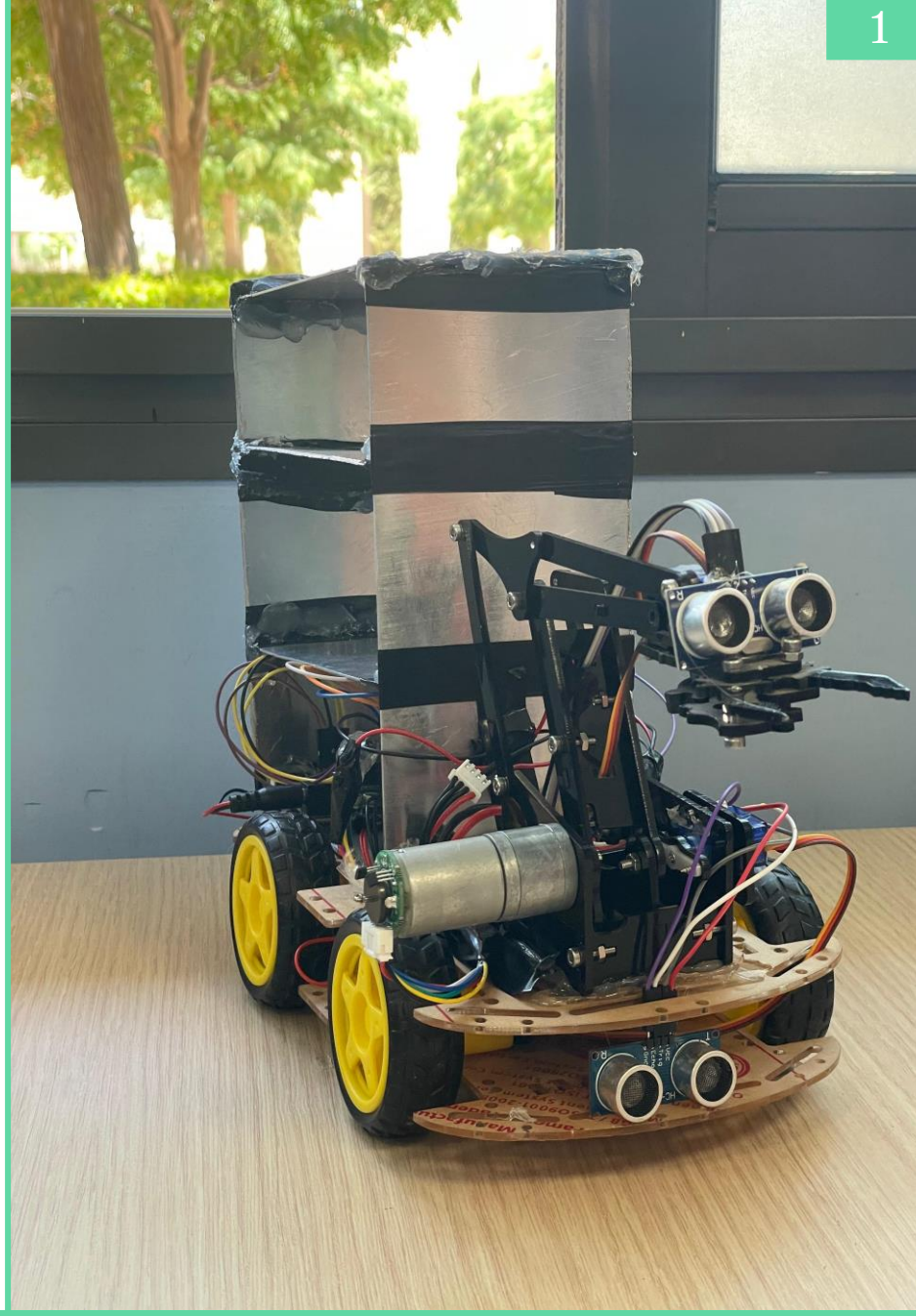
HôpitAssist

un robot de livraison au sein des hôpitaux



Nom et Prénom : MRIMI MOUAD | Code Cnc: BE062T | Session: 2025

PLAN

- Introduction
- Objectifs :
  - Navigation précise vers la chambre du patient.
  - Choix du capteur de détection d'obstacles.
  - Valider le diamètre et le matériau de l'arbre moteur.
  - Analyse les contraintes appliquées sur les étagères et le choix du matériau adapté
  - Construire un prototype pour valider la mise en œuvre du système.
- Conclusion

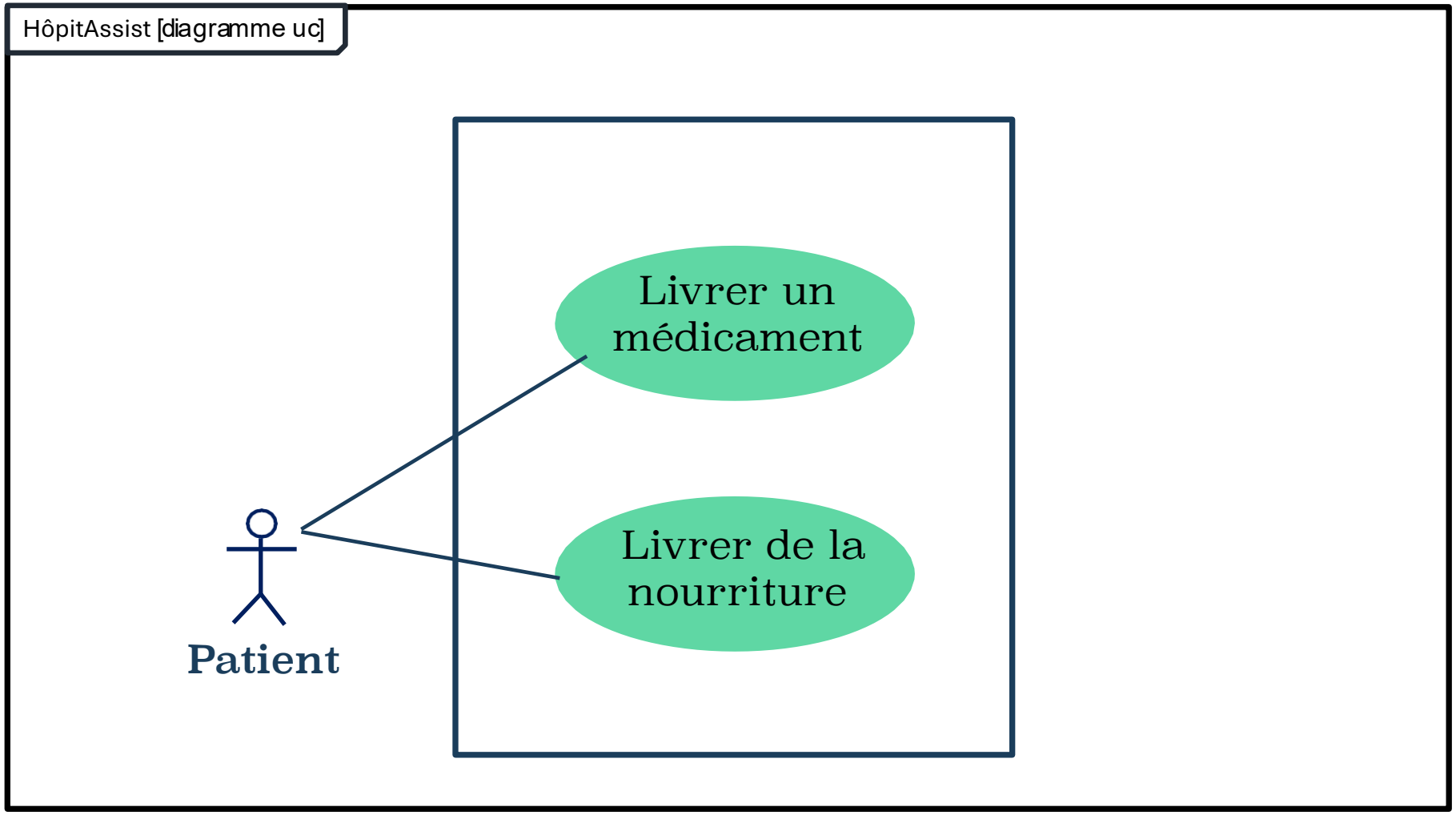
HôpitAssist : quelle valeur ajoutée ?

https://fr.reemanrobot.com/

Problématique:

Comment automatiser la livraison de médicaments et de repas jusqu'aux chambres des patients, tout en assurant une navigation efficace dans les couloirs?

Diagramme des cas d'utilisation:



1er objectif:

Navigation précise vers la chambre du patient:

- **Principe de l'algorithme BFS.**
- **Cartographie et optimisation.**
- **Comparaison des algorithmes BFS et Dijkstra.**
- **Odométrie.**
- **Stratégie de commande: logique floue.**
- **Asservissement en vitesse.**
- **Simulation sur MATLAB.**

Principe de l'algorithme BFS

1. Construction de la grille:

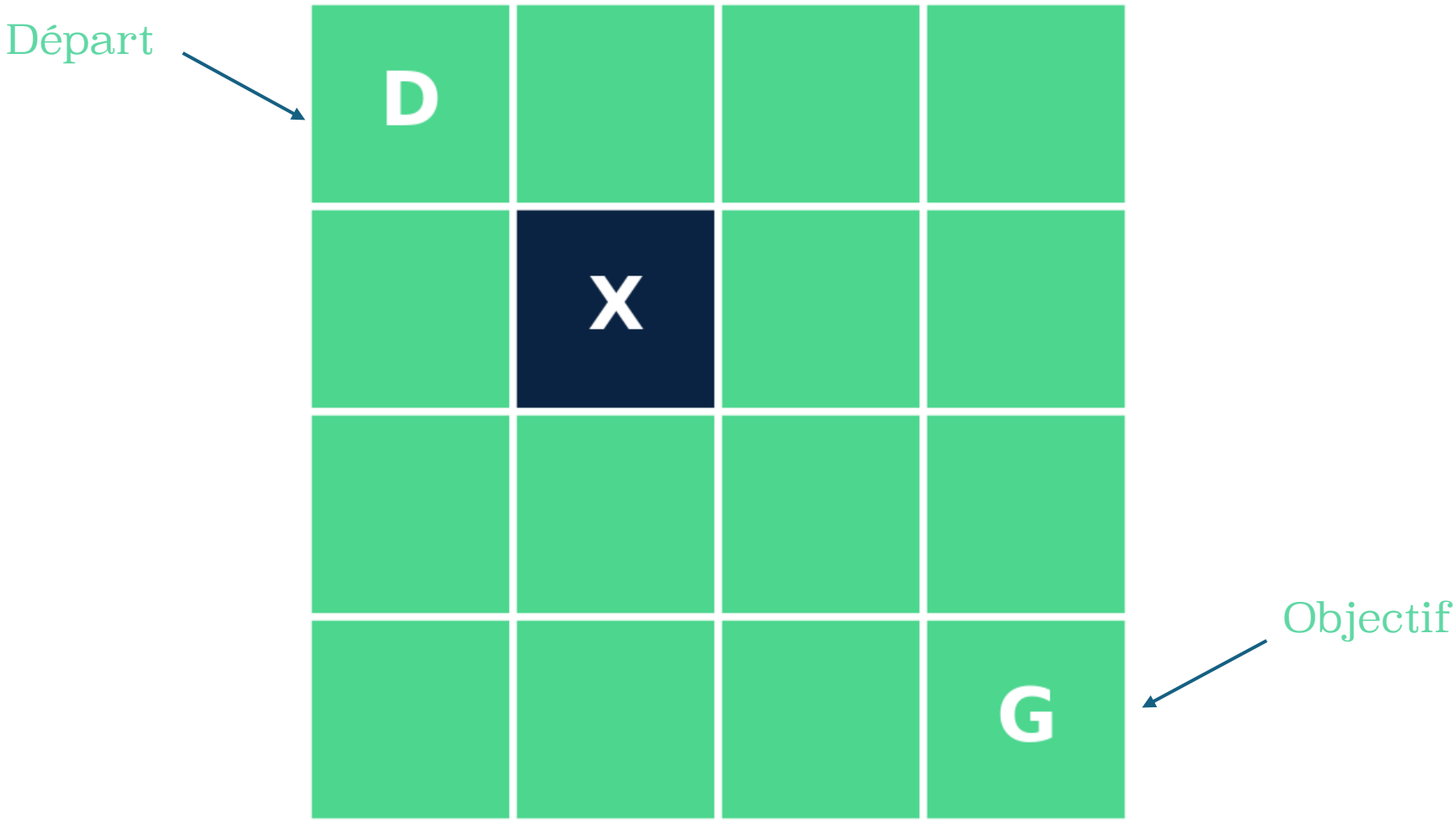


Figure 1.1

Principe de l'algorithme BFS

2. Grille → Graphe de navigation:

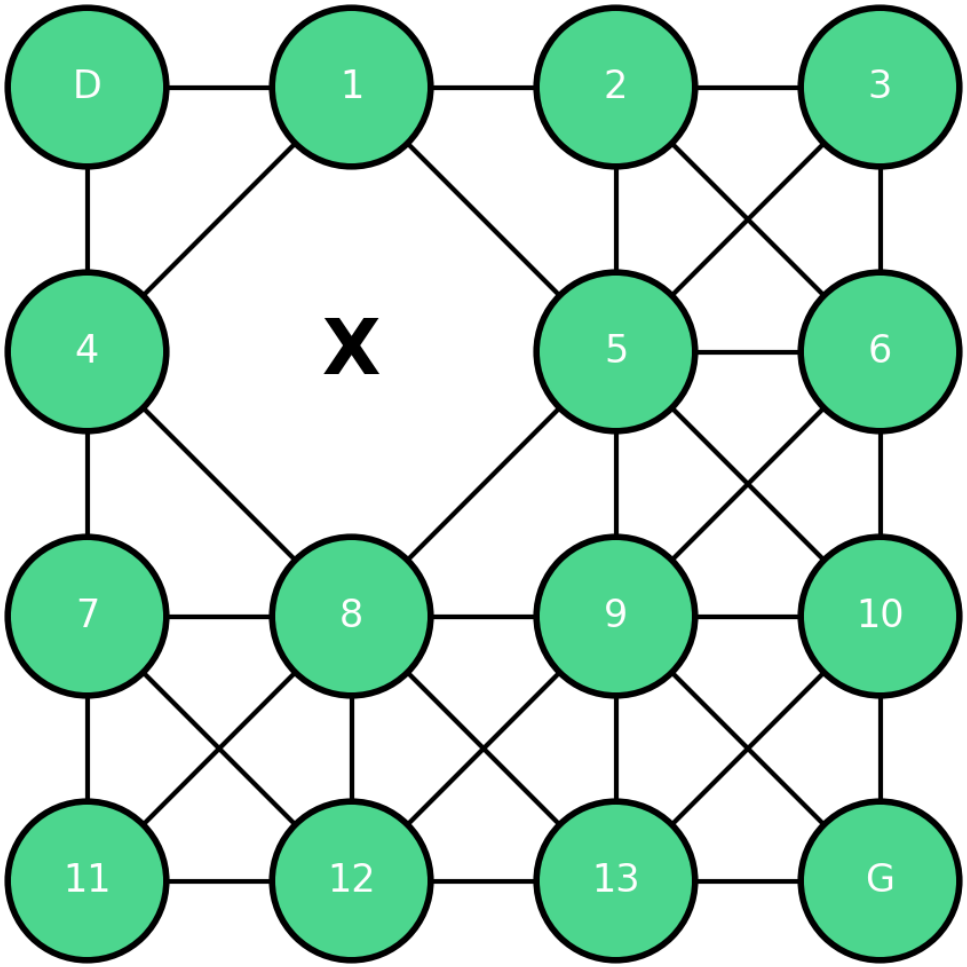


Figure 1.2

Principe de l'algorithme BFS

3. Couches de distance:

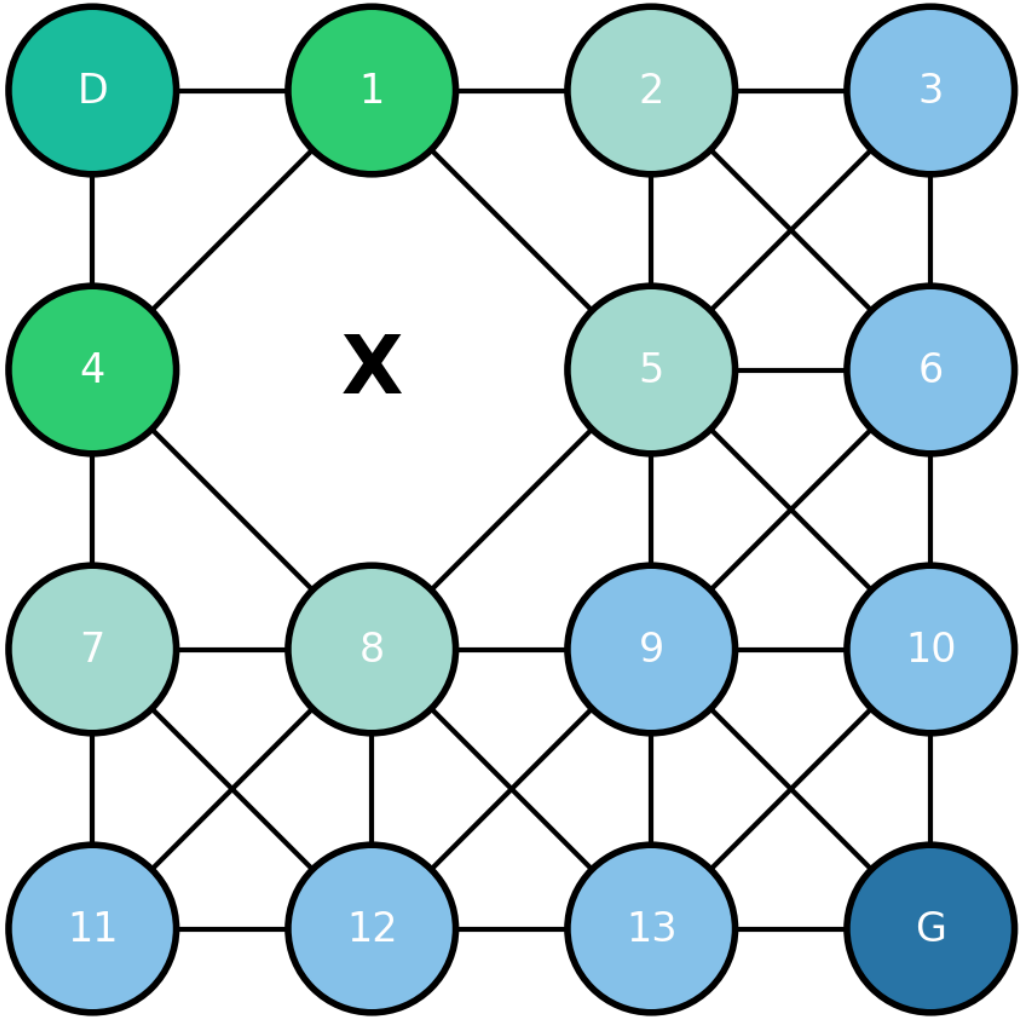
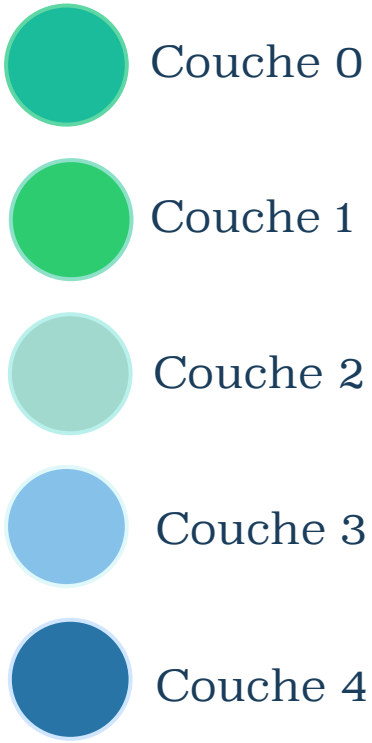


Figure 1.3



Principe de l'algorithme BFS

4. Exploration par couches:

