemble des colonnes **A**, **B** et **C**) et horizontalement entre les lignes 1 et 3 (c.à.d. l'ensemble des lignes 1, 2 et 3), ces données sont stockées dans la feuille **FeuilData1** dans le fichier **FExcel1.xlsx** (Figure II.6).

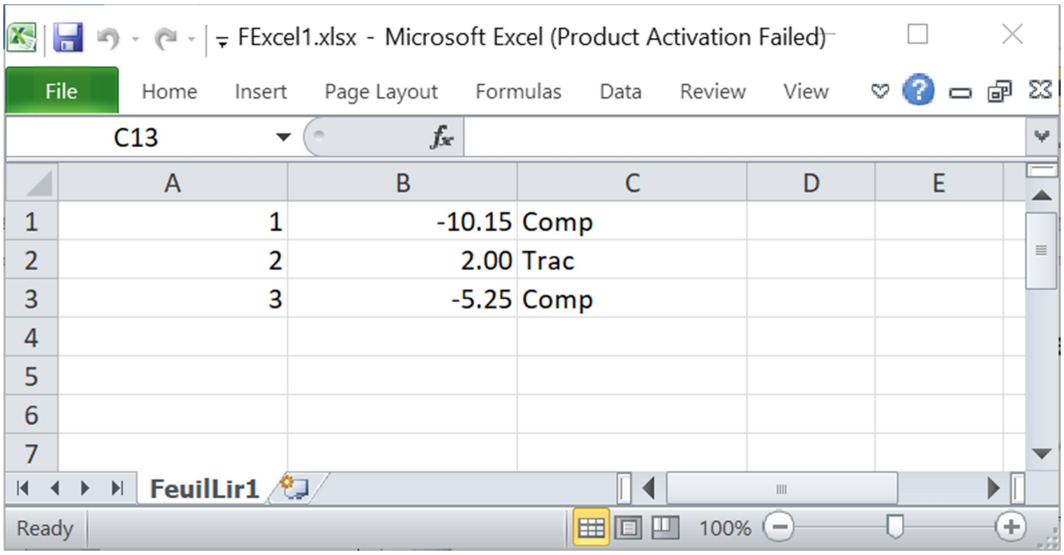


Figure II.6 : Données à importer à partir d'un fichier Excel.

Les instructions détaillées sont :

FicLec = 'FExcel1.xlsx' ;

Chaîne de caractères précisant le nom du fichier Excel à lire.

FeuLec = 'FeuilLir1' ;

Chaîne de caractères précisant le nom de la feuille-Excel à lire.

LimLec = 'A1:C3' ;

Chaîne de caractères précisant les limites des cellules à lire de verticalement de la colonne **A** vers la colonne **C** et horizontalement de la ligne 1 vers la ligne 3.

[P1,P2] = xlsread(FicLec, FeuLec, LimLec)

P1 : la partie stockant les valeurs numériques des données lues.

```
P1 =
   1.0000  -10.1500
   2.0000    2.0000
   3.0000   -5.2500
```

P2 : la partie stockant les valeurs non-numérique des données lues.

```
P2 =
   'Comp'
   'Trac'
   'Comp'
```

```
P2T = cell2mat(P2)
```

Finalement, la partie **P2** est un vecteur de type **"cell"**, pour le transformer en un vecteur de type **"char"**, il faut utiliser la commande **cell2mat** (conversion de type **"cell"** vers le type **"char"**).

```
P2T =
Comp
Trac
Comp
```

Alors ; pour exporter des données vers un fichier Excel ; il faut utiliser la commande **xlswrite** en précisant la feuille-Excel et les cellules concernées par l'écriture. L'instruction complète est :

```
xlswrite('NomFichier', MData,'NomFeuille', 'Limite')
```

Les arguments de l'instruction sont :

'NomFichier' : est le nom du fichier Excel d'écriture ;

MData : est la matrice contenant les données à exporter ;

'NomFeuille' : est le nom de la feuille-Excel d'écriture ;

'Limite' : sont les limites (colonnes et lignes) des cellules concernées par l'écriture.

L'exemple ci-dessous crée le fichier **FExcel2.xlsx** dans lequel on export le vecteur **Vx** et les racines carrées de ses éléments :

```
Vx = 0:2:10
```

Vecteur composé des entiers allons de 0 à 10 avec un pas de 2.

```
Vy = sqrt(Vx)
```

Vecteur des racines carrées du vecteur **Vx**.

```
MO = [Vx; Vy]
```

Matrice à exporter est composée de deux lignes ; en premier les éléments du vecteur **Vx** et en deuxième les éléments du vecteur **Vy**.

ME = MO'

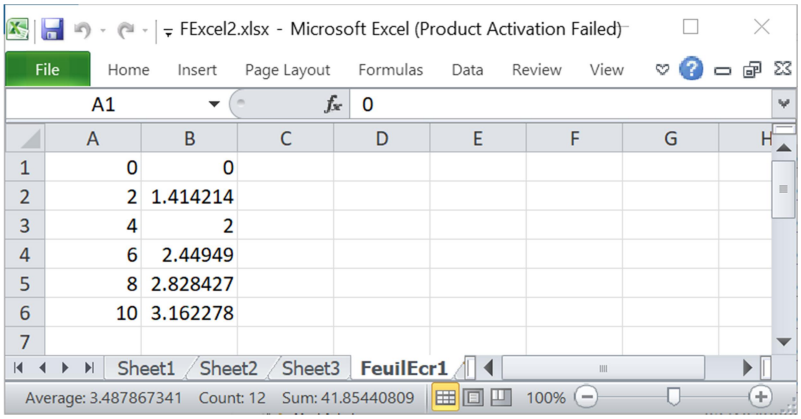
La commande **MO'** indique la transposée de la matrice à exporter. Les informations (nom du fichier Excel, feuille de données et les colonnes-lignes concernées) sont présentées sur la figure II.7.

FicLec = 'FExcel1.xlsx' ;

FeuLec = 'FeuilData1' ;

LimLec = 'A1:C3' ;

FicEcr = 'FExcel2.xlsx';



	A	B
1	0	0
2	2	1.414214
3	4	2
4	6	2.44949
5	8	2.828427
6	10	3.162278

Figure II.7 : Résultats exportés vers un fichier Excel.

Chaîne de caractères précisant le nom du fichier Excel d'écriture.

FeuEcr = 'FeuilEcr1';

Chaîne de caractères précisant le nom de la feuille-Excel d'écriture.

LimEcr = 'A1:B6';

Chaîne de caractères précisant les cellules concernées par l'écriture ; verticalement de la colonne **A** vers la colonne **B** et horizontalement de la ligne 1 vers la ligne 6.

xlswrite(FicEcr, ME, FeuEcr, LimEcr)

Exporter vers la feuille **FeuEcr** du fichier Excel identifié par l'argument **FicEcr** les éléments de la matrice **ME** vers les cellules encadrées par l'argument **LimEcr**.

fclose('all');

A la fin d'écriture le fichier crée est refermé.

II.3.2. Manipulation des sous-programmes

A. Principe

Pour des raisons techniques telles que la détections rapide des erreurs, futurs extensions, mises à jour et bien d'autres ; les programmes sont écrits d'une façon modulaire et hiérarchique ; ils comportent un programme principal (P.P.) et des sous-programmes (S.P.) ces derniers sont souvent écrits sous forme des fonctions-utilisateur (F.U.). Chaque sous-programme (aussi appelé module) est une partie autonome du programme, elle comporte des éléments locales constitués des valeurs d'entrée, un ensemble des instructions et des valeurs de sortie. (Figure II.8)

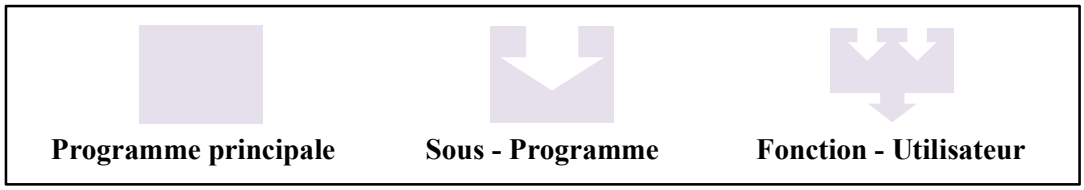
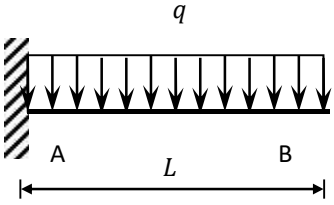


Figure II.8 : Schématisation des composantes modulaires d'un programme.

Par défaut, les variables créées sont locales ; cela signifie qu'elles ne sont connues que dans le script où elles ont été créées. Généralement pour transmettre un argument d'un script aux autres ; il faut le associe à des sous-programmes exécutés séquentiellement. Dans certains cas, il est également possible d'utiliser la notion de variable globale qui permet de rendre visible des variables d'un script à l'autre ; c'est-à-dire que la variable est déclarée comme globale dans le script principal ainsi que dans les scripts attachés à l'aide de la commande **global**.

B. Exemple

L'exemple de la console (Figure II.9) déjà abordé dans la partie de la représentation graphique au première chapitre où il a été demandé de tracer les graphes de l'effort tranchant (Équation II.1) et du moment fléchissant (Équation II.2) ; est récrit dans ce chapitre de forme modulaire en deux version : version en variables locales et une version en variables locales / globales.



$$T(x) = -qx + qL \tag{II.1}$$

$$M(x) = -\frac{q}{2}x^2 + qLx - q\frac{L^2}{2} \tag{II.2}$$

Figure II.9 : Exemple choisi pour la programmation modulaire.

La structuration hiérarchique formulée permet les échanges d'informations entre le programme principal et les trois sous-programmes ; en premier enchaînement le programme principal, communique le nécessaire des données aux sous-programmes **FPTX** (destiné au calcul des valeurs de l'effort tranchant) et **FPMX** (destiné au calcul des valeurs du moment fléchissant), Après recevoir les valeurs de l'effort tranchant et du moment fléchissant ; le programme principal, les communique en deuxième enchaînement au sous-programme **TracTM** pour les représenter graphiquement.

- **Version par variables locales**

Dans cette version le programme principal, les deux fonctions-utilisateur destinées aux calculs des sollicitations et même le sous-programme de traçage sont écrits avec des variables locales (longueur **L**, nombre de division sur la longueur **ND** et la charge uniformément répartie **q**). (Figure II.10)

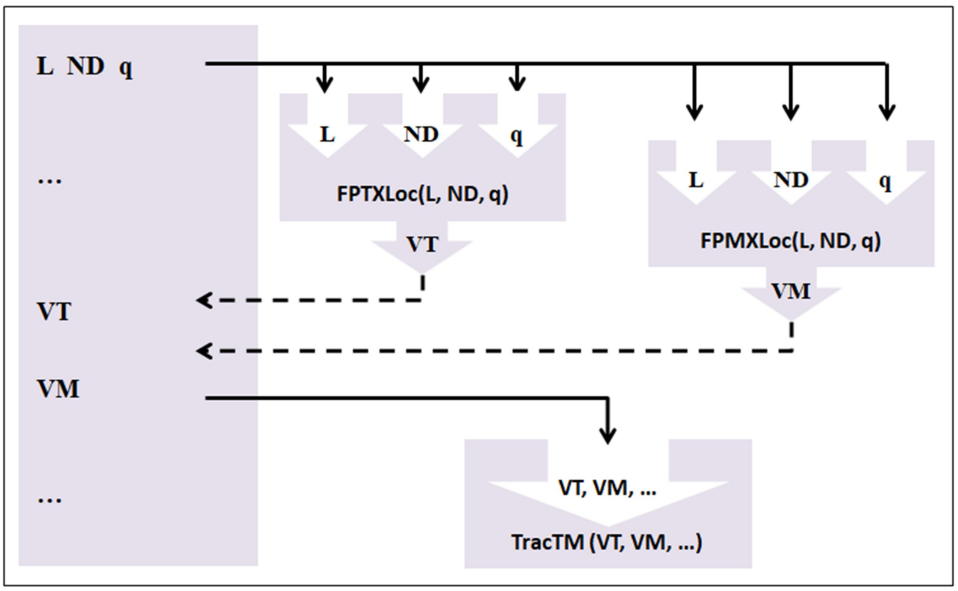


Figure II.10 : Programmation modulaire de l'exemple [schémas de la version a variables locales].

L'ensemble de cette version est détaillé dans les points suivants :

a) Listing commenté du programme principal a termes locales

Introduction des valeurs des données (longueur, nombre de division sur la longueur et la charge uniformément répartie).

```
clc, clear all, close all
% Programme principal par variables locales
L = 2;
ND = 4;
```

q = 5;

Création du vecteur des positions **VX** de 0 à **L** avec quatre (**ND** = 4) divisions.

VX = 0:L/ND:L

Création du vecteur caractéristique d'un polynôme à termes nuls et calcul des valeurs correspondantes.

PY = [0];

VY = polyval(PY,VX)

Le programme principal, communique les données à la fonction-utilisateur **FPTXLoc** (destinée au calcul des valeurs de l'effort tranchant) à termes locales. (L, ND et q).

[VT] = FPTXLoc(L, ND, q)

Le programme principal, communique les données à la fonction-utilisateur **FPMXLoc** (destinée au calcul des valeurs du moment fléchissant) à termes locales. (L, ND et q).

[VM] = FPMXLoc(L, ND, q)

Le programme principal, fait appel au sous-programme **TracTM**, il le communique les valeurs de l'effort tranchant et du moment fléchissant pour les représenter graphiquement.

TracTM

b) Listing commenté de la fonction-utilisateur FPTXLoc a termes locales

Introduction des arguments (longueur, nombre de division sur la longueur et la charge uniformément répartie) de la fonction-utilisateur **FPTXLoc** destinée au calcul des valeurs de l'effort tranchant et quatre lignes de commentaires précédées par le caractère (%).

function [VT] = FPTXLoc(L, ND, q)

% Poutre Isostatique sous Charge Uniformement Repartie

% Entrees : Longueur L(m), Nbr de Division ND et CUR q(N)

% Sorties : VX : vecteur des positions, VVT : valeurs de T(x)

% Utilisation des variables locales

Création du vecteur des positions **VX** de **0** à **L** avec quatre (**ND = 4**) divisions.

VX = 0:L/ND:L;

Création du vecteur caractéristique du polynôme T(x) de l'effort tranchant.

PT = [-q q*L/2];

Calcul des valeurs de l'effort tranchant le long de la console.

VT = polyval(PT,VX);

Clôture de de la fonction-utilisateur **FPTXLoc**.

end

c) Listing commenté de la fonction-utilisateur FPMXLoc a termes locales

Introduction des arguments (longueur, nombre de division sur la longueur et la charge uniformément répartie) de la fonction-utilisateur **FPMXLoc** destinée au calcul des valeurs du moment fléchissant et quatre lignes de commentaires précédées par le caractère (%).

```
function [VM] = FPMXLoc(L, ND, q)
% Poutre Isostatique sous Charge Uniformement Repartie
% Entrees : Longueur L(m), Nbr de Division ND et CUR q(N)
% Sorties : VX : vecteur des positions, VVM : valeurs de M(x)
% Utilisation des variables locales
```

Création du vecteur des positions VX de 0 à L avec quatre (ND = 4) divisions.

```
VX = 0:L/ND:L;
```

Création du vecteur caractéristique du polynôme M(x) du moment fléchissant.

```
PM = [-q/2 q*L -q*(L^2)/2];
```

Calcul des valeurs du moment fléchissant le long de la console.

```
VM = polyval(PM,VX);
```

Clôture de de la fonction-utilisateur **FPTXLoc**.

```
end
```

d) Listing commenté du sous-programme TracTM

Traçage des couples de vecteurs : **(VX,VY)** relatif à la ligne horizontale, **(VX,VT)** relatif à l'effort tranchant en trait continu de couleur bleue d'un marqueur astérisque et **(VX,VM)** relatif au moment fléchissant en trait discontinu de couleur verte d'un marqueur en forme carrée. L'ensemble des lignes ont une épaisseur doublée.

```
% Sous-programme pour trcage de T(x) et M(x)
plot(VX,VY,VX,VT,'-b*',VX,VM,'--gs','LineWidth',2)
```

Création de la légende ayant les indications **(x)**, **T(x)** et **M(x)**. La légende est positionnée au Nord-Est de la fenêtre graphique.

```
legend({'(x)','T(x)','M(x)'},'Location','NorthEast')
```

Création du titre de la fenêtre graphique formé de la chaîne de caractères **'Graphes de T(x) et M(x)'**.

```
title('Graphes de T(x) et M(x)')
```

Création du titre de l'axe horizontale formé de la chaîne de caractères **'Position (x)'**.

```
xlabel('Position (x)')
```

Création du titre de l'axe verticale formé de la chaîne de caractères **'T(x) en N et M(x) en N.m'**.

```
ylabel('T(x) en N et M(x) en N.m')
```

- **Version par variables locales / globales**

Autrement ; dans cette version ; à part le sous-programme de traçage écrit avec des variables locales, le programme principal et les deux fonctions-utilisateur destinées aux calculs des sollicitations sont écrits avec des variables locales (la charge uniformément répartie q) et globales (longueur L et le nombre de division sur la longueur ND). (Figure II.11)

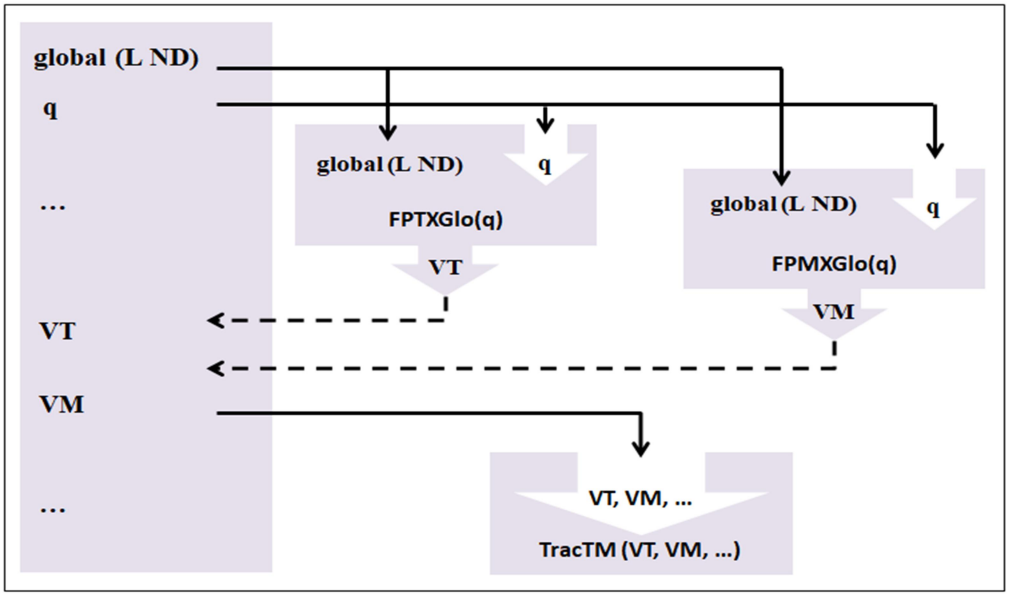


Figure II.11 : Programmation modulaire de l'exemple [schémas de la version a variables locales / globales].

L'ensemble de cette version est détaillé dans les points suivants :

a) Listing commenté du programme principal a termes locales / globales

Déclaration des données globales (longueur, nombre de division sur la longueur) et Introduction des valeurs des données (longueur, nombre de division sur la longueur et la charge uniformément répartie).

```
clc, clear all, close all
% Programme principal par variables globales
global L ND
L = 2;
ND = 4;
q = 5;
```

Création du vecteur des positions **VX** de **0** à **L** avec quatre (**ND = 4**) divisions.

```
VX = 0:L/ND:L
```

Création du vecteur caractéristique d'un polynôme à termes nuls et calcul des valeurs correspondantes.

```
PY = [0];
VY = polyval(PY,VX)
```

Le programme principal, communique les données à la fonction-utilisateur **FPTXGlo** (destinée au calcul des valeurs de l'effort tranchant) à termes locales : (q) uniquement.

```
[VT] = FPTXGlo(q)
```

Le programme principal, communique les données à la fonction-utilisateur **FPMXGlo** (destinée au calcul des valeurs du moment fléchissant) à termes locales : (q) uniquement.

```
[VM] = FPMXGlo(q)
```

Le programme principal, fait appel au sous-programme **TracTM**, il le communique les valeurs de l'effort tranchant et du moment fléchissant pour les représenter graphiquement.

```
TracTM
```

b) Listing commenté de la fonction-utilisateur FPTXGlo a termes locales / globales

Le listing rassemble ; l'introduction des arguments (charge uniformément) et la déclaration des données globales (longueur et nombre de division sur la longueur) de la fonction-utilisateur **FPTXGlo** destinée au calcul des valeurs de l'effort tranchant et quatre lignes de commentaires précédées par le caractère (%).

```
function [VT] = FPTXGlo(q)
global L ND
% Poutre Isostatique sous Charge Uniformement Repartie
% Entrees : CUR q(N)
% Sorties : VX : vecteur des positions, VVT : valeurs de T(x)
% Utilisation des variables globales pour L et ND
```

Création du vecteur des positions **VX** de **0** à **L** avec quatre (**ND = 4**) divisions.

```
VX = 0:L/ND:L;
```

Création du vecteur caractéristique du polynôme T(x) de l'effort tranchant.

```
PT = [-q q*L/2];
```

Calcul des valeurs de l'effort tranchant le long de la console.

```
VT = polyval(PT,VX);
```

Clôture de de la fonction-utilisateur **FPTXGlo**.

```
end
```

c) Listing commenté de la fonction-utilisateur FPMXGlo a termes locales / globales

Le listing regroupe ; l'introduction des arguments (charge uniformément répartie) et la déclaration des données globales (longueur et nombre de division sur la longueur) de la fonction-utilisateur **FPTXGlo** destinée au calcul des valeurs du moment fléchissant et quatre lignes de commentaires précédées par le caractère (%).

```
function [VT] = FPMXGlo(q)
global L ND
% Poutre Isostatique sous Charge Uniformement Repartie
% Entrees : CUR q(N)
% Sorties : VX : vecteur des positions, VVT : valeurs de M(x)
% Utilisation des variables globales pour L et ND
```

Création du vecteur des positions VX de 0 à L avec quatre (ND = 4) divisions.

```
VX = 0:L/ND:L;
```

Création du vecteur caractéristique du polynôme $M(x)$ du moment fléchissant.

```
PM = [-q/2 q*L -q*(L^2)/2];
```

Calcul des valeurs du moment fléchissant le long de la console.

```
VM = polyval(PM,VX);
```

Clôture de de la fonction-utilisateur **FPMXGlo**.

```
end
```

d) Listing commenté du sous-programme TracTM

Le sous-programme de traçage **TracTM** est identique dans les deux versions.

- **Synthèse**

Comme indiqué précédemment ; il est avantageux d'écrire les programmes d'une façon modulaire et hiérarchique ; ils doivent comporter plus d'un programme principal des sous-programmes. Selon la nature du problème rencontré ; les variables à utiliser peuvent être d'un type locale ou/et d'un type globale.

Chapitre III : Programmation structurée par le langage MATLAB

III.1. Introduction

Les opérateurs relationnels et logiques employés par le langage MATLAB seront abordés dans ce chapitre. Par la suite les syntaxes des structures de contrôle seront exposées à savoir : les branchements conditionnels (IF) et les instructions de choix (SWITCH) ainsi que les boucles (FOR) et (WHILE).

III.2. Opérateurs relationnels et logiques

Le langage de programmation MATLAB possède des opérateurs qui permettent d'établir des expressions renvoyant en résultat une valeur logique, c'est-à-dire **0** ou **1**. Ces expressions logiques sont généralement utilisées dans les structures de contrôle présentées par la suite.

==	Egal à	~=	Différent de
>	Strictement supérieur à	<=	Inférieur ou égal à
<	Strictement inférieur à	>=	Supérieur ou égal à

Lorsque deux scalaires sont comparés, le résultat est un scalaire qui vaut **1** si la relation est vraie et **0** si elle est fausse. La relation étant testée élément par élément lorsqu'il s'agit des vecteurs, matrices ou chaînes de caractères.

Dans le premier exemple on compare les deux vecteurs **Vx** et **Vy** ; les résultats de la comparaison terme à terme seront stockées dans le vecteur **ans** qui prend la nature et la taille des éléments comparés c.à.d. un vecteur à deux éléments.

```
>> Vx = [0 6]
>> Vy = [3 5]
```

```
>> Vx < Vy
ans =
   1    0
```

Puisque **0** est inférieur à **3** la première valeur du vecteur **ans** prend la valeur de **1** (**vrai**), la deuxième valeur du vecteur **ans** prend la valeur de **0** (**faux**) car le chiffre 5 n'est pas inférieur au chiffre **6**. La comparaison des deux chaînes de caractères **CC1** et **CC2** en utilisant le syntaxe « **~= Différent de** » résulte à la première position d'une réponse négative (**0** c.à.d. **faux**) car la lettre **M** existe dans les deux chaînes dans la même position, tandis que le reste des lettres sont clairement distinctes (A≠I et X≠N) pour cela leurs résultats de comparaison sont positifs (**1** c.à.d. **vrai**).

```
>> CC1 = ('MAX')
>> CC2 = ('MIN')
```

```
>> COMP = CC1 ~= CC2
COMP =
     0     1     1
```

III.3. Structures de contrôle :

Les structures de contrôle sont des mécanismes qui permettent de modifier la séquence d'exécution des instructions.

III.3.1. Structure conditionnelle (IF ... THEN ... ELSE ...)

On a parfois besoin d'exécuter une séquence d'instructions seulement dans le cas où une condition donnée est vérifiée au préalable (Figure III.1). Le mot clé **IF** dirige l'exécution : vers la suite d'instructions 1 si la condition est vraie , ou vers la suite d'instructions 2 si la condition est fausse.

```
if <expression booléenne>
   <suite d'instructions 1 à exécuter si l'expression est VRAIE>
else
  <suite d'instructions 2 à exécuter si l'expression est FAUSSE>
end
```

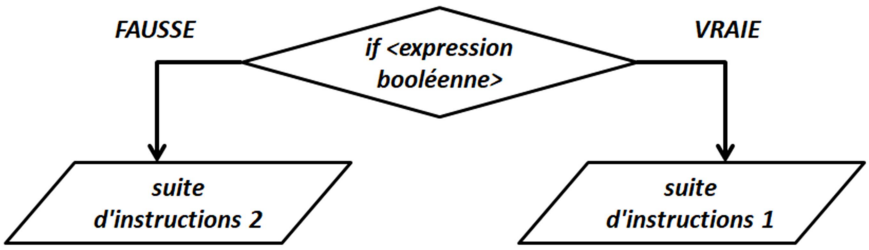


Figure III.1 : Branchement conditionnel (IF) a expression booléenne unique.

En cas de plusieurs expressions booléennes cas de la figure III. 2 ; la séquence prend la forme suivante :

```
if <expression booléenne 1>
    <suite d'instructions 1 exécutée si l'expression 1 est VRAIE>
    elseif <expression booléenne 2>
          <suite d'instructions 2 exécutée si l'expression 1 est FAUSSE
          et que l'expression 2 est VRAIE>
…
else
    <suite d'instructions n exécutée si aucune des expressions n'est VRAIE >
end
```

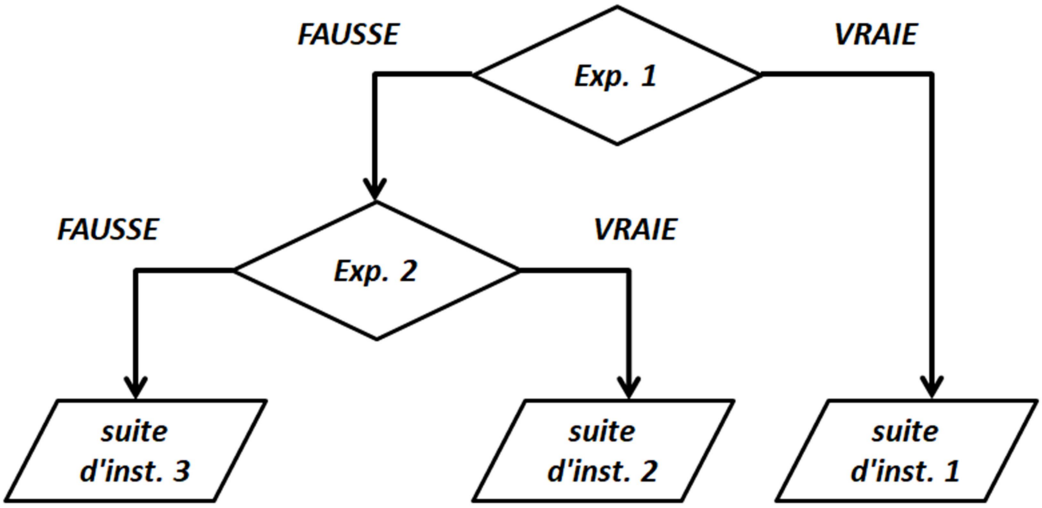


Figure III.2 : Structure conditionnelle (IF) a deux expressions booléennes.

Exemple

Il s'agit de tester la valeur entrée du scalaire **N** non nul ; pour calculer sa valeur absolue **M** et afficher l'expression « Cas négatif » ou « Cas positif ». La logique correspondante est détaillée sous forme d'algorithme et représentée sous forme d'organigramme (Figure III.3).

- **Algorithme**

Données
N : scalaire
Résultats
M : scalaire
Début
Afficher (' Saisir N non nul avec son signe : ')
Saisir (N)
Si N < 0
Alors
M ← -1×N
afficher ('Cas négatif')
Sinon
M ← N
afficher ('Cas positif ')
FinSi
Fin

- **Organigramme**

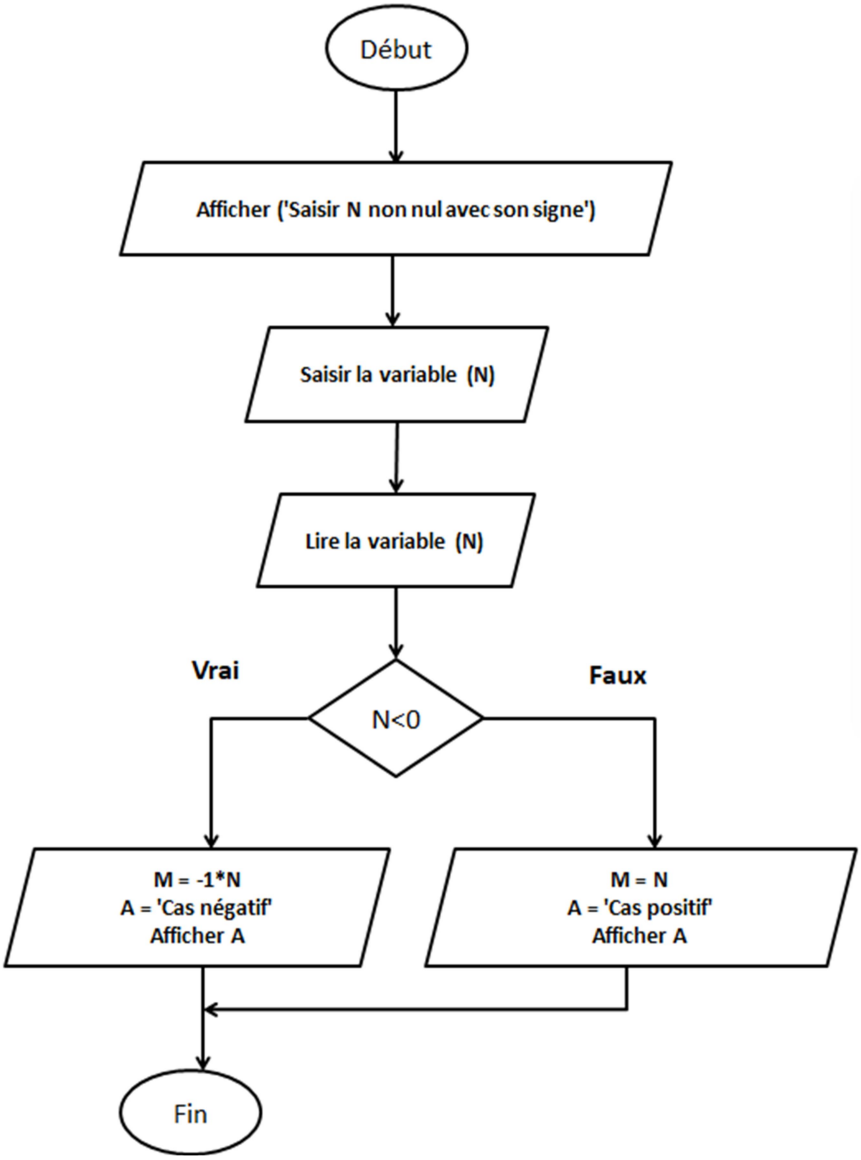


Figure III.3 : Exemple d'utilisation du branchement conditionnel (IF).

- **Programme**

```
N = input('Saisir N non nul avec son signe : ')
if N < 0
   M = -1*N
   disp ('Cas negatif')
else
   M = N
   disp ('Cas positif')
end
```

L'exécution en deux reprise (pour N = 3 et N = -6) donne les informations suivantes :

```
Saisir N non nul avec son signe : 3
N =
    3
M =
    3
Cas positif
```

Et pour N = -6

```
Saisir N non nul avec son signe : -6
N =
   -6
M =
    6
Cas negatif
```

III.3.2. Structure aux choix multiples (Instruction SWITCH)

L'instruction **SWITCH** est une instruction de choix similaire à l'instruction **IF** mais avec la particularité de pouvoir effectuer aisément plusieurs branchements ; celui du le cas 1, sinon celui du cas 2, …, ainsi de suite. Le mot clé **OTHERWISE** signifie l'arrêt de la commande **SWITCH** et que la seule suite d'instructions logiquement possible sera exécuter (Figure III.4).

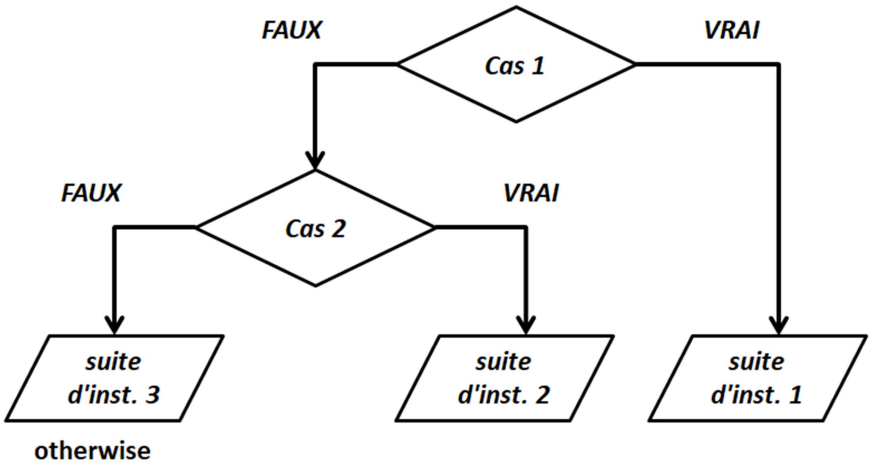


Figure III.4 : Instruction SWITCH à deux choix.

Si le cas 1 est le cas satisfaisant ; le programme exécute la première suite d'instructions. Autrement si le cas satisfaisant est le cas 2 ; le programme exécute la deuxième suite d'instructions. Si non (le cas satisfaisant n'est ni le premier ni le deuxième) ; le programme exécute automatiquement la troisième suite d'instructions.

Exemple

Il s'agit de tester la valeur du scalaire **N** pour afficher le cas correspondant : Si N = 1 ; le programme affiche l'expression « Cas A », Si N = 2 ; il affiche l'expression « Cas B » et Si N ≠ 1 et N ≠ 2 ; il affiche l'expression « Cas C ». La logique correspondante est détaillée sous forme d'algorithme et représentée sous forme d'organigramme (Figure III.5).

- **Algorithme**

```
Données
   N : scalaire
Résultats
   Cas : Ecriture
Début
   Afficher ('Pour Cas A saisir N = 1, Cas B saisir N = 2, N = Autre pour Cas C : ')
   Saisir (N)
   Cas où N Vaut
        1
          Afficher ('Cas A')
        2
          Afficher ('Cas B')
        Autre (ni 1 ni 2)
          Afficher ('Cas C')
   FinCas
Fin
```

- **Organigramme**

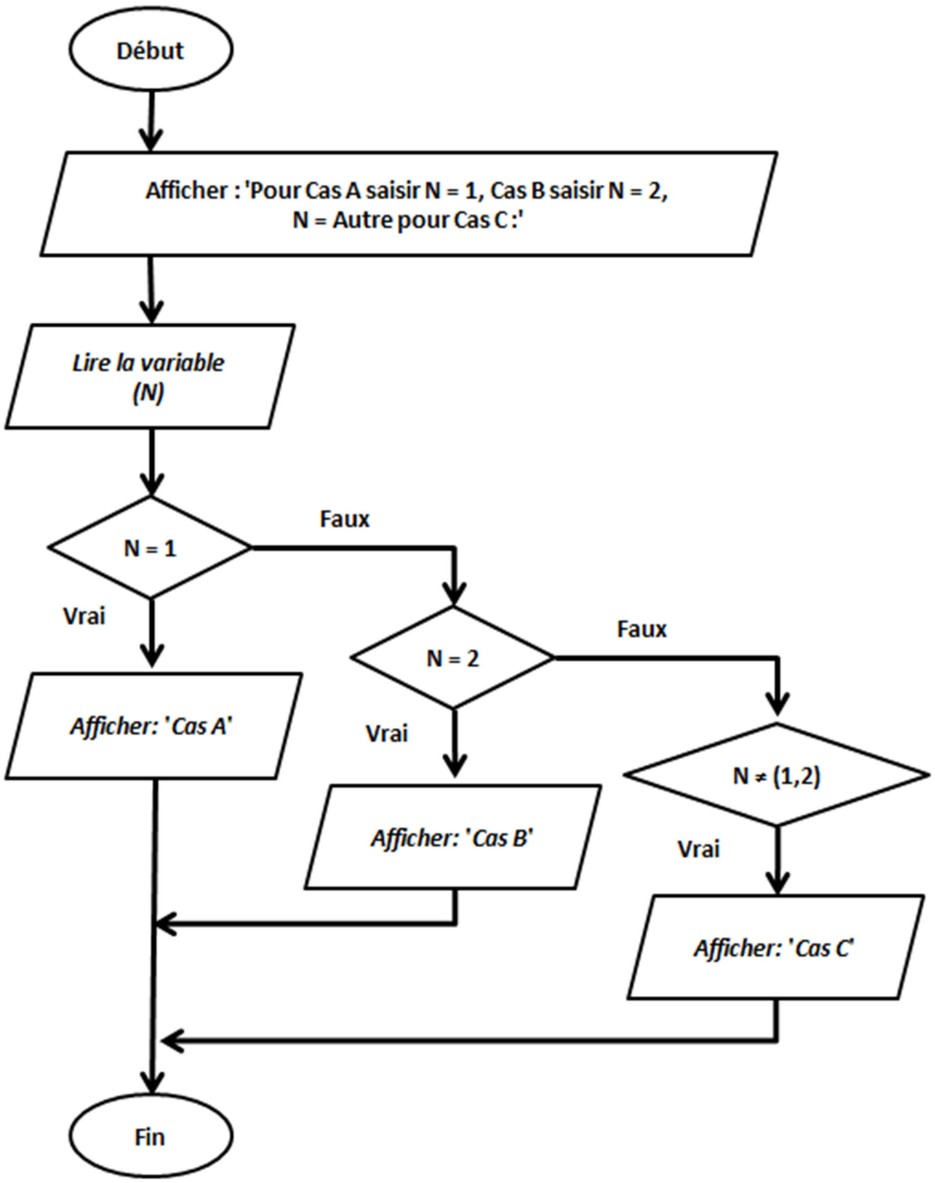


Figure III.5 : Exemple d'utilisation de l'instruction de choix (SWITCH).

- **Programme**

```
N = input('Pour Cas A saisir N = 1, Cas B saisir N = 2, N = Autre pour Cas C : ') ;
switch N
case 1
disp('Cas A')
case 2
disp('Cas B')
otherwise
disp('Cas C')
end
```

L'introduction des nombres 1, 2 et 4 (ni 1 ni 2) permet d'avoir respectivement les résultats suivants :

Pour Cas A saisir N = 1, Cas B saisir N = 2, N = Autre pour Cas C : 1
Cas A

Et

Pour Cas A saisir N = 1, Cas B saisir N = 2, N = Autre pour Cas C : 2
Cas B

Et

Pour Cas A saisir N = 1, Cas B saisir N = 2, N = Autre pour Cas C : 4
Cas C

III.3.3. Structure répétitive (Boucle FOR)

La boucle **for** répète une suite d'instruction pour un nombre prédéfini de fois. Sa structure est la suivante :

for var = <liste des valeurs>
<suite d'instruction>
end

La boucle **for** va exécuter la <**suite d'instruction**> pour chaque élément de la <**liste des valeurs**> en affectant l'élément à la variable **var** (Figure III.6).

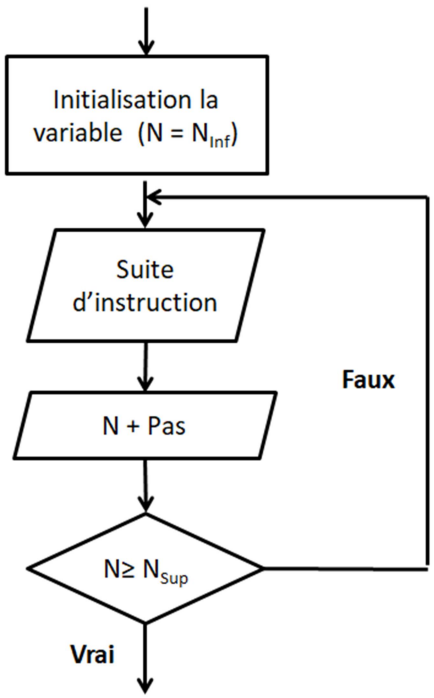


Figure III.6 : Fonctionnement de la boucle (FOR).

Exemple

Pour traiter l'affichage d'une suite de chiffre allant de 0 jusqu'a 4 avec un pas de 2, toute en calculant le carrée de chaque chiffre affiché ; la logique correspondante est détaillée sous forme d'un algorithme et représentée sous forme d'un organigramme (Figure III.7).

- **Algorithme**

Données
i : entier (compteur de boucle)
Résultats
M : scalaire
Début
Pour i allant de 0 à 4 avec un pas de 2
Faire
i
M ← i^2
FinFaire
Fin

- **Organigramme**

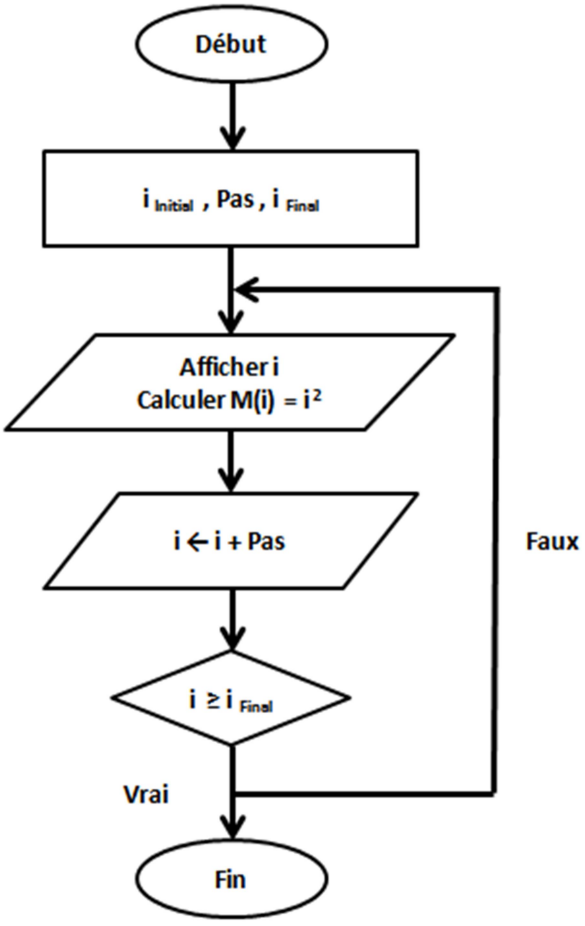


Figure III.7 : Exemple d'utilisation de la boucle (FOR).

- **Programme**

```
for i = 0 : 2 : 4
  i
  M = i^2
end
```

Les lignes suivantes seront affichées :

```
i =
   0
M =
   0
```

```
i =
   2
M =
   4
```

```
i =
   4
M =
  16
```

III.3.4. Structure répétitive - conditionnelle (Boucle WHILE)

Il arrive que nous souhaitions répéter une suite d'instructions jusqu'à qu'une condition soit satisfaite. Si l'on ne connait pas le nombre d'itérations nécessaire à l'avance, une boucle **WHILE** est préférable par rapport à une boucle for :

while <expression booléenne >
<suite d'instructions>
end

La <suite d'instructions> va être répétée tant que l'<expression booléenne> est vrai, ou dit autrement jusqu'à ce que l'<expression booléenne> soit fausse (Figure III.8).

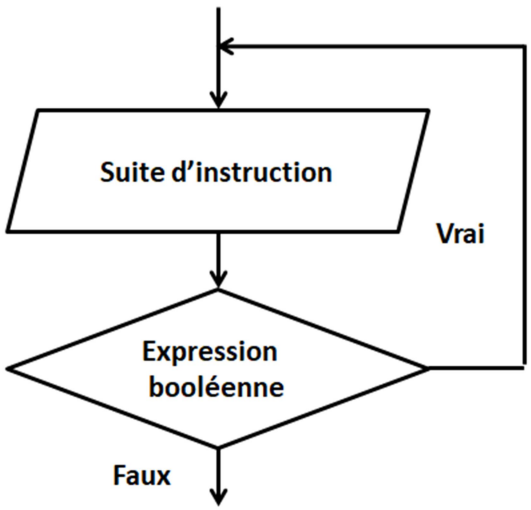


Figure III.8 : Fonctionnement de la boucle (WHILE).

Exemple

On demande à l'utilisateur d'introduire un nombre positif, l'instruction de demande est répétée au tant que le nombre introduit n'est pas positif. La logique correspondante est détaillée sous forme d'algorithme et représentée sous forme d'organigramme (Figure III.9).

- **Algorithme**

Données
N : scalaire
Résultats
Ecriture
Début
Afficher ('Entrez un nombre positif')
Saisir (N)
Tant que N<0
Faire
Afficher ('Entrez un nombre positif')
FinFaire
Fin

- **Organigramme**

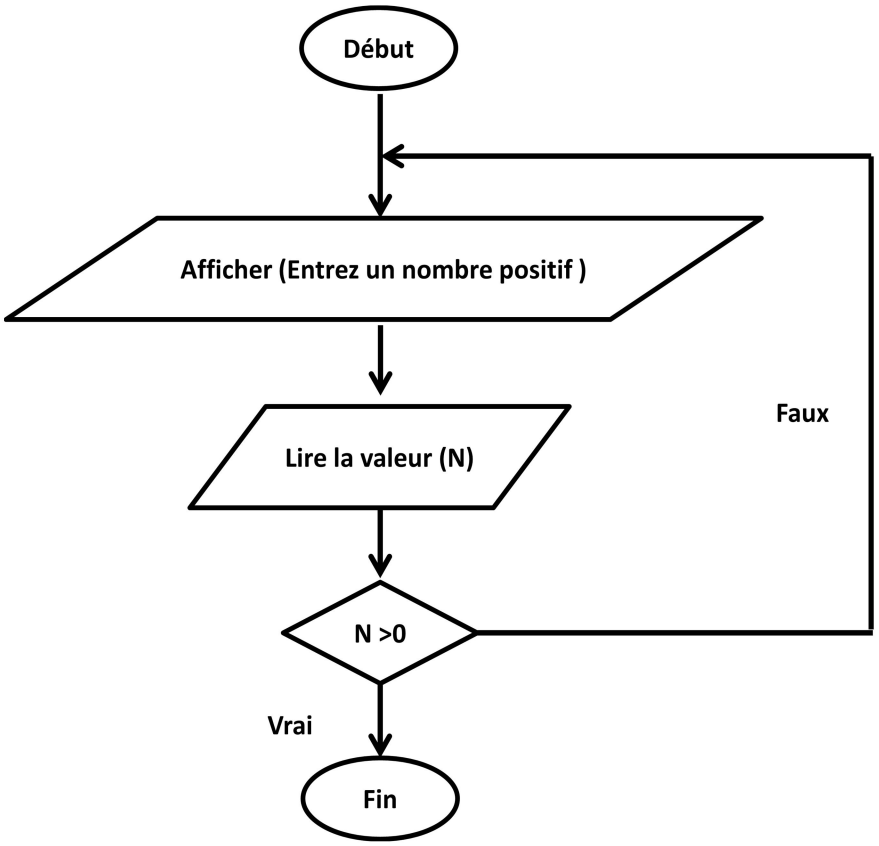


Figure III.9 : Exemple d'utilisation de la boucle (WHILE).

- **Programme**

```
a = input('Entrez un nombre positif : ');
while a<0
a = input('Entrez un nombre positif : ');
end
disp(a)
```

L'exécution affiche les informations suivantes :

```
Entrez un nombre positif : -1
Entrez un nombre positif : -2
Entrez un nombre positif :  3
    3
```

Chapitre IV :	Exemples d'application

IV.1. Introduction

Les connaissances de programmation actualisées dans le premier chapitre et approfondies dans le deuxième et le troisième chapitre sont appliquées dans le présent chapitre sur des exemples d'application liés à la spécialité «STRUCTURES» : de résistances des matériaux, de calcul des structures, de béton armé et de charpente métallique.

IV.2. Description des exemples traités

Les exemples a traiter dans les séances des travaux pratiques sont organisés dans l'ordre des idées formulées dans les chapitres précédents. Le tableau IV.1 regroupe les différents exemples :

Tableau IV.1 : Exemples traités [intitulés et application appliquée].

N° :	Intitulé	Application appliquée
TP1	Opérations sur les scalaires	Résistances des matériaux
TP2	Résolution des systèmes linéaires	Calcul des structures
TP3	Opérations sur les polynômes	Calcul des structures
TP4	Utilisation des User-Fonctions	Calcul des structures
TP5	Programmation structurée par l'instruction **IF**	Béton armé
TP6	Programmation structurée par l'instruction **SWITCH**	Charpente métallique
TP7	Programmation structurée par l'instruction **FOR**	Calcul des structures
TP8	Programmation structurée par l'instruction **WHILE**	Calcul des structures

Pour chaque exemple ; l'énoncé du travail pratique est suivi par un listing commenté des différentes étapes, des rappels de calcul théorique sont regroupés dans l'annexe 2.

En raison de la complexité de certain exemples ; l'organigramme et l'algorithme sont ainsi ajoutés ; les exemples concernés sont :

- Programmation modulaire par l'instruction IF ;
- Programmation modulaire par l'instruction SWITCH ;
- Programmation modulaire par l'instruction FOR ;
- Programmation modulaire par l'instruction WHILE.

IV.2.1. Exemple d'application n° 01

Pour l'exemple n° 01 ; intitulé « Opérations sur les scalaires » appliqué au calcul des structures ; l'énoncé et le listing commenté du programme sont les suivants :

A. Enoncé

Soit la poutre isostatique présentée sur la figure IV.1, sous charge uniformément répartie (**q** = 100N/m) de longueur (**L** = 4m) de module de Young (**E** = 20GPa) et de section rectangulaire (**b** = 0.30m et **h** = 0.45m).

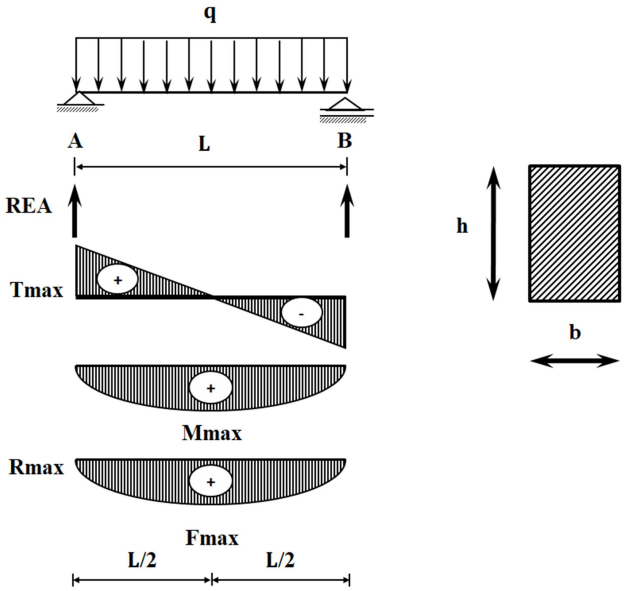


Figure IV.1 : Exemple n° 01 [géométrie, chargement et résultats].

Calculer par le langage MATLAB les valeurs :

1) Du moment d'inertie ;
2) Des réactions aux appuis ;
3) Des sollicitations maximales : l'effort tranchant et le moment fléchissant ;
4) Des déformations maximales : la rotation et la flèche.

B. Listing du programme

```
L = 4
b = 0.30
```

```
h = 0.45
q = 100
```

```
E = 20*(10^9)
```

Déclaration des données : longueur : **L** = 4m, section rectangulaire : **b** = 0.30m et **h** = 0.45m, CUR : **q** = 100N/m et module de Young : **E** = 20 10^9N/m^2.

```
INE = b*(h^3)/12
```

Calcul du moment d'inertie : INE = b h^3 /12 = 2.2781×10^{-3} m^4.

```
REA = q*L/2
```

Calcul des réactions aux appuis : REA = q L /2 = 200N.

Tmax = q*L/2

Calcul de l'effort tranchant maximal : Tmax = q L/2 = 200N.

Mmax = q*(L^2)/8

Calcul du moment fléchissant maximal : Mmax = q L^2 / 8 = 200N.m.

Rmax = q*(L^3)/(24*E*INE)

Calcul de la rotation maximale : Rmax = q L^3 / 24EI = 5.8528×10^{-6} rad.

Fmax = 5*q*(L^4)/(384*E*INE)

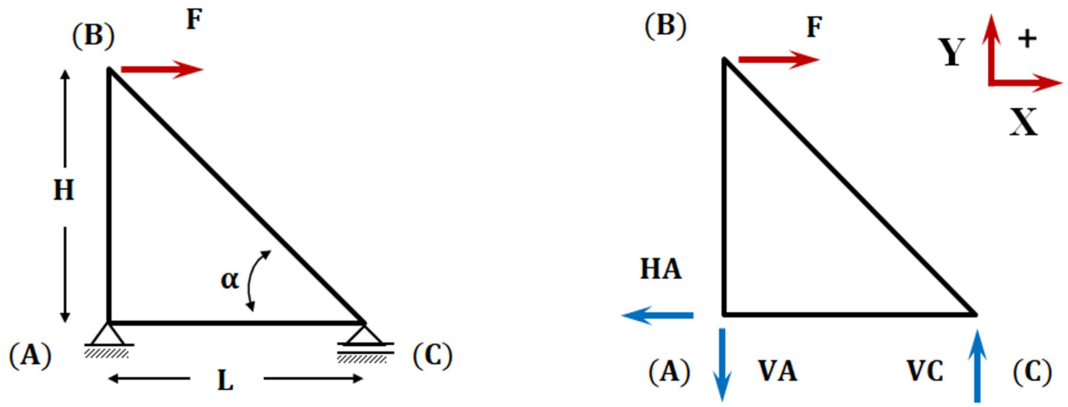
Calcul de la flèche maximale : Fmax = 5 q L^4 / 384EI = 7.3160×10^{-6} m.

IV.2.2. Exemple d'application n° 02

Pour l'exemple n° 02 ; intitulé « Résolution des systèmes linéaires » appliqué au calcul des structures ; l'énoncé et le listing commenté du programme sont les suivants :

A. Enoncé

Pour la structure en treillis présentée sur la figure IV.2 ; on demande d'évaluer les efforts dans l'ensemble des barres par la méthode des nœuds.



(a) **(b)**

Figure IV.2 : Exemple n° 02 [(a) géométrie et chargement, (b) réactions et efforts].

Les étapes à suivre sont :

- **Calcul du sinus et cosinus**

$sin(\alpha) = H/\sqrt{H^2 + L^2}$ et $\cos(\alpha) = L/\sqrt{H^2 + L^2}$

- **Calcul des réactions des appuis**

$$\sum F_X = 0 \rightarrow HA - F = 0 \rightarrow \quad HA = F$$

$$\sum F_Y = 0 \rightarrow VC - VA = 0 \rightarrow VA = VC$$

$$\sum M_A = 0 \rightarrow L.VC - H.F = 0 \rightarrow VC = F\,H/L$$

- **Formulation du système d'équations**
  - *Equilibre au nœud (B)*

$$\sum F_X = 0 \rightarrow F + N_{BC}\cos(\alpha) = 0 \rightarrow \quad N_{BC}\cos(\alpha) = -F$$

$$\sum F_Y = 0 \rightarrow N_{AB} + N_{BC}\sin(\alpha) = 0 \rightarrow \quad N_{BC}\sin(\alpha) + N_{AB} = 0$$

  - *Equilibre au nœud (C)*

$$\sum F_X = 0 \rightarrow -N_{AC} - N_{BC}\cos(\alpha) = 0 \rightarrow N_{BC}\cos(\alpha) + N_{AC} = 0$$

  - *Le système d'équations est*

$$\begin{cases} N_{BC}\cos(\alpha) = -F \\ N_{BC}\sin(\alpha) + N_{AB} = 0 \\ N_{BC}\cos(\alpha) + N_{AC} = 0 \end{cases} \Leftrightarrow \begin{bmatrix} \cos(\alpha) & 0 & 0 \\ \sin(\alpha) & 1 & 0 \\ \cos(\alpha) & 0 & 1 \end{bmatrix} \begin{Bmatrix} N_{BC} \\ N_{AB} \\ N_{AC} \end{Bmatrix} = \begin{Bmatrix} -F \\ 0 \\ 0 \end{Bmatrix}$$

- **Résolution du système d'équations**

$$\{\boldsymbol{VN}\} = \begin{Bmatrix} \boldsymbol{N_{BC}} \\ \boldsymbol{N_{AB}} \\ \boldsymbol{N_{AC}} \end{Bmatrix} = \boldsymbol{Inv} \begin{bmatrix} \boldsymbol{\cos(\alpha)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\sin(\alpha)} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{\cos(\alpha)} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \begin{Bmatrix} \boldsymbol{-F} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{Bmatrix} \Leftrightarrow \{\boldsymbol{VN}\} = \boldsymbol{Inv}[\boldsymbol{MC}] \times \{\boldsymbol{VF}\}$$

B. Listing du programme

```
H = 2
L = 2
F = 500
```

Déclaration des données : hauteur : **H** = 2m, longueur : **L** = 2m et force : **F** = 500N.

```
S = H/sqrt(H^2+L^2)
C = L/sqrt(H^2+L^2)
```

Calcul du **sinus** et **cosinus**.

```
HA = F
VC = F*H/L
VA = CY
```

Calcul des réactions des appuis.

```
VF = [-F ; 0 ; 0]
```

Vecteur des forces.

$$\{\boldsymbol{VF}\} = \begin{Bmatrix} \boldsymbol{-F} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{Bmatrix}$$

MC = [C 0 0 ; S 1 0 ; C 0 1]

Matrice des coefficients.

$$[\boldsymbol{MC}] = \begin{bmatrix} \boldsymbol{\cos(\alpha)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\sin(\alpha)} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{\cos(\alpha)} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

VN = inv(MC)*VF

Inversion et résolution.

NBC = VN(1)
NAB = VN(2)
NAC = VN(3)

Obtention par indexation les efforts dans les barres.

$$\{\boldsymbol{VN}\} = \begin{Bmatrix} \boldsymbol{N_{BC}} \\ \boldsymbol{N_{AB}} \\ \boldsymbol{N_{AC}} \end{Bmatrix}$$

IV.2.3. Exemple d'application n° 03

Pour l'exemple n° 03 ; intitulé « Opérations sur les polynômes » appliqué au calcul des structures ; l'énoncé et le listing commenté du programme sont les suivants :

A. Enoncé

Pour la poutre isostatique présentée auparavant sur la figure IV.1 ; de longueur **L** = 4m, sous charge uniformément répartie **q** = 100N/m ; l'expression de l'effort tranchant $T(x) = -qx + qL/2$. Il est demandé de programmer par MATLAB les points suivants :

- calcul des valeurs de l'effort tranchant le long de la poutre (chaque 1m) ;
- calcul de la valeur maximale de l'effort tranchant ;
- localisation de la position de l'effort tranchant nul ;
- calcul de la valeur de l'effort tranchant pour la position L/4 ;
- traçage de $T(x)$ le long de la poutre.

B. Listing du programme

L = 4
q = 100

Déclaration des données : **L** : longueur de la poutre = 4m, **q** : Charge répartie = 100N/m.

ND = 4

Les informations seront évaluées chaque 1m ; d'où le nombre de division **ND** est égal à 4m/1m = 4.

VX = 0:L/ ND:L

Définir le vecteur des positions $0 \leq x \leq L$ avec un incrément de L / ND.

VCT = [-q q*L/2]

Définir le vecteur qui contient les coefficients du polynôme $T(x) = -qx + qL/2$.

VVT = polyval(VCT,VX)

Calcul du polynôme $T(x)$ pour l'ensemble des positions $0 \leq x \leq L$.

Tmax = max(VVT)

Chercher la valeur maximale du polynôme $T(x)$: $T_{max} = ?$

polyval(VCT,L/4)

Calculer la valeur du polynôme $T(x)$ pour la position L/4 : $T_{(x=L/4)} = ?$

x1 = roots(VCT)

Localiser la position de l'effort tranchant nul c.à.d. les racines du polynôme T(x) : $T_{x=?} = 0$.

```
figure(1)
plot(VX,VVT)
title('Graphe de T(x)')
xlabel('x(m)')
ylabel('T(N)')
```

Tracer le graphe de T(x) le long de L c.à.d. pour l'ensemble des positions $0 \leq x \leq L$.

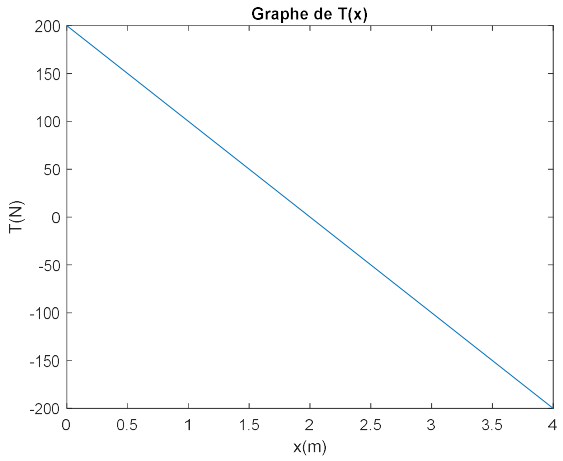


Figure IV.3 : Exemple n° 03 [graphe de T(x) le long de la poutre].

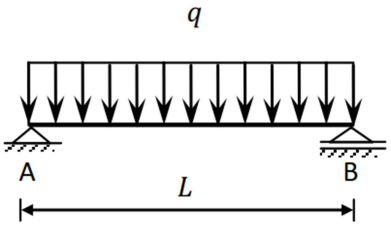
IV.2.4. Exemple d'application n° 04

Pour l'exemple n° 04 ; intitulé « Utilisation des User-Fonctions » appliqué au calcul des structures ; l'énoncé et le listing commenté du programme sont les suivants :

A. Enoncé

Pour la poutre isostatique présentée sur la figure IV.4 ; de longueur (**L**), sous charge uniformément répartie (**q**) ; il est demandé d'écrire une fonction-utilisateur qui prend en charge les points suivants :

- Le calcul des valeurs du moment fléchissant le long de la longueur avec un nombre de division fixé ;
- Le calcul de la valeur maximale du moment fléchissant.



Expression du moment fléchissant $\mathbf{M(x) = q\frac{L}{2}x - q\frac{x^2}{2}}$

Figure IV.4 : Exemple n° 04 [géométrie et chargement].

B. Listing du programme

```
function [VX, VVM, Mmax] = PICUR(L, q, ND)
```

Création de la **FONCTION-UTILISATEUR** nommée **PICUR** (Poutre Isostatique sous Charge Uniformément Répartie), les entrées de cette fonction sont : la longueur **(L)**, la charge uniformément répartie **(q)** et le nombre de division **(ND)**. Les sorties sont le vecteur de positions **(VX)**, le vecteur **(VVM)** des valeurs du moment fléchissant et la valeur maximale du moment fléchissant **(Mmax)**.

```
% Poutre Iso Statique sous CUR
% Entrees : Longueur L(m), CUR q(N) et nombre de division ND
% Sorties : VX : vecteur des positions, VVM : valeurs de M(x)
%           Mmax : valeur maximale de M(x) et graphe de M(x)
```

Quatre lignes de commentaires offrant un identifiant de la fonction, l'identifiant peut être consulté par la syntaxe **help**.

```
>> help PICUR
```

L'instruction **help PICUR** permet d'afficher l'identifiant de la fonction.

L'identifiant de la fonction est présenté dans les lignes suivantes :

Poutre Iso Statique sous CUR
Entrees : Longueur L(m), CUR q(N) et nombre de division ND
Sorties : VX : vecteur des positions, VVM : valeurs de M(x)
Mmax : valeur maximale de M(x) et graphe de M(x)

VX = 0:L/ND:L

Définition du vecteur des positions $0 \leq x \leq$ L avec un incrément de L / ND.

VCM = [-q/2 q*L/2 0]

Définition du vecteur qui contient les coefficients du polynôme $M(x) = -q\frac{x^2}{2} + q\frac{L}{2}x$.

VVM = polyval(VCM,VX)

Calcul du polynôme $M(x)$ pour l'ensemble des positions $0 \leq x \leq$ L.

Mmax = max(VVM)

Chercher la valeur maximale du polynôme $M(x)$: $M_{max} = ?$

end

La description de la fonction est clôturée par la syntaxe **end**.

>> [VX, VVM, Mmax] = PICUR(6, 100, 4)

Pour le cas d'une poutre de longueur **L**= 6m, sous charge uniformément répartie **q** = 100N/m avec un nombre de division **ND** fixé à 4 :

L'utilisation de la FONCTION-UTILISATEUR **PICUR** résulte des informations suivantes :

```
VX =
     0    1.5000    3.0000    4.5000    6.0000
VVM =
     0  337.5000  450.0000  337.5000         0
Mmax =
     450
```

IV.2.5. Exemple d'application n° 05

Pour l'exemple n° 05 ; intitulé « Programmation modulaire par l'instruction IF » appliqué au calcul en béton armé ; l'énoncé et le listing commenté du programme sont les suivants :

A. Enoncé

Etablir le programme du calcul en béton armé des sections rectangulaires sollicitées en traction simple ; la démarche de calcul à suivre est schématisée sur l'organigramme de la figure IV.5 et détaillée sur l'algorithme ci-après :

- **L'organigramme :**

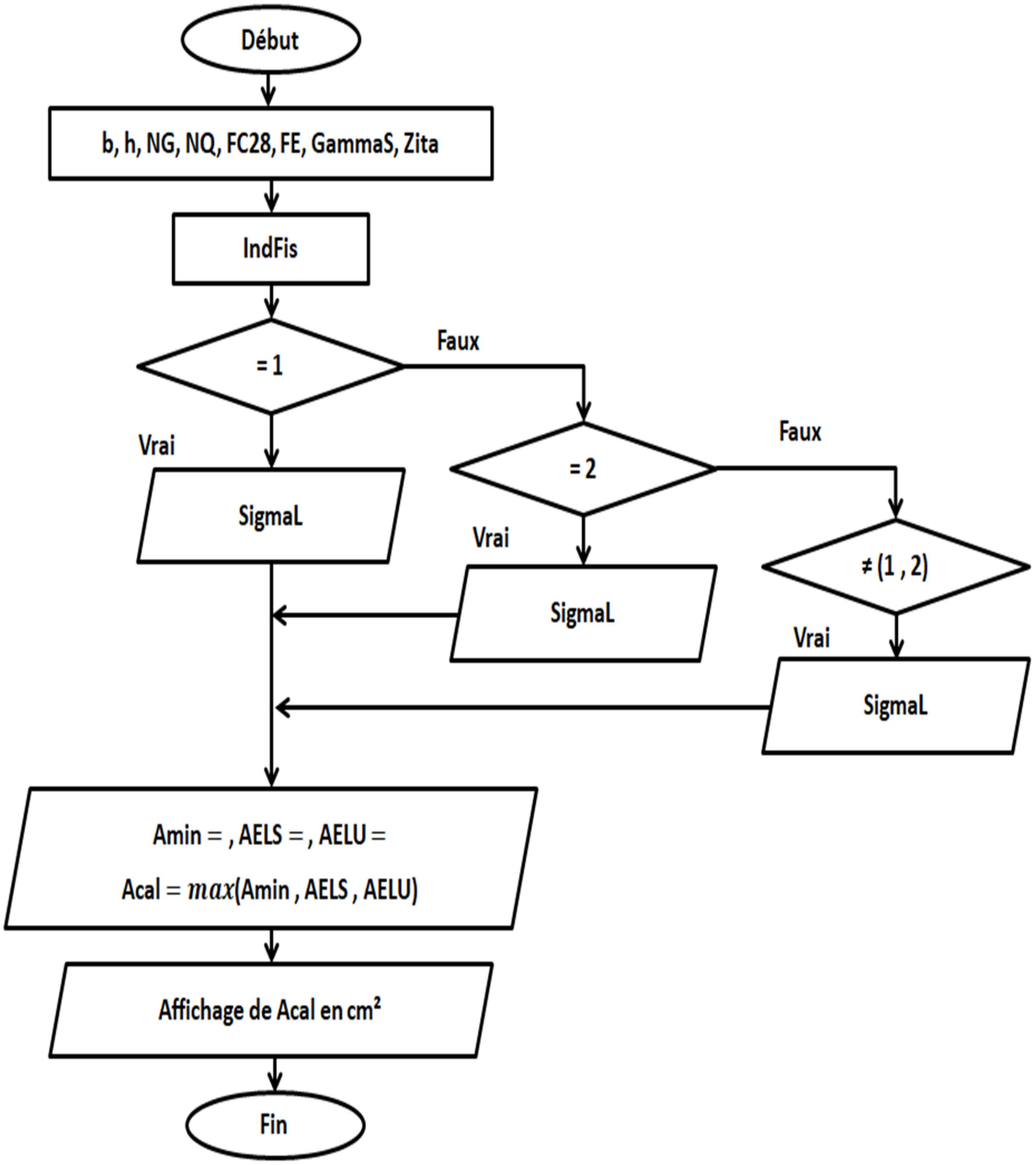


Figure IV.5 : Organigramme de calcul en béton armé des sections rectangulaires sollicitées en traction simple.

- **L'algorithme :**

Données

b : scalaire
h : scalaire
NG : scalaire
NQ : scalaire
FC28 : scalaire
FE : scalaire
GammaS : scalaire
Zita : scalaire

Résultats

ACAL : scalaire

Début

B ← b×h
NELS ← (1.00×NG)+(1.00×NQ)
NELU ← (1.35×NG)+(1.50×NQ)
FT28 ← 0.6+(0.06×FC28)
IndFis = 1
Si IndFis = 1
  Alors
    SigmaL ← FE/GammaS
Si IndFis = 2
  Alors
    SigmaL ← min((2×FE/3, 110×sqrt(Zita×FT28))
Sinon
    SigmaL ← min((FE/2), 90×sqrt(Zita×FT28))
FinSi
AMin ← B×(FT28/FE)
AELS ← NELS/SigmaL
AELU ← NELU/(FE/GammaS)
ACAL ← max(AMin;AELS;AELU)
Afficher ('Ferraillage (cm2) : ')

Fin

B. Listing du programme

Déclaration des données **:** les unités utilisées sont : (m) pour les dimensions, (N) pour les charges et (cm²) pour la section de ferraillage :

```
b = 0.30
h = 0.40
```

b : Largeur et h : Hauteur de la section rectangulaire à calculer en (m).

NG = 134000
NQ = 26000

N_G : Effort des charges permanentes et N_Q : Effort des charges d'exploitation en (N).

FC28 = 25*(10^6)

f_{c28} : Contrainte caractéristique à la compression du béton à l'âge de 28 jours en Pa.

FE = 400*(10^6)

f_e : Contrainte limite d'acier en Pa.

GammaS = 1.00

γ_s : Coefficient de sécurité d'acier.

Zita = 1.6

η : Coefficient relatif à la nuance d'acier (1.6 cas des barres de haute adhérence, 1.0 cas des ronds lisses).

B = b*h

Section de béton : $B = b \times h$

NELS = (1.00*NG)+(1.00*NQ)

Effort combiné à l'état limite de service : $N_{ELS} = 1.00 \times N_G + 1.00 \times N_Q$

NELU = (1.35*NG)+(1.50*NQ)

Effort combiné à l'état limite ultime : $N_{ELU} = 1.35 \times N_G + 1.50 \times N_Q$

FT28 = 0.6+(0.06*FC28)

Contrainte caractéristique à la traction du béton à l'âge de 28 jours : $f_{t28} = 0.60 + 0.06 \times f_{c28}$

IndFis = 1

IndFis : Indicateur de type de fissuration :

- **IndFis** = 1 → Fissuration Peu-Préjudiciable.
- **IndFis** = 2 → Fissuration Préjudiciable.
- **IndFis** ≠ (ni 1 ni 2) → Fissuration Très-Préjudiciable.

```
if IndFis == 1
  SigmaL = FE/GammaS
elseif IndFis == 2
  SigmaL = min((2/3)*FE, 110*sqrt(Zita*FT28))
else
  SigmaL = min((1/2)*FE, 90*sqrt(Zita*FT28))
end
```

$\bar{\sigma}$: Contrainte limite de calcul est relative à la valeur de l'indicateur de fissuration **IndFis** soit IndFis = 1 ou IndFis = 2 ou IndFis ≠ (ni 1 ni 2)

Si : IndFis = 1 → Fis. Peu-Préjudiciable → $\bar{\sigma} = \frac{f_e}{\gamma_s}$

Si non : IndFis = 2 → Fis. Préjudiciable → $\bar{\sigma} = \min\left(\frac{2}{3} f_e \text{ et } 110\sqrt{\eta \times f_{t28}}\right)$

Alors : IndFis ≠ (ni 1 ni 2) → Fis. Très-Préjudiciable → $\bar{\sigma} = \min\left(\frac{1}{2} f_e \text{ et } 90\sqrt{\eta \times f_{t28}}\right)$

Le basculement entre les formules possibles de la contrainte limite est arrêté par la syntaxe **end**

Amin = B*(FT28/FE)

Section minimale de ferraillage : $A_{min} = B \times \frac{f_{t28}}{f_e}$

AELS = NELS/SigmaL

Section de ferraillage à l'état limite de service en fonction de la contrainte $\bar{\sigma}$ calculée : $A_{ELS} = \frac{N_{ELS}}{\bar{\sigma}}$

AELU = NELU/(FE/GammaS)

Section de ferraillage à l'état limite ultime : $A_{ELU} = \frac{N_{ELU}}{f_e} \times \gamma_s$

ACAL = max([Amin;AELS;AELU])

Section de ferraillage calculée : $A_{Cal} = \max(A_{min}, A_{ELS}, A_{ELU})$

disp(['Ferraillage (cm2) : ' num2str(ACAL*10000)])

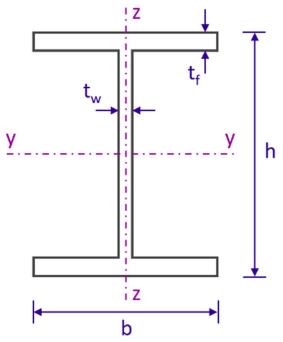
Affichage de la section de ferraillage calculée en centimètre carré.

IV.2.6. Exemple d'application n° 06

A. Enoncé

En employant l'instruction SWITCH du langage MATLAB ; il est demandé d'ordonner les caractéristiques (Dimensions et Poids) des profilés métalliques IPE mentionnées sur le tableau suivant :

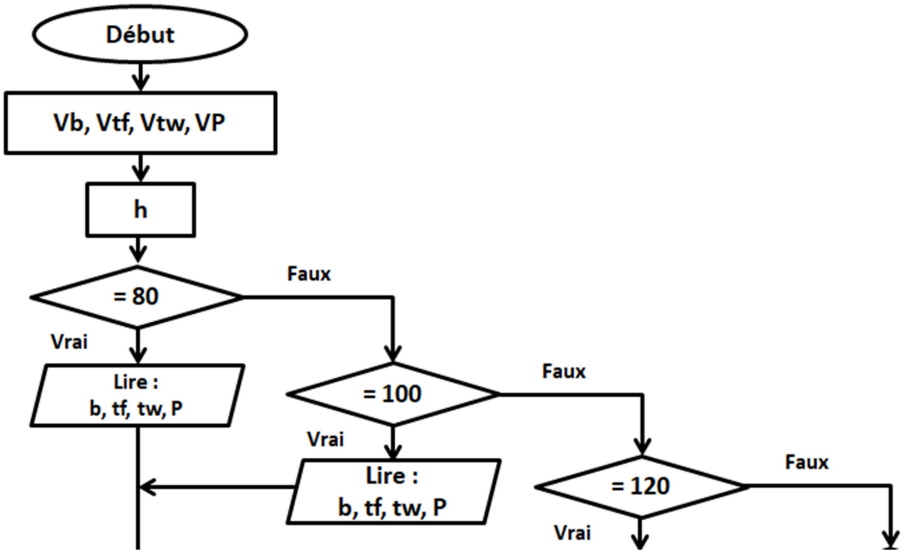
Tableau IV.2 : Caractéristiques géométriques des profilés métalliques IPE80 à IPE140.



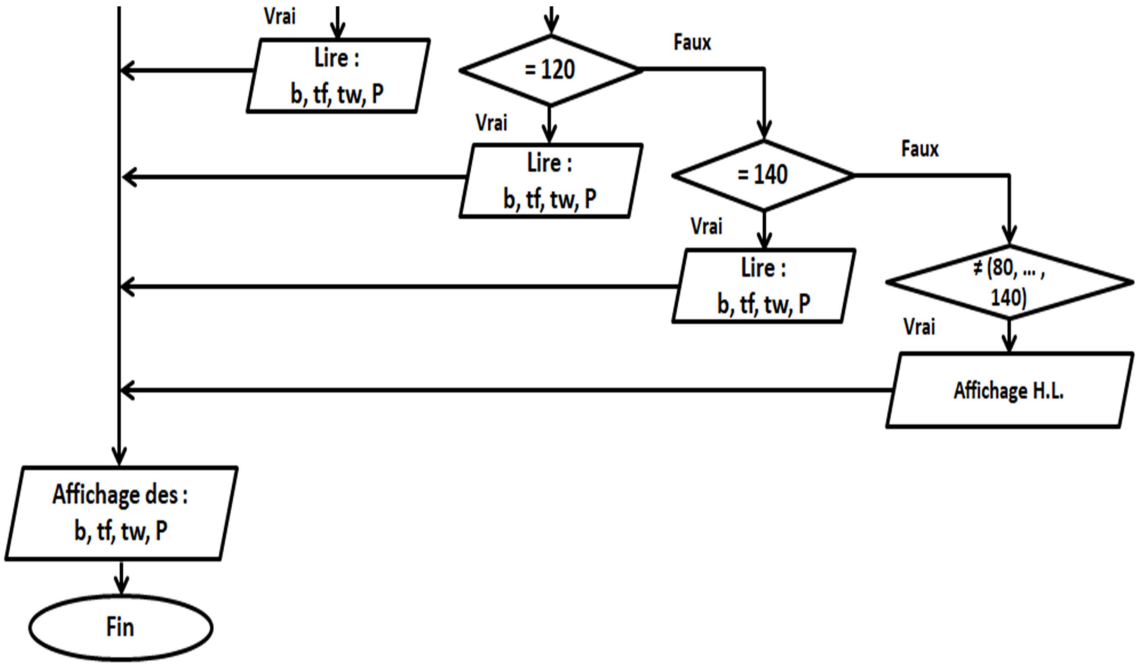
Profilé	Dimensions (mm)				Poids (kg/m)
	h	b	t_f	t_w	
IPE80	80	46	5.2	3.8	6.0
IPE100	100	55	5.7	4.1	8.1
IPE120	120	64	6.3	4.4	10.4
IPE140	140	73	6.9	4.7	12.9

La démarche à suivre est schématisée sur l'organigramme de la figure IV.6 et détaillée sur l'algorithme ci-après :

- **L'organigramme :**



Partie (a)



Partie (b)

Figure IV.6 : Organigramme d'ordonnancement des caractéristiques géométriques des profilés métalliques IPE.

- **L'algorithme**

Données

Vb : vecteur
Vtf : vecteur
Vtw : vecteur
VP : vecteur

Début

Afficher ('Caracteristiques geometriques des Profiles IPE80 a IPE140')
Afficher ('Notations : ')
Afficher ('h : Hauteur totale')
Afficher ('b : Largueur de semelle')
Afficher ('tf : Epaisseur de semelle')
Afficher ('tw : Epaisseur d ame')
Afficher ('P : Poids')
Afficher ('Entrez la hauteur totale H (mm) : ')
Saisir (h)
Cas où h Vaut
80
  Lire Vb(1), Vtf(1), Vtw(1) et VP(1)
100
  Lire Vb(2), Vtf(2), Vtw(2) et VP(2)

```
    120
      Lire Vb(3), Vtf(3), Vtw(1)  et VP(3)
    140
      Lire Vb(4), Vtf(4), Vtw(4)  et VP(4)
    Autre (ni 80 ni 100 ni 120 ni 140)
      Affecter b ← 'H.L.', tf ← 'H.L.', tw ← 'H.L.', P ← 'H.L.'
  FinCas
  Afficher ('Caracteristiques du Profile IPE sont : ')
  Afficher ('h(mm), b(mm), tf(mm), tw(mm), P(kg/m))
Fin
```

B. Listing du programme

```
Vb = [46; 55; 64; 73];
```

Créer le vecteur des valeurs relatives de la dimension (**b**) pour les profilés IPE80 à IPE140.

```
Vtf = [5.2; 5.7; 6.3; 6.9];
```

Créer le vecteur des valeurs relatives de la dimension ($\mathbf{t_f}$) pour les profilés IPE80 à IPE140.

```
Vtw = [3.8; 4.1; 4.4; 4.7];
```

Créer le vecteur des valeurs relatives de la dimension ($\mathbf{t_w}$) pour les profilés IPE80 à IPE140.

```
VP = [6.0; 8.1; 10.4; 12.9];
```

Créer le vecteur des valeurs relatives du poids (**P**) pour les profilés IPE80 à IPE140.

```
disp('Caracteristiques geometriques des Profiles IPE80 a IPE140')
disp('Notations : ')
disp('h  : Hauteur totale')
disp('b  : Largueur de semelle')
disp('tf : Epaisseur de semelle')
disp('tw : Epaisseur d ame')
disp('P  : Poids')
```

Créer les affichages indicatifs des notations utilisées.

```
h = input('Entrez la hauteur totale H (mm) : ') ;
```

Introduction de la hauteur totale (**h**) du profilé sélectionné.

```
hS = num2str(h);
```

Création d'une chaîne de caractères relative au paramètre (**h**) pour des futurs affichages.

```
switch h
```

Basculement de la valeur de la hauteur (**h**) entre les cas possibles : cas 80 ou cas 100 ou cas 120 et finalement 140 c.à.d. $(\mathbf{h}) \in [80, \dots, 140]$, sinon la valeur est hors liste où $(\mathbf{h}) \notin [80, \dots, 140]$.

```
case 80
    b = num2str(Vb(1)); tf = num2str(Vtf(1));
    tw = num2str(Vtw(1)); P = num2str(VP(1));
```

Pour le cas h = 80; lecture des valeurs relatives des dimensions : (**b**) est la composante n° 1 du vecteur **Vb**, ($\mathbf{t_f}$) est la composante n° 1 du vecteur **Vtf**, ($\mathbf{t_w}$) est la composante n° 1 du vecteur **Vtw** et (**P**) est la composante n° 1 du vecteur **VP**. Les valeurs lues sont transformées à des chaînes de caractères pour des futurs affichages.

```
case 100
    b = num2str(Vb(2)); tf = num2str(Vtf(2));
    tw = num2str(Vtw(2)); P = num2str(VP(2));
```

Pour le cas **h** = 100; lecture des valeurs relatives des dimensions : (**b**) est la composante n° 2 du vecteur **Vb**, ($\mathbf{t_f}$) est la composante n° 2 du vecteur **Vtf**, ($\mathbf{t_w}$) est la composante n° 2 du vecteur **Vtw** et (**P**) est la composante n° 2 du vecteur **VP**. Les valeurs lues sont transformées à des chaînes de caractères pour des futurs affichages.

```
case 120
    b = num2str(Vb(3)); tf = num2str(Vtf(3));
    tw = num2str(Vtw(3)); P = num2str(VP(3));
```

Pour le cas h = 120; lecture des valeurs relatives des dimensions : (**b**) est la composante n° 3 du vecteur **Vb**, ($\mathbf{t_f}$) est la composante n° 3 du vecteur **Vtf**, ($\mathbf{t_w}$) est la composante n° 3 du vecteur **Vtw** et (P) est la composante n° 3 du vecteur **VP**. Les valeurs lues sont transformées à des chaînes de caractères pour des futurs affichages.

```
case 140
    b = num2str(Vb(4)); tf = num2str(Vtf(4));
    tw = num2str(Vtw(4)); P = num2str(VP(4));
```

Pour le cas **h** = 140; lecture des valeurs relatives des dimensions : (**b**) est la composante n° 4 du vecteur **Vb**, ($\mathbf{t_f}$) est la composante n° 4 du vecteur **Vtf**, ($\mathbf{t_w}$) est la composante n° 4 du vecteur **Vtw** et (**P**) est la composante n° 4 du vecteur **VP**. Les valeurs lues sont transformées à des chaînes de caractères pour des futurs affichages.

```
otherwise
    b = 'H.L.'; tf = 'H.L.'; tw = 'H.L.'; P = 'H.L.';
```

Pour le cas où la valeur de (**h**) est hors liste (ni 80, …, ni 140) désigné par la syntaxe **otherwise** ; la chaîne de caractères **'H.L.'** est affectée à l'ensemble des dimensions.

```
end
```

Le basculement de la valeur de la hauteur (**h**) entre les cas possibles est arrêté par la syntaxe **end**.

```
disp(['Caracteristiques du Profile IPE' hS ' sont : '])
```

Affichage du le nom du profilé sélectionné crée par union des chaînes de caractères.

```
disp(['h(mm) : ' hS ', b(mm) : ' b ...
   ', tf(mm) : ' tf ', tw(mm) : ' tw ', P(kg/m) : ' P])
```

Affichage des caractéristiques géométriques relatives au profilé sélectionné crées par union des chaînes de caractères. Vu la longueur de cette linge ; l'instruction est divisée en deux morceaux séparés par la syntaxe **...**

L'exécution et l'introduction de la valeur de 120 au paramètre (**h**) ; résultent des lignes suivantes contenant les notations utilisées et les caractéristiques géométriques relatives au profilé IPE120 sélectionné.

```
Caracteristiques geometriques des Profiles IPE80 a IPE140
Notations :
h  : Hauteur totale
b  : Largueur de semelle
tf : Epaisseur de semelle
tw : Epaisseur d ame
P  : Poids
Entrez la hauteur totale H (mm) : 120
Caracteristiques du Profile IPE120 sont :
h(mm) : 120, b(mm) : 64, tf(mm) : 6.3, tw(mm) : 4.4, P(kg/m) : 10.4
```

L'exécution et l'introduction de la valeur de 300 au paramètre (**h**) ; résultent des lignes suivantes contenant les notations utilisées et l'abréviation **H.L** pour l'ensemble des caractéristiques géométriques relatives au profilé hors liste ($IPE300 \notin [IPE80, \dots, IPE140]$

```
Caracteristiques geometriques des Profiles IPE80 a IPE140
Notations :
h  : Hauteur totale
b  : Largueur de semelle
tf : Epaisseur de semelle
tw : Epaisseur d ame
P  : Poids
Entrez la hauteur totale H (mm) : 300
Caracteristiques du Profile IPE300 sont :
h(mm) : 300, b(mm) : H.L., tf(mm) : H.L., tw(mm) : H.L., P(kg/m) : H.L.
```

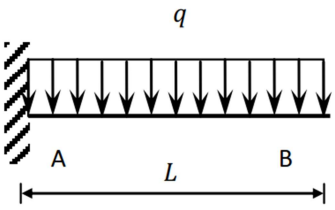
IV.2.7. Exemple d'application n° 07

Pour l'exemple n° 07 ; intitulé « Programmation modulaire par l'instruction FOR » appliqué au calcul des structures ; l'énoncé et le listing commenté du programme sont les suivants :

A. Enoncé

En utilisant l'instruction FOR du langage MATLAB, il est demandé d'évaluer l'expression du moment fléchissant (Eq. 1) relative à la console sous une charge répartie uniformément

présentée sur la figure IV.7. Les valeurs limites du moment fléchissant sont aussi demandées.



$$\boldsymbol{M(x) = -\frac{q}{2}(L - x)^2}$$

Figure IV.7 : Exemple n° 07 [géométrie et chargement].

La démarche de calcul à suivre est schématisée sur l'organigramme de la figure IV.8 et détaillée sur l'algorithme ci-après :

- **L'organigramme**

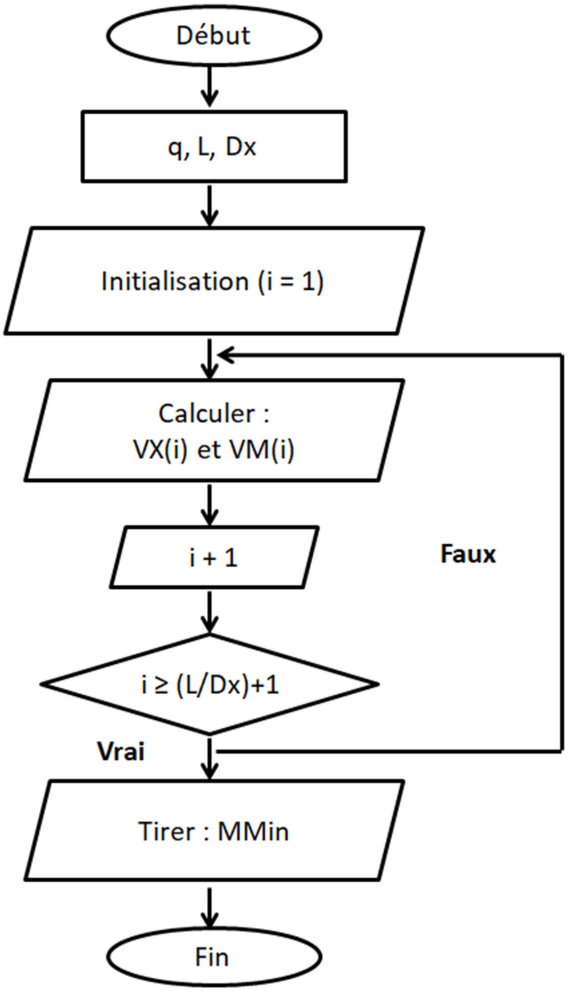


Figure IV.8 : Organigramme de traitement de l'expression du moment fléchissant.

- **L'algorithme :**

Données
q : scalaire
L : scalaire
Dx : scalaire
i : entier (compteur de boucle)
Résultats
VX : vecteur
VM : vecteur
Début
Pour i allant de 1 à (L/Dx)+1 avec un pas de 1
Faire
i
VX ← Dx (i-1)
VM ← -q ((L-VX)^2)/2
FinFaire
Fin

B. Listing du programme

```
q = 100
L = 1.25
Dx = 0.25
```

Introduction des données : **q** = 100N pour la charge uniformément répartie, **L** = 1.25m à la longueur et **Dx** = 0.25m au pas de calcul.

```
for i = 1:(L/Dx)+1
```

Création un compteur de boucle (i) ; allant de la valeur initiale 1 vers la valeur limite 6 car (1.25/0.25)+1 = 6

```
   VX(i) = Dx*(i-1);
```

Calcul de la position **VX** relative à la valeur actuelle du compteur de boucle (i), la position prend la valeur zéro pour $(i = 1)$ car $0.25(1 - 1) = 0$ et prend la valeur **L** pour $(i = 6)$ car $0.25(6 - 1) = 1.25$

```
   VM(i) = -q*((L-VX(i))^2)/2;
```

Calcul de la valeur du moment fléchissant relative à la position actuelle **VX**.

```
end
```

La boule est arrêtée par la syntaxe **end** dès que le compteur de boucle atteint la valeur limite.

```
VX
VM
```

Les valeurs calculés de la position et du moment fléchissant sont stockées respectivement dans les vecteurs **VX** et **VM**.

MMin = min(VM)
MMax = max(VM)

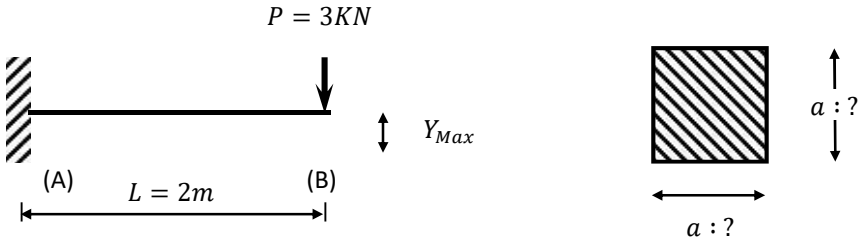
La valeur minimale **MMin** et maximale **MMax** sont évaluées parmi les élément du vecteur du moment fléchissant **VM**.

IV.2.8. Exemple d'application n° 08

Pour l'exemple n° 08 ; intitulé « Programmation modulaire par l'instruction WHILE » appliqué au calcul des structures ; l'énoncé et le listing commenté du programme sont les suivants :

A. Enoncé

En utilisant l'instruction WHILE du langage MATLAB, il est demandé de trouver la section carrée convenable à la console présentée sur la figure IV.9, pour laquelle la flèche admissible est limitée à 0.8cm.



$Y_{Max} = \frac{PL^3}{3EI}$ avec E = 20GPa, $a_{Intial} = 7.5cm$ et $\Delta a = 2.5cm$

Figure IV.9 : Exemple n° 08 [géométrie et chargement].

La démarche de calcul à suivre est schématisée sur l'organigramme de la figure IV.10 et détaillée sur l'algorithme ci-après :

- **L'organigramme**

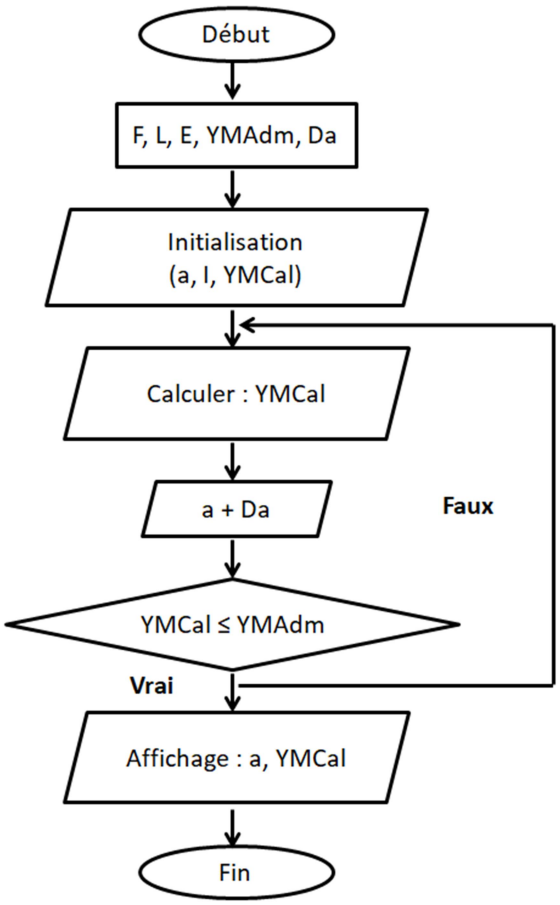


Figure IV.10 : Organigramme de dimensionnement d'une console.

- **L'algorithme :**

Données
P : scalaire
L : scalaire
E : scalaire
YMAdm : scalaire
Da : scalaire
Résultats
a
YMCal
Début
I ← $a^4/12$
YMCal ← P $L^3/3EI$
Tant que YMCal > YMAdm
Faire
I ← $a^4/12$
YMCal ← P $L^3/3EI$

```
    Afficher ('Dimension a (m)      :)
    Afficher (Fleche YMCal (m)   :)
    a ← a + Da
FinFaire
Fin
```

B. Listing du programme

```
P = 3000
L = 2
E = 20*(10^9)
YMAdm = 0.8*(10^-2)
Da = 0.025
```

Introduction des données : $\mathbf{P} = 3000\text{N}$ pour la force ponctuelle, $\mathbf{L} = 2\text{m}$ à la longueur, $\mathbf{E} = 20\ 10^{9}\text{N/m}^2$ au module de Young, $\mathbf{YMAdm} = 0.8\ 10^{-2}\text{m}$ pour la flèche maximale admissible et $\mathbf{Da} = 0.025\text{m}$ à l'incrément sur la dimension **a**.

```
a = 0.075;
```

Introduction de la valeur initiale de la dimension **a**.

```
I = (a^4)/12;
```

Calcul du moment d'inertie relatif à la valeur initiale de la dimension **a**.

```
YMCal = P*(L^3)/(3*E*I);
```

Calcul de flèche maximale relative à la valeur initiale de la dimension **a**.

```
while YMCal > YMAdm
```

Tant que la flèche maximale calculée (relative à valeur actuelle de la dimension **a**) dépasse la valeur de flèche maximale admissible ; la boucle procède au :

```
  I = (a^4)/12;
```

Calcul du moment d'inertie relatif à la valeur actuelle de la dimension **a**.

```
  YMCal = P*(L^3)/(3*E*I);
```

Calcul de flèche maximale relative à la valeur actuelle de la dimension **a**.

```
  disp(['Dimension a (m)       : ' num2str(a)])
```

Affichage de la valeur actuelle de la dimension **a**.

```
  disp([' Fleche YMCal (m)     : ' num2str(YMCal)])
```

Affiche de la valeur actuelle de la flèche maximale calculée (calculée pour la valeur actuelle de la dimension **a**)

```
  a = a + Da;
```

Passage au calcul suivant par l'amplification de la valeur actuelle de la dimension **a** par l'incrément **Da**.

```
end
```

La boule est arrêtée par la syntaxe **end** dès que la condition de la flèche sera inversée c.à.d. **YMCal ≤ YMAdm** où l'inertie est amplifiée pour réduire la flèche maximale calculée par rapport à la valeur maximale admissible.

Liste des références bibliographiques

[1] A. KNIGHT, Basics of MATLAB and beyond, Chapman and Hall, 2000, 202 p.

[2] A. GILAT, MATLAB: An Introduction with Applications, 5th éd., Wiley, 2014, 414 p.

[3] S. ESHKABILOV, Beginning MATLAB and SIMULINK: From Beginner to Pro, 2nd éd., Apress, 2022, 632 p.

[4] M. MOKHTARI, A. MESBAH, Apprendre et maîtriser MATLAB, Springer, 1997, 728 p.

[5] J.T. LAPRESTE, MATLAB : Aide mémoire, Ellipses, 2002, 302 p.

[6] Y. ARIBA, J. CADIEUX, Manuel MATLAB, ICAM de Toulouse, Départements GEI & Mécanique, 2012, 55 p.

[7] J.P. GRENIER, Débuter en Algorithmique avec MATLAB et SCILAB, Ellipses, 2007, 160 p.

[8] M.F. SAAD, Programmer en MATLAB, Collections des ressources informatiques, Eni, 2020, 407p.

[9] S. ATTAWAY, MATLAB : A Practical Introduction to Programming and Problem Solving, 3rd éd., Elsevier, 2013, 560 p.

[10] D.M. ETTER, Engineering problem solving with MATLAB, 2e éd., Prentice Hall, 1997, 329 p.

[11] H. MOORE, MATLAB for Engineers, 5th éd., Pearson, 2018, 688 p.

[12] A. BIRAN, M. BREINER, MATLAB pour l'ingénieur versions 7, 2 éd., Pearson, 2009, 524 p.

[13] J.T. LAPRESTE, C. VIAL, Outils mathématiques pour l'étudiant, l'ingénieur et le chercheur avec MATLAB, Ellipses, 2008, 350 p.

[14] D. ZAOUI, Travaux dirigés de la matière 3LGC F511 : RDM II, UTMB, 2019.

[15] D. ZAOUI, Travaux dirigés de la matière 3LGC F611 : Calcul des structures, UTMB, 2021.

[16] Ministère de l'habitat, D.T.R. - B.C. 2-41 Règles de conception et de calcul des structures en béton armé C.B.A.93, CGS, 1993.

[17] C. HAZARD, F. LELONG, B. QUINZAIN, MEMOTECH : STRUCTURES METALLIQUES, El Educalivre - Casteilla, 1997, 353 p.

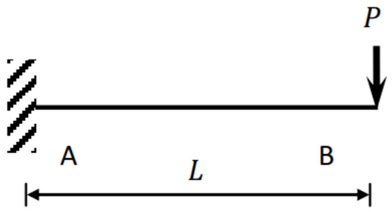
Annexe 1 :	**Fiches des travaux pratiques**
TP1.	Opérations sur les scalaires
TP2.	Résolution des systèmes linéaires
TP3.	Opérations sur les polynômes
TP4.	Utilisation des User-Fonctions
TP5.	Programmation structurée par l'instruction IF
TP6.	Programmation structurée par l'instruction SWITCH
TP7.	Programmation structurée par l'instruction FOR
TP8.	Programmation structurée par l'instruction WHILE

1MGCS M111 : Complément de programmation

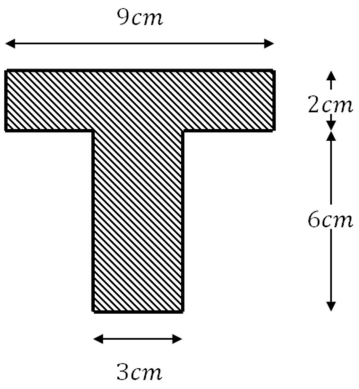
TP1 : Opérations sur les scalaires - application en RDM

Soit la console de longueur (L = 1m) sous une force concentrée (P = 0.51kN) de section en T détaillée ci-après ; pour laquelle on demande de calculer par le langage MATLAB les valeurs de :

1) la valeur limite du moment fléchissant ;
2) la position de l'axe neutre par rapport au centre de la fibre la plus basse ;
3) le moment d'inertie par rapport au centre de gravité ;
4) les contraintes maximales de flexion aux fibres extrêmes.



(a)



(b)

Figure TP1 : Console sous une force concentrée
[(a) Longueur et chargement, (b) Détails de la section].

1MGCS M111 : Complément de programmation

TP2 : Résolution des systèmes linéaires - application en calcul des structures

Pour la poutre en treillis présentée ci-après ; on demande d'évaluer par le langage MATLAB les efforts dans l'ensemble des barres en utilisant la méthode des nœuds :

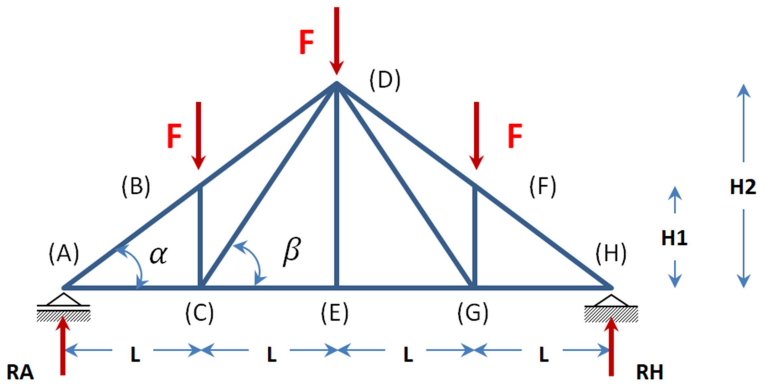


Figure TP2 : Poutre en treillis.

Données :

L = 4m , H1 = 3m , H2 = 6m et F = 8kN

Calculs :

$\sin(\alpha) = H1/\sqrt{H1^2 + L^2}$ et $\cos(\alpha) = L/\sqrt{H1^2 + L^2}$

$\sin(\beta) = H2/\sqrt{H2^2 + L^2}$ et $\cos(\beta) = L/\sqrt{H2^2 + L^2}$

Réactions :

R_A=1.5×F et R_H =1.5×F

Efforts dans les barres :

$F_{AB} \sin(\alpha) = R_A$

$F_{AB} - F_{BD} = 0$

$F_{BC} - F_{CD} \sin(\beta) = 0$

$F_{DE} = 0$

$F_{AB} \cos(\alpha) - F_{AC} = 0$

$F_{BC} = F$

$F_{AC} - F_{CD} \cos(\beta) - F_{CE} = 0$

Par symétrie : $F_{FH} = F_{AB}$, $F_{GH} = F_{AC}$, $F_{FG} = F_{BC}$, $F_{DF} = F_{BD}$, $F_{DG} = F_{CD}$, $F_{EG} = F_{CE}$

1MGCS M111 : Complément de programmation

TP3 : Opérations sur les polynômes - application en calcul des structures

Pour la poutre isostatique présentée ci-après ; on demande par le langage MATLAB :

1) d'évaluer et tracer les polynômes de la rotation (eq. 1 et 3) et de la flèche (eq. 2 et 4) ;
2) d'évaluer les valeurs extrêmes de la rotation et de la flèche.

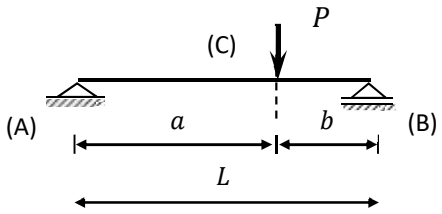


Figure TP3 : Poutre à étudier [longueur et chargement].

Pour les conditions aux limites ;
le système d'équations à résoudre est :

$$\begin{cases} C_2 = 0 \\ C_3 + \dfrac{C_4}{L} = \dfrac{a\,L\,P}{3} \\ -C_1 + C_3 + \dfrac{C_4}{a} = \dfrac{a^2\,P}{3} \\ -C_1 + C_3 = \dfrac{a^2\,P}{2} \end{cases}$$

On donne : Les expression des rotations et des flèches :

Coupe (1) : $0 \le x \le a$

$$\varphi_1(x) = \frac{1}{EI}\left[-\frac{b\,P}{2\,L}x^2 + C_1\right] \quad \text{Eq. 1}$$

$$y_1(x) = \frac{1}{EI}\left[-\frac{b\,P}{6\,L}x^3 + C_1 x + C_2\right] \quad \text{Eq. 2}$$

Coupe (2) : $a \le x \le L$

$$\varphi_2(x) = \frac{1}{EI}\left[\frac{a\,P}{2\,L}x^2 - a\,Px + C_3\right] \quad \text{Eq. 3}$$

$$y_2(x) = \frac{1}{EI}\left[\frac{a\,P}{6\,L}x^3 - \frac{a\,P}{2}x^2 + C_3 x + C_4\right] \quad \text{Eq. 4}$$

Application numérique : L = 3m, a = 2m, P = 6kN, E = 30GN/m^2, Section rectangulaire : h = 0.40m et b = 0.30m

1MGCS M111 : Complément de programmation

TP4 : Utilisation des User-fonctions - application en calcul des structures

Soit la poutre présentée ci-après, pour laquelle les expressions de la rotation (Eq. 1) et de la flèche (Eq. 2) sont :

$$\varphi(x) = \frac{q}{24EI}[4x^3 - 6Lx^2 + L^3] \qquad \text{Eq. 1}$$

$$y(x) = \frac{q}{24EI}[x^4 - 2Lx^3 + L^3x] \qquad \text{Eq. 2}$$

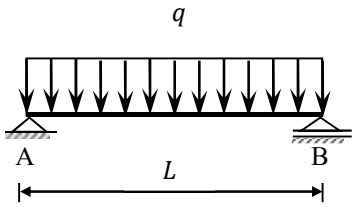


Figure TP4 : Poutre à étudier [longueur et chargement].

On demande d'écrire en le langage MATLAB une fonction-utilisateur qui :

1) évalue et trace le graphe de la rotation le long de la poutre et évalue sa valeur maximale ;
2) évalue et trace le graphe de la flèche le long de la poutre et évalue sa valeur maximale.

Application numérique : L = 2m, q = 1000 N/m, E = 30GN/m^2, h = 0.40m et b = 0.30m

1MGCS M111 : Complément de programmation

TP5 : Programmation structurée par l'instruction IF
- Application en béton armé -

Afin de programmer par le langage MATLAB (l'instruction IF) le calcul des armatures longitudinales des sections rectangulaires soumises à la compression centrée selon le code BAEL 91/99 ; il est demandé :

1) d'établir l'algorithme et l'organigramme nécessaires ;
2) d'établir le programme demandé ;
3) de vérifier pas à pas les résultats obtenus.

Rappel sur les étapes de calcul :

a) Données :

l_0
N_G et N_Q
Béton : f_{c28} et γ_b
Acier : f_e et γ_s
Section rectangulaire : a et b

b) Calculs préliminaires :

$l_f = 0.70 \times l_0$
$N_U = 1.35 \times N_G + 1.50 \times N_Q$

c) Calculs en béton armé :

$B = a \times b$
$B_r = (a - 2cm) \times (b - 2cm)$
$U = 2(a + b)$
$\lambda = 2\sqrt{3} l_f / \min(a, b) = 32.3316 \leq 50 \rightarrow$

$\alpha = \frac{0,85}{1+0.2\left(\frac{\lambda}{35}\right)^2}$ si $\lambda \leq 50$ si non $\alpha = 0.6\left(\frac{50}{\lambda}\right)^2$

$A_{TH} = \left(\frac{N_U}{\alpha} - \frac{B_r f_{c2}}{0,9\gamma_b}\right)\frac{\gamma_s}{f_e}$
$A_{Min1} = 4cm^2/U(1m)$
$A_{Min2} = 0.2\% B$
$A_{Min} = \max(A_{Min1} \text{ et } A_{Min2})$
$A_{RET} = \max(A_{TH} \text{ et } A_{Min})$

1MGCS M111 : Complément de programmation

TP6 : Programmation structurée par l'instruction SWITCH - Application en charpente métallique -

En utilisant l'instruction SWITCH du langage MATLAB, ordonner les caractéristiques géométriques des profilés métalliques IPE mentionnées sur la figure et le tableau suivant :

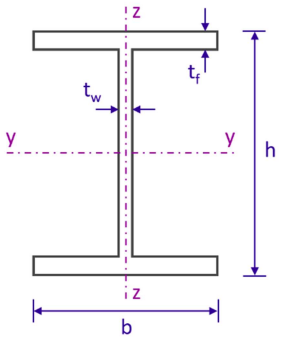


Figure TP6 : Profilés métalliques IPE [dimensions et axes].

Tableau TP6 : Caractéristiques géométriques des profilés métalliques IPE80 à IPE200.

Profilé	Surface (cm^2)	Moments d'inertie (cm^4)		Modules de résistance élastique (cm^3)		Rayons de giration (cm)	
	A	I_y	I_z	W_y	W_z	i_y	i_z
IPE80	7.64	80.1	8.49	20.0	3.69	3.24	1.05
IPE100	10.3	171	15.9	34.2	5.79	4.07	1.24
IPE120	13.2	318	27.7	53.0	8.65	4.90	1.45
IPE140	16.4	541	44.9	77.3	12.3	5.74	1.65
IPE160	20.1	869	68.3	109	16.7	6.58	1.84
IPE180	23.9	1317	101	146	22.2	7.42	2.05
IPE200	28.5	1943	142	194	28.5	8.26	2.24

1MGCS M111 : Complément de programmation

TP7 : Programmation structurée par l'instruction FOR - application en calcul des structures

En utilisant l'instruction FOR du langage MATLAB, il est demandé d'évaluer et de tracer les expressions des lignes d'influence des réactions (Eq. 1 et Eq. 2), de l'effort tranchant au point (C) (Eq. 3) et du moment fléchissant au point (C) (Eq. 4) relatives de la poutre isostatique présentée sur la figure ci-après :

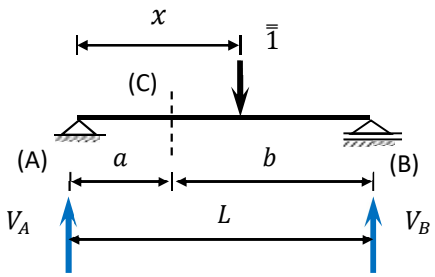


Figure TP7 : Lignes d'influence à étudier.

Les expressions des réactions, de l'effort tranchant T_C et du moment fléchissant M_C sont :

- **Réactions**

$$V_A = 1 - \frac{x}{L} \qquad \text{Eq. 1}$$

$$V_B = \frac{x}{L} \qquad \text{Eq. 2}$$

- **Effort tranchant au point (C)**

$$T_C = \begin{cases} -\frac{x}{L} & 0 \le x \le a \\ 1 - \frac{x}{L} & a \le x \le L \end{cases} \qquad \text{Eq. 3}$$

- **Moment fléchissant au point (C)**

$$M_C = \begin{cases} b\frac{x}{L} & 0 \le x \le a \\ a\left(1 - \frac{x}{L}\right) & a \le x \le L \end{cases} \qquad \text{Eq. 4}$$

Application numérique : L = 6m et a = 1 .5m

1MGCS M111 : Complément de programmation

TP8 : Programmation structurée par l'instruction WHILE - application en calcul des structures

En employant l'instruction WHILE du langage MATLAB, il est demandé de dimensionner la section carrée convenable à la console présentée ci-après, pour laquelle les contraintes admissibles sont limitées aux $\bar{\tau}$ pour le cisaillement, $\overline{\sigma_c}$ pour la compression et $\bar{\sigma}_t$ pour la traction.

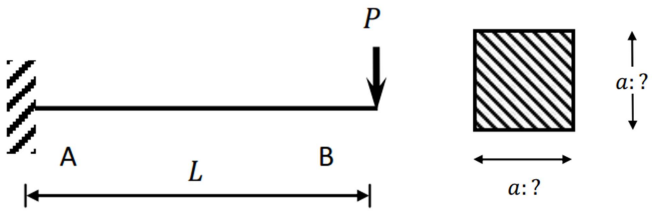


Figure TP8 : Console sous force concentrée.

Application numérique :

L = 2m, P = 1000N/m, $a_{Intial} = 5cm$ et $\Delta a = 1cm$

$\bar{\tau} = 1.25\text{MPa}$, $\bar{\sigma}_t = 5\text{MPa}$ et $\overline{\sigma_c} = 10\text{MPa}$

Annexe 2 :	Rappels des calculs théoriques
TP1.	Opérations sur les scalaires
TP2.	Résolution des systèmes linéaires
TP3.	Opérations sur les polynômes
TP4.	Utilisation des **User-Fonctions**
TP5.	Programmation structurée par l'instruction **IF**
TP6.	Programmation structurée par l'instruction **SWITCH**
TP7.	Programmation structurée par l'instruction **FOR**
TP8.	Programmation structurée par l'instruction **WHILE**

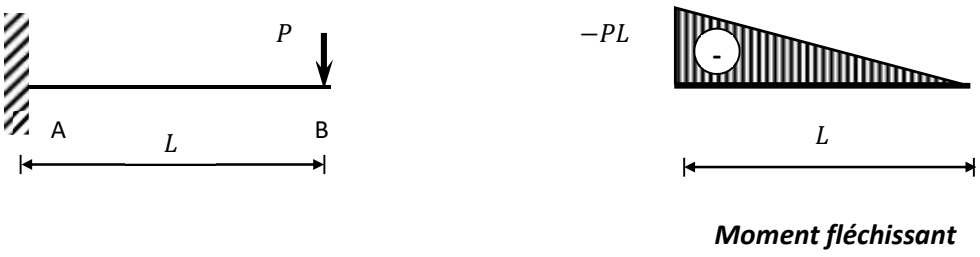
1MGCS M111 : Complément de programmation

TP1 : Rappels des calculs théoriques

Pour la console de longueur (L = 1m) sous une force concentrée (P = 0.51kN) de section en T détaillée ci-après ; on calcule :

1) la valeur limite du moment fléchissant ;
2) la position de l'axe neutre par rapport au centre de la fibre la plus basse ;
3) le moment d'inertie par rapport au centre de gravité ;
4) les contraintes maximales de flexion aux fibres extrêmes.

1) Valeur limite du moment fléchissant:



Moment fléchissant

La valeur limite du moment fléchissant est : $M^{-}_{Max} = -PL = -0.51KN \times 1m = -0.51KN.m$

2) Calcul de la position de l'axe neutre :

La section en T est décomposée en deux sections rectangulaires schématisées ci-après, les calculs sont regroupés dans le tableau suivant :

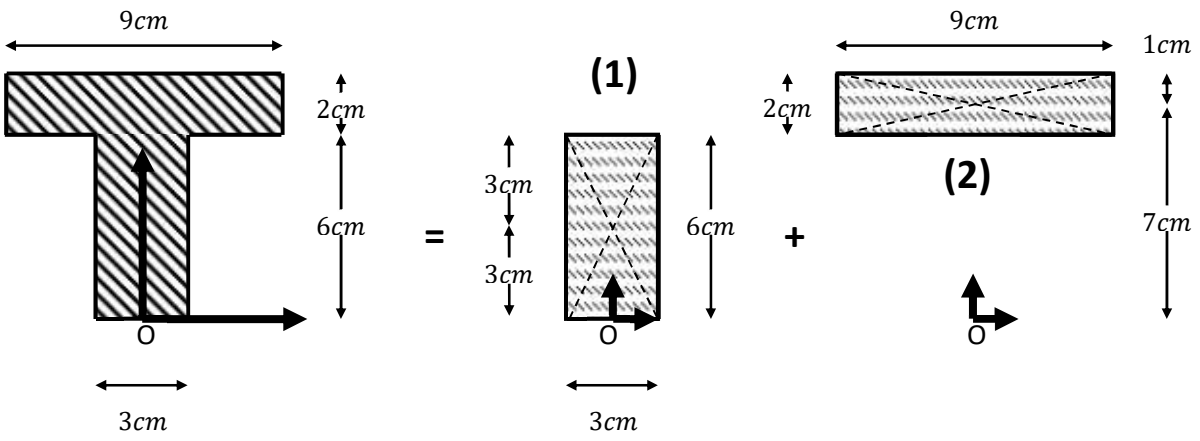
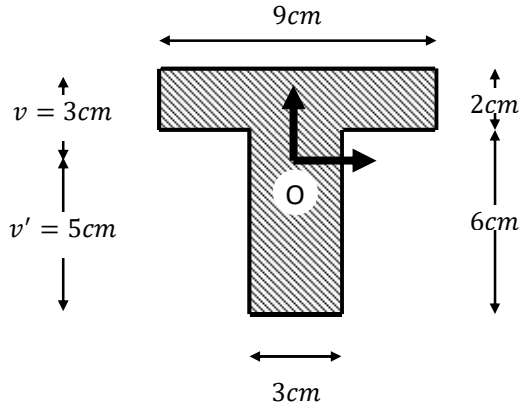


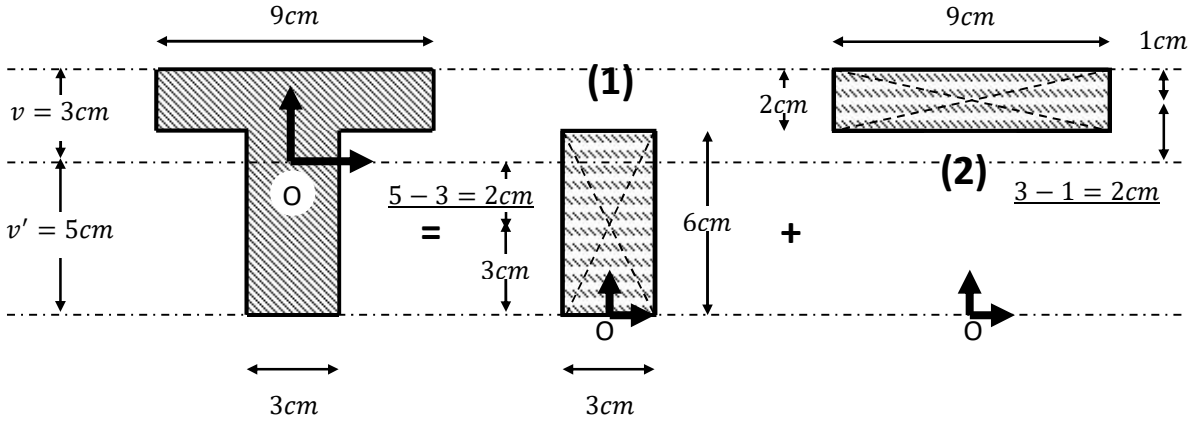
Tableau TP1 : Caractéristiques géométriques de la section en T décomposée.

Partie	Section A (cm²)	Y_0 (cm)	$A.Y_0$ (cm^3)
(1)	3x6 = 18	3	18x3 = 54
(2)	9x2 = 18	7	18x7 = 126
	$\sum$ = **36**		$\sum$ = **180**

$Y_G = \frac{\sum A_i Y_i}{\sum A_i} = \frac{A_1 Y_1 + A_2 Y_2}{A_1 + A_2} = \frac{180}{36} = 5cm$ D'où : $v' = 5cm$ et $v = 8 - 5 = 3cm$



3) Calcul du moment d'inertie :



Les moments partiels d'inertie par rapport à l'axe neutre sont :

$$I_1 = I_{1/01} + {d_1}^2 \times A_1 = \frac{3 \times 6^3}{12} + 2^2 \times 18 = 126\text{cm}^4$$

$$I_2 = I_{2/02} + {d_2}^2 \times A_2 = \frac{9 \times 2^3}{12} + 2^2 \times 18 = 78\text{cm}^4$$

D'où le moment total d'inertie est $I = I_1 + I_2 = 126 + 78 = 204\text{cm}^4$

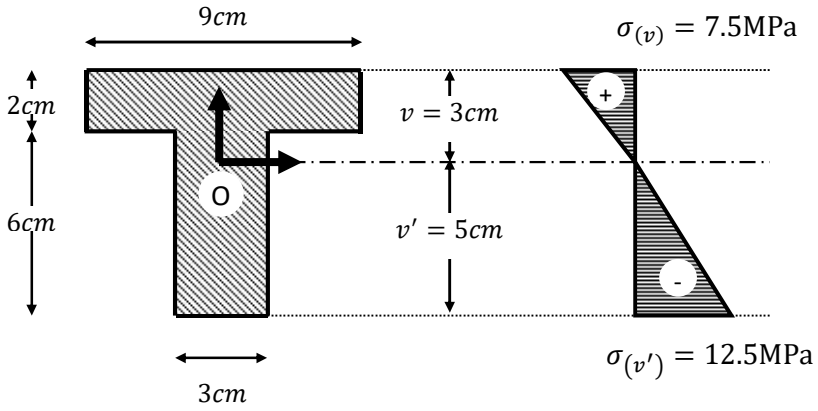
4) Les contraintes maximales de flexion aux fibres extrêmes sont :

A la fibre supérieure :

$$\sigma_{(v)} = -\frac{M_{Max}}{I}(v) = -\frac{-0.51.10^3}{204.(10^{-2})^4}(3.10^{-2}) = +7.5\ 10^6 \frac{\text{N}}{\text{m}^2} = +7.5\text{MPa}$$

A la fibre inférieure :

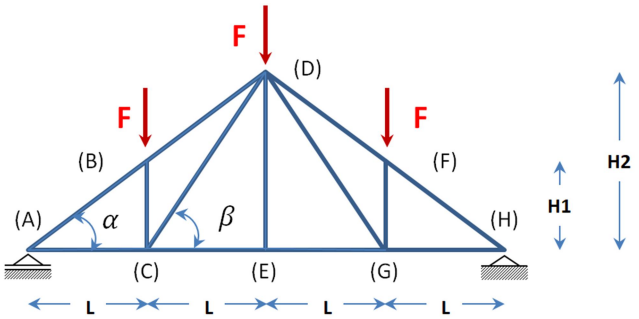
$$\sigma_{(v')} = -\frac{M_{Max}}{I}(-v') = -\frac{-0.51.10^3}{204.(10^{-2})^4}(-5.10^{-2}) = -12.5\ 10^6 \frac{\text{N}}{\text{m}^2} = -12.5\text{MPa}$$



1MGCS M111 : Complément de programmation

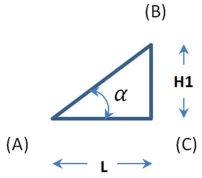
TP2 : Rappels des calculs théoriques

Pour la poutre en treillis présentée ci-après ; on demande d'évaluer les efforts dans l'ensemble des barres en utilisant la méthode des nœuds :



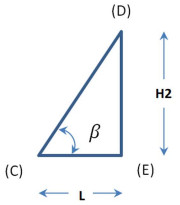
Données : L = 4m , H1 = 3m , H2 = 6m et F = 8kN

Géométrie :



$\sin(\alpha) = H1/\sqrt{H1^2 + L^2} = 3/\sqrt{3^2 + 4^2} = 3/5$

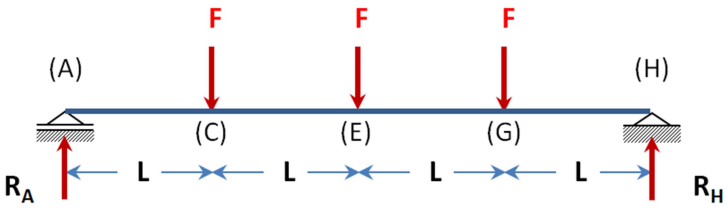
$\cos(\alpha) = L/\sqrt{H1^2 + L^2} = 4/\sqrt{3^2 + 4^2} = 4/5$



$\sin(\beta) = H2/\sqrt{H2^2 + L^2} = 6/\sqrt{4^2 + 6^2} = 6/\sqrt{52}$

$\cos(\beta) = L/\sqrt{H2^2 + L^2} = 4/\sqrt{4^2 + 6^2} = 4/\sqrt{52}$

Calcul des réactions : Par symétrie



$R_A = R_H = 3F/2 = 1.5 \times F = 1.5 \times 8 = 12kN$

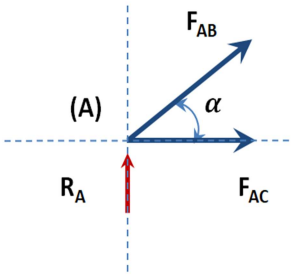
Efforts dans les barres ;

Au nœud (A)

$\sum F_V = 0$

$R_A + F_{AB} \sin(\alpha) = 0$

$F_{AB} = - R_A / \sin(\alpha) = -12/(3/5) = -20kN$



$\sum F_H = 0$

$F_{AB} \cos(\alpha) + F_{AC} = 0$

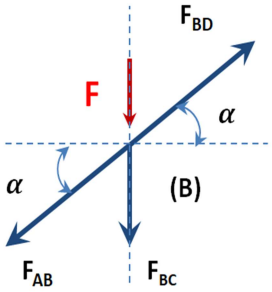
$F_{AC} = - F_{AB} \cos(\alpha) = -(-20)(4/5) = 16kN$

Au nœud (B)

$\sum F_H = 0$

$F_{BD} \cos(\alpha) - F_{AB} \cos(\alpha) = 0$

$F_{BD} = F_{AB} = -20kN$



$\sum F_V = 0$

$- F_{BC} - F + F_{BD} \sin(\alpha) - F_{AB} \sin(\alpha) = 0$

$F_{BC} = - F + F_{BD} \sin(\alpha) - F_{AB} \sin(\alpha)$

$F_{BC} = - 8 + (-20)(3/5) - (-20)(3/5) = -8kN$

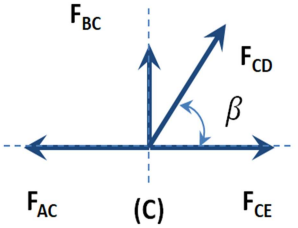
Au nœud (C)

$\sum F_V = 0$

$F_{BC} + F_{CD} \sin(\beta) = 0$

$F_{CD} = - F_{BC} / \sin(\beta) = 0$

$F_{BC} = - (-8) / (6/\sqrt{52}) = 4\sqrt{52}/3kN$



$\sum F_H = 0$

$F_{CE} + F_{CD} \cos(\beta) - F_{AC} = 0$

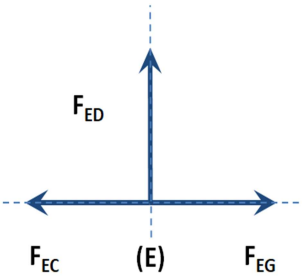
$F_{CE} = F_{AC} - F_{CD} \cos(\beta)$

$F_{CE} = 16 - (4\sqrt{52}/3) \times (4/\sqrt{52}) = 32/3kN$

Au nœud (E)

$\sum F_V = 0$

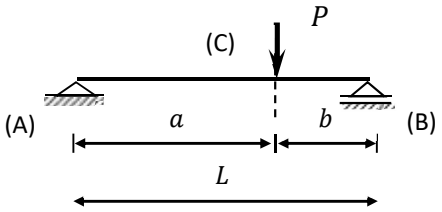
$F_{DE} = 0kN$



1MGCS M111 : Complément de programmation

TP3 : Rappels des calculs théoriques

Soit la poutre présentée ci-après, pour laquelle on calcule les expressions de la rotation et de la flèche :



Réactions aux appuis :

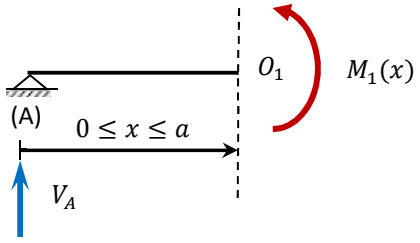
$$\sum F_V = 0 \Leftrightarrow V_A + V_B - P = 0 \Leftrightarrow V_A + V_B = P$$

$$\sum M_{/A} = 0 \Leftrightarrow -L\,V_B + a\,P = 0 \Leftrightarrow V_B = \frac{a\,P}{L}$$

$$\sum M_{/B} = 0 \Leftrightarrow L\,V_A - b\,P = 0 \Leftrightarrow V_A = \frac{b\,P}{L}$$

Expressions de la rotation et la flèche :

Coupe (1) : $0 \leq x \leq a$

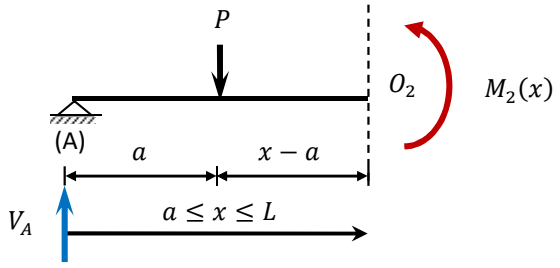


$$M_1(x) = V_A x = \frac{b\,P}{L}x \Leftrightarrow EI\frac{d^2y_1}{dx^2} = -M_1(x) = -\frac{b\,P}{L}x$$

$$\Leftrightarrow EI\varphi_1(x) = EI\frac{dy_1}{dx} = \int M_1(x)\,dx = \int\left(-\frac{b\,P}{L}x\right)dx \Leftrightarrow EI\varphi_1(x) = -\frac{b\,P}{2L}x^2 + C_1$$

$$\Leftrightarrow EIy_1(x) = \int\left(-\frac{b\,P}{2L}x^2 + C_1\right)dx \Leftrightarrow EIy_1(x) = -\frac{b\,P}{6L}x^3 + C_1x + C_2$$

Coupe (2) : $a \le x \le L$



$$M_1(x) = \frac{b\,P}{L}x - P(x-a) = \frac{b\,P}{L}x - P(x-a) = -\frac{a\,P}{L}x + a\,P$$

$$\Leftrightarrow EI\frac{d^2y_2}{dx^2} = -M_2(x) = \frac{a\,P}{L}x - a\,P$$

$$\Leftrightarrow EI\varphi_2(x) = EI\frac{dy_2}{dx} = \int M_2(x)\,dx = \int\left(\frac{a\,P}{L}x - Pa\right)dx \Leftrightarrow EI\varphi_2(x) = \frac{a\,P}{2L}x^2 - a\,P\,x + C_3$$

$$\Leftrightarrow EIy_2 = \int\left(\frac{a\,P}{2L}x^2 - Pax + C_3\right)dx \Leftrightarrow EIy_2(x) = \frac{a\,P}{6L}x^3 - \frac{a\,P}{2}x^2 + C_3x + C_4$$

Au point (A) $\quad x = 0 : y_1(0) = 0 \Leftrightarrow -\frac{b\,P}{6L}0^3 + C_1 0 + C_2 = 0 \Leftrightarrow C_2 = 0$

Au point (B) $\quad x = L : y_2(L) = 0 \Leftrightarrow C_3 + \frac{C_4}{L} = \frac{a\,L\,P}{3}$

Au point (C) $\quad x = a : y_1(a) = y_2(a)$

$$\Leftrightarrow -\frac{b\,P}{6L}a^3 + C_1 a + C_2 = \frac{a\,P}{6L}a^3 - \frac{a\,P}{2}a^2 + C_3 a + C_4 \Leftrightarrow -C_1 + C_3 + \frac{C_4}{a} = \frac{a^2\,P}{3}$$

Au point (C) $\quad x = a : \varphi_1(a) = \varphi_2(a)$

$$\Leftrightarrow -\frac{b\,P}{2L}x^2 + C_1 = \frac{a\,P}{2L}x^2 - Pax + C_3 \Leftrightarrow -C_1 + C_3 = \frac{a^2\,P}{2}$$

Système d'équations à résoudre :

$$\begin{vmatrix} C_2 = 0 \\ C_3 + \frac{C_4}{L} = \frac{a\,L\,P}{3} \\ -C_1 + C_3 + \frac{C_4}{a} = \frac{a^2\,P}{3} \\ -C_1 + C_3 = \frac{a^2\,P}{2} \end{vmatrix} \Leftrightarrow \begin{vmatrix} C_1 = \frac{a\,P}{6L}[2L^2 + a^2 - 3aL] \\ C_2 = 0 \\ C_3 = \frac{P}{6L}[2aL^2 + a^3] \\ C_4 = -\frac{a^3\,P}{6} \end{vmatrix}$$

D'où les expressions des rotations et des flèches sont :

Coupe (1) : $0 \leq x \leq a$

$$\varphi_1(x) = \frac{1}{EI}\left[-\frac{b\,P}{2L}x^2 + C_1\right]$$

$$y_1(x) = \frac{1}{EI}\left[-\frac{b\,P}{6L}x^3 + C_1 x + C_2\right]$$

Coupe (2) : $a \leq x \leq L$

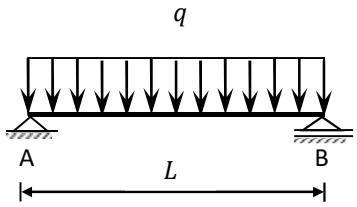
$$\varphi_2(x) = \frac{1}{EI}\left[\frac{a\,P}{2L}x^2 - a\,Px + C_3\right]$$

$$y_2(x) = \frac{1}{EI}\left[\frac{a\,P}{6L}x^3 - \frac{a\,P}{2}x^2 + C_3 x + C_4\right]$$

1MGCS M111 : Complément de programmation

TP4 : Rappels des calculs théoriques

Soit la poutre présentée ci-après, pour laquelle on s'intéresse à évaluer les expressions de la rotation et de la flèche :

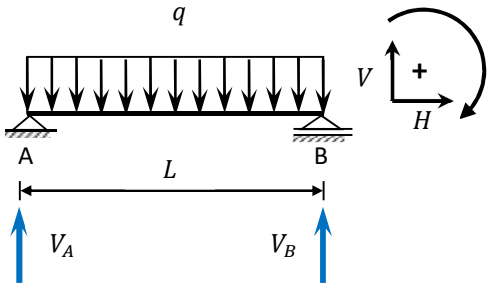


Réactions aux appuis :

$$\sum F_V = 0 \Leftrightarrow V_A + V_B - qL = 0 \Leftrightarrow V_A + V_B = qL$$

$$\sum M_{/A} = 0 \Leftrightarrow -V_B L + qL\frac{L}{2} = 0 \Leftrightarrow V_B = qL/2$$

$$\sum M_{/B} = 0 \Leftrightarrow V_A L - qL\frac{L}{2} = 0 \Leftrightarrow V_A = qL/2$$

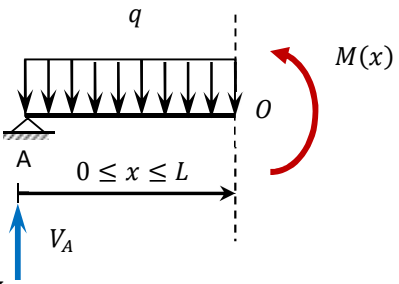


Sollicitations internes :

$$\sum M_{/O} = 0 \Leftrightarrow V_A x - qx\frac{x}{2} - M(x) = 0$$

$$\Leftrightarrow M(x) = V_A x - q\frac{x^2}{2} = q\frac{L}{2}x - q\frac{x^2}{2}$$

$$EI\frac{d^2y}{dx^2} = -M(x) = q\frac{x^2}{2} - q\frac{L}{2}x$$



$$EI\frac{dy}{dx} = \int M(x)\,dx = \int \left(q\frac{x^2}{2} - q\frac{L}{2}x\right)dx = q\frac{1}{6}x^3 - q\frac{L}{4}x^2 + C_1$$

$$EIy = \int \left(\frac{1}{6}x^3 - q\frac{L}{4}x^2 + C_1\right)dx = q\frac{1}{24}x^4 - q\frac{L}{12}x^3 + C_1 x + C_2$$

A u point (A) $x = 0 : y(0) = 0 \Leftrightarrow q\frac{1}{24}0^4 - q\frac{L}{12}0^3 + C_1 0 + C_2 = 0 \Leftrightarrow C_2 = 0$

A u point (B) $x = L : y(L) = 0 \Leftrightarrow q\frac{1}{24}L^4 - q\frac{L}{12}L^3 + C_1 L = 0 \Leftrightarrow C_1 = q\frac{1}{24}L^3$

Alors : $\varphi(x) = \frac{q}{24EI}[4x^3 - 6Lx^2 + L^3]$ et $y(x) = \frac{q}{24EI}[x^4 - 2Lx^3 + L^3 x]$

1MGCS M111 : Complément de programmation

TP5 : Rappels des calculs théoriques

Rappel sur les étapes de calcul des armatures longitudinales des sections rectangulaires soumises à la compression centrée selon le code BAEL 91/99 :

a) Données :

l_0 = 4.0m
N_G = 0.8 MN et N_Q = 0.4MN
Béton : f_{c28} = 25MPa et γ_b = 1.5
Acier : f_e = 400MPa et γ_s = 1.15
Section rectangulaire : a = 0.3m et b = 0.4m

b) Calculs préliminaires :

l_0 = 4.0m, l_f = 0.7×4.0 = 2.8m
N_G = 0.8 MN et N_Q = 0.4MN → N_U = 1.35× N_G + 1.50× N_Q = 1.35×0.8 + 1.50×0.4 = 1.68MN

c) Calculs en béton armé :

$B = a \times b$ = 0.3×0.4 = 0.12m²

$B_r = (a - 0.02) \times (b - 0.02)$ = (0.3-0.02)×(0.4-0.02) = 0.1064m²

$U = 2(a + b)$ = 2(0.3+0.4) = 1.40m

$\lambda = 2\sqrt{3} l_f / \min(a, b)$=$2\sqrt{3}$ 2.8/0.3 = 32.3316 ≤ 50 →

$$\alpha = \frac{0,85}{1+0,2\left(\frac{\lambda}{35}\right)^2} = \frac{0,85}{1+0,2\left(\frac{32.3316}{35}\right)^2} = 0.7261, \text{ si non } \alpha = 0,6\left(\frac{50}{\lambda}\right)^2$$

$$A_{TH} = \left(\frac{N_U}{\alpha} - \frac{B_r f_{c28}}{0,9\gamma_b}\right)\frac{\gamma_s}{f_e} = \left(\frac{1.68\times10^6}{0.7261} - \frac{0.1064\times25\times10^6}{0.9\ 1.5}\right)\frac{1.15}{400\times10^6} = 9.8733 \times 10^{-4}\ m^2$$

$$A_{Min1} = 4cm^2/U(1m) = 4 \times 10^{-4} \times 1.40 = 5.6 \times 10^{-4} m^2$$

$$A_{Min2} = 0.2\%\ B = (0.2/100) \times 0.12 = 2.4 \times 10^{-4} m^2$$

$$A_{Min} = \max(A_{Min1} \text{ et } A_{Min2}) = \max(5.6 \times 10^{-4} \text{ et } 2.4 \times 10^{-4}) = 5.6 \times 10^{-4} cm^2$$

$$A_{RET} = \max(A_{TH} \text{ et } A_{Min}) = \max(9.8733 \times 10^{-4} \text{ et } 5.6 \times 10^{-4}) = 9.8733 \times 10^{-4}\ m^2$$

d) Résultat final :

Section nécessaire : $9.8733 \times 10^{-4}\ m^2 = 9.8733\ cm^2$

1MGCS M111 : Complément de programmation

TP6 : Rappels des calculs théoriques

Les caractéristiques géométriques des profilés métalliques IPE sont regroupées sur les tableaux suivants :

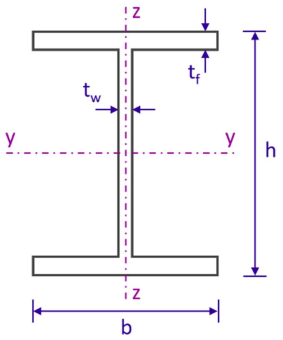


Tableau TP6-1 : Caractéristiques géométriques des profilés métalliques IPE80 à IPE200 [dimensions et poids].

Profilé	Dimensions (mm)				Poids (kg/m)
	h	b	t_f	t_w	
IPE80	80	46	5.2	3.8	6.0
IPE100	100	55	5.7	4.1	8.1
IPE120	120	64	6.3	4.4	10.4
IPE140	140	73	6.9	4.7	12.9
IPE160	160	82	7.4	5.0	15.8
IPE180	180	91	8.0	5.3	18.8
IPE200	200	100	8.5	5.6	22.4

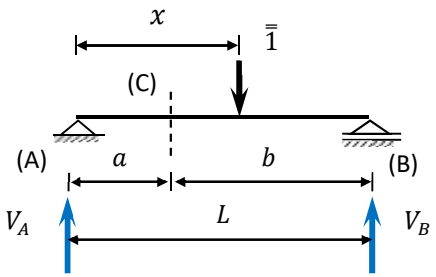
Tableau TP6-2 : Caractéristiques géométriques des profilés métalliques IPE80 à IPE200 [surface, moments d'inertie, modules de résistance et rayons de giration].

Profilé	Surface (cm^2)	Moments d'inertie (cm^4)		Modules de résistance élastique (cm^3)		Rayons de giration (cm)	
	A	$\mathbf{I_y}$	$\mathbf{I_z}$	$\mathbf{W_y}$	$\mathbf{W_z}$	$\mathbf{i_y}$	$\mathbf{i_z}$
IPE80	7.64	80.1	8.49	20.0	3.69	3.24	1.05
IPE100	10.3	171	15.9	34.2	5.79	4.07	1.24
IPE120	13.2	318	27.7	53.0	8.65	4.90	1.45
IPE140	16.4	541	44.9	77.3	12.3	5.74	1.65
IPE160	20.1	869	68.3	109	16.7	6.58	1.84
IPE180	23.9	1317	101	146	22.2	7.42	2.05
IPE200	28.5	1943	142	194	28.5	8.26	2.24

1MGCS M111 : Complément de programmation

TP7 : Rappels des calculs théoriques

Evaluer et tracer les lignes d'influence relatives de la poutre isostatique présentée sur la figure ci-après :



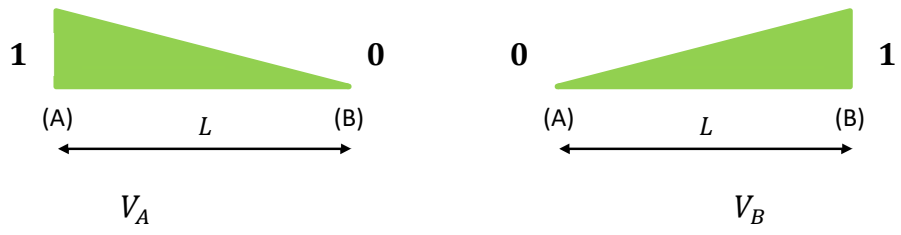
Lignes d'influence des réactions

$$\Sigma M/_C = 0 \leftrightarrow -V_A(L) + \bar{\bar{1}}(L-x) = 0 \leftrightarrow V_A = \bar{\bar{1}}\,\frac{L-x}{L} = 1 - \frac{x}{L}$$

- Pour $x = 0 \rightarrow V_A = 1 - \frac{0}{L} = 1$
- Pour $x = L \rightarrow V_A = 1 - \frac{L}{L} = 0$

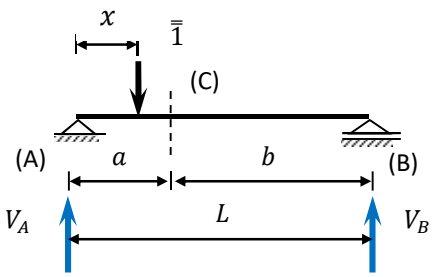
$$\Sigma M/_A = 0 \leftrightarrow -\bar{\bar{1}}(x) + V_B(L) = 0 \leftrightarrow V_B = \bar{\bar{1}}\,\frac{x}{L} = \frac{x}{L}$$

- Pour $x = 0 \rightarrow V_B = \frac{0}{L} = 0$
- Pour $x = L \rightarrow V_B = \frac{L}{L} = 1$



Lignes d'influence des sollicitations au point (C)

Coupe $0 \leq x \leq a$

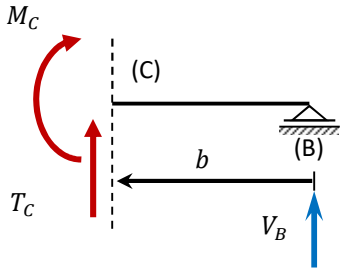


$\Sigma F/_y = 0 \leftrightarrow V_B + T_C = 0$

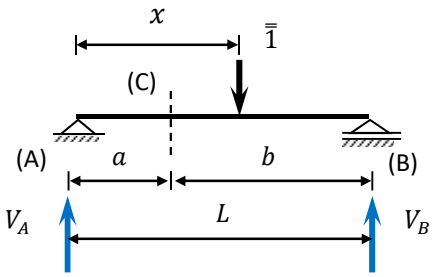
$\leftrightarrow T_C = -V_B = -\frac{x}{L}$

$\Sigma M/_C = 0 \leftrightarrow V_B(b) - M_C = 0$

$\leftrightarrow M_C = b\,V_B = b\frac{x}{L}$



Coupe $a \leq x \leq L$

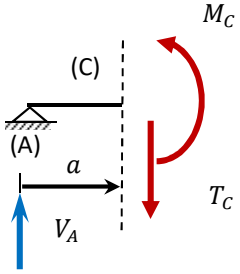


$\Sigma F/_y = 0 \leftrightarrow V_A - T_C = 0$

$\leftrightarrow T_C = V_A = 1 - \frac{x}{L}$

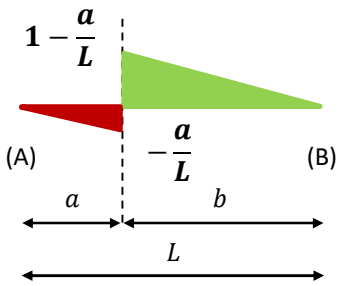
$\Sigma M/_C = 0 \leftrightarrow M_C - V_A(a)$

$\leftrightarrow M_C = a\,V_A = a\left(1 - \frac{x}{L}\right)$



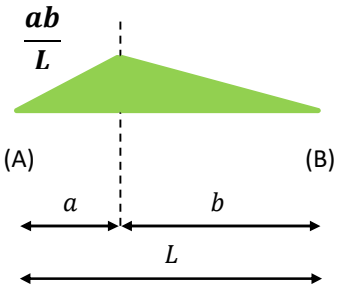
Lignes d'influence de l'effort tranchant au point (C)

Alors $T_C = \begin{cases} -\frac{x}{L} & 0 \le x \le a \\ 1 - \frac{x}{L} & a \le x \le L \end{cases}$

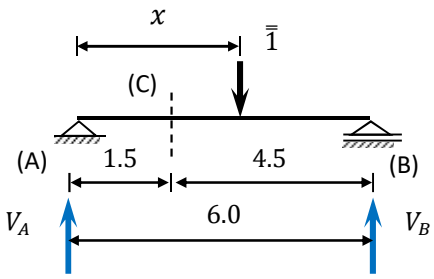


Lignes d'influence du moment fléchissant au point (C)

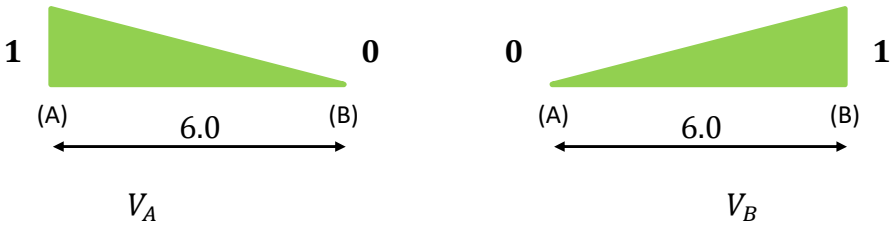
Alors $M_C = \begin{cases} b\frac{x}{L} & 0 \le x \le a \\ a\left(1 - \frac{x}{L}\right) & a \le x \le L \end{cases}$



Application numérique : L = 6m, a = 1 .5m et b = 4 .5m

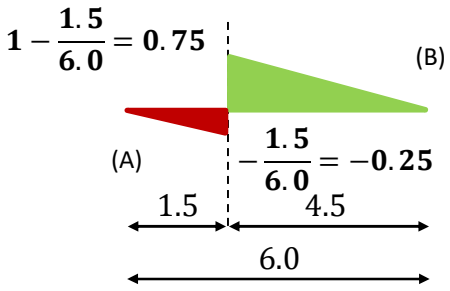


Lignes d'influence des réactions



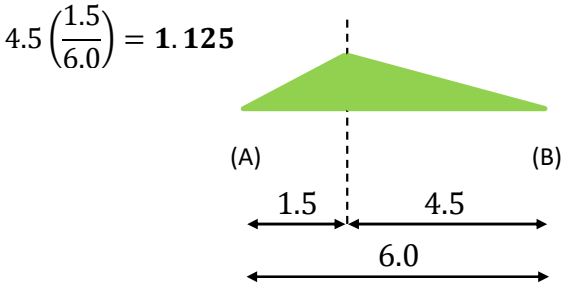
Lignes d'influence de l'effort tranchant au point (C)

Alors $T_C = \begin{cases} -\frac{x}{6.0} & 0 \le x \le 1.5 \\ 1 - \frac{x}{6.0} & 1.5 \le x \le 6.0 \end{cases}$



Lignes d'influence du moment fléchissant au point (C)

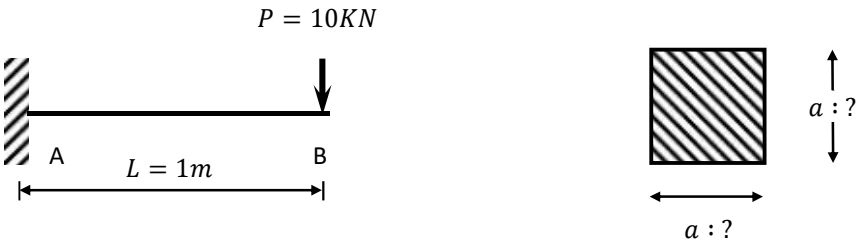
Alors $M_C = \begin{cases} 4.5\left(\frac{x}{6.0}\right) & 0 \le x \le 1.5 \\ 1.5\left(1 - \frac{x}{6.0}\right) & 1.5 \le x \le 6.0 \end{cases}$



1MGCS M111 : Complément de programmation

TP8 : Rappels des calculs théoriques

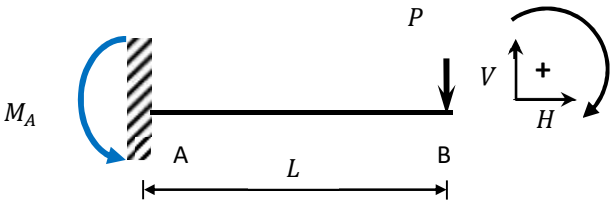
Dimensionnement d'une console : Trouver la section carrée convenable à la console présentée ci-après, pour laquelle les contraintes admissibles sont limitées aux $\bar{\tau} = 1.25$ MPa pour le cisaillement, $\overline{\sigma_c} = 10$ MPa pour la compression et $\overline{\sigma_t} = 5$ MPa pour la traction.



Réactions aux appuis :

$\sum F_V = 0 \Leftrightarrow V_A - P = 0 \Leftrightarrow V_A = P$

$\sum M_{/A} = 0 \Leftrightarrow PL - M_A = 0 \Leftrightarrow M_A = PL$



Sollicitations internes :

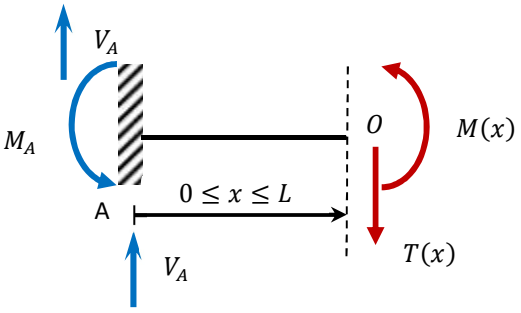
$\sum F_V = 0 \Leftrightarrow V_A - T(x) = 0 \Leftrightarrow T(x) = V_A$

$\Leftrightarrow T(x) = P$

$\sum M_{/O} = 0 \Leftrightarrow -M_A + V_A x - M(x) = 0$

$\Leftrightarrow M(x) = -M_A + V_A x$

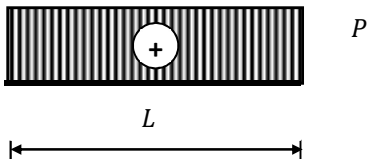
$\Leftrightarrow M(x) = -PL + Px = P(x - L)$



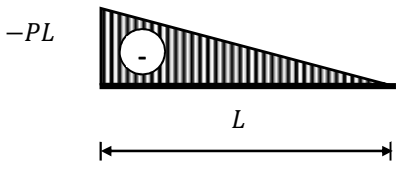
- $Pour\ x = 0 \rightarrow M(0) = -PL$
- $Pour\ x = L \rightarrow M(L) = 0$

Diagrammes des sollicitations :

Effort tranchant



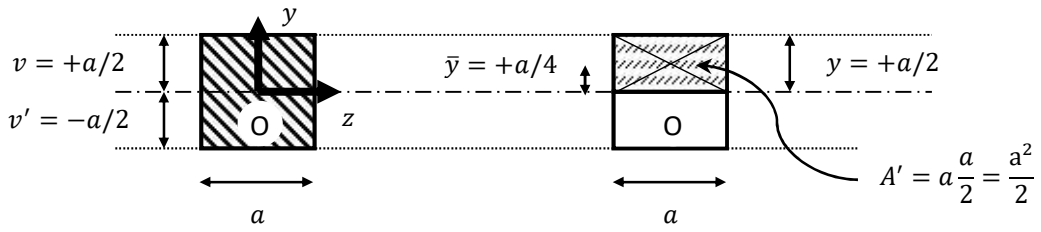
Moment fléchissant



La valeur limite de l'effort tranchant : $T(x) = P \; \forall \, x \; \rightarrow T_{Max} = \mathrm{P}$

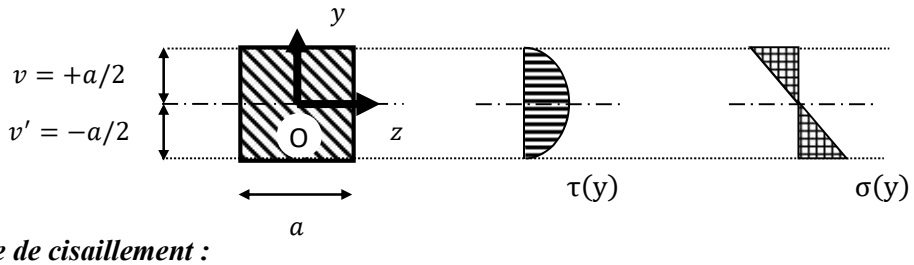
La valeur limite du moment fléchissant : à l'encastrement : $\mathrm{M}^{-}_{\mathrm{Max}} = -\mathrm{PL} \rightarrow M_{Max} = |-\mathrm{PL}\,| = \mathrm{PL}$

Caractéristiques géométriques de la section :



Position de l'axe neutre	*Moment statique*	*Moment d'inertie*
$v = +a/2$ et $v' = -a/2$	$S = \bar{y}A' = \frac{a}{4}\frac{\mathrm{a}^2}{2} = \frac{\mathrm{a}^3}{8}$	$\mathrm{I_z} = \frac{\mathrm{a} \times \mathrm{a}^3}{12} = \frac{\mathrm{a}^4}{12}$

Etat de contrainte :



Contrainte de cisaillement :

$$\tau_{Max} = \tau_{(0)} = \frac{S}{\mathrm{I_z}\, a} T_{Max} = \frac{\frac{\mathrm{a}^3}{8}}{\frac{\mathrm{a}^4}{12}\, a} T_{Max} = \frac{3}{2\mathrm{a}^2} T_{Max} = \frac{3}{2\mathrm{a}^2}\mathrm{P} \leq \bar{\tau}$$

$$\Leftrightarrow \mathrm{a}^2 \geq \frac{3\mathrm{P}}{2\bar{\tau}} \Leftrightarrow \mathrm{a}^2 \geq \frac{3 \times 10\;10^3}{2 \times 1.25\;10^6} = 1.2\;10^{-2}\mathrm{m}^2 \Leftrightarrow \mathbf{a \geq 0.109m}$$

Contraintes de flexion :

$$\sigma_{(v)} = -\frac{M_{Max}}{\mathrm{I_z}}(v) = -\frac{M_{Max}}{\frac{\mathrm{a}^4}{12}}\left(+\frac{\mathrm{a}}{2}\right) = -\frac{6}{\mathrm{a}^3} M_{Max} = -\frac{6}{\mathrm{a}^3}(\mathrm{PL}) = \left|-\frac{6}{\mathrm{a}^3}\mathrm{PL}\right| \leq \overline{\sigma_c}$$

$$\Leftrightarrow \mathrm{a}^3 \geq \frac{6\mathrm{PL}}{\overline{\sigma_c}} \Leftrightarrow \mathrm{a}^3 \geq \frac{6 \times 10.10^3 \times 1}{10.10^6} = 6.10^{-3}\mathrm{m}^3 \Leftrightarrow \mathbf{a \geq 0.182m}$$

$$\sigma_{(v')} = -\frac{M_{Max}}{\mathrm{I_z}}(-v') = -\frac{M_{Max}}{\frac{\mathrm{a}^4}{12}}\left(-\frac{\mathrm{a}}{2}\right) = +\frac{6}{\mathrm{a}^3} M_{Max} = +\frac{6}{\mathrm{a}^3}(\mathrm{PL}) = \left|+\frac{6}{\mathrm{a}^3}\mathrm{PL}\right| \leq \overline{\sigma_t}$$

$$\Leftrightarrow \mathrm{a}^3 \geq \frac{6\mathrm{PL}}{\overline{\sigma_t}} \Leftrightarrow \mathrm{a}^3 \geq \frac{6 \times 10.10^3 \times 1}{5.10^6} = 12.10^{-3}\mathrm{m}^3 \Leftrightarrow \mathbf{a \geq 0.229m}$$

Pour satisfaire les trois inégalités : $\mathrm{a} \geq (0.109, 0.182, 0.229) \Leftrightarrow \mathrm{a} \geq 0.229\mathrm{m}$. Alors ; la section convenable est $0.23\mathrm{m}$; soit $\mathbf{a = 23cm}$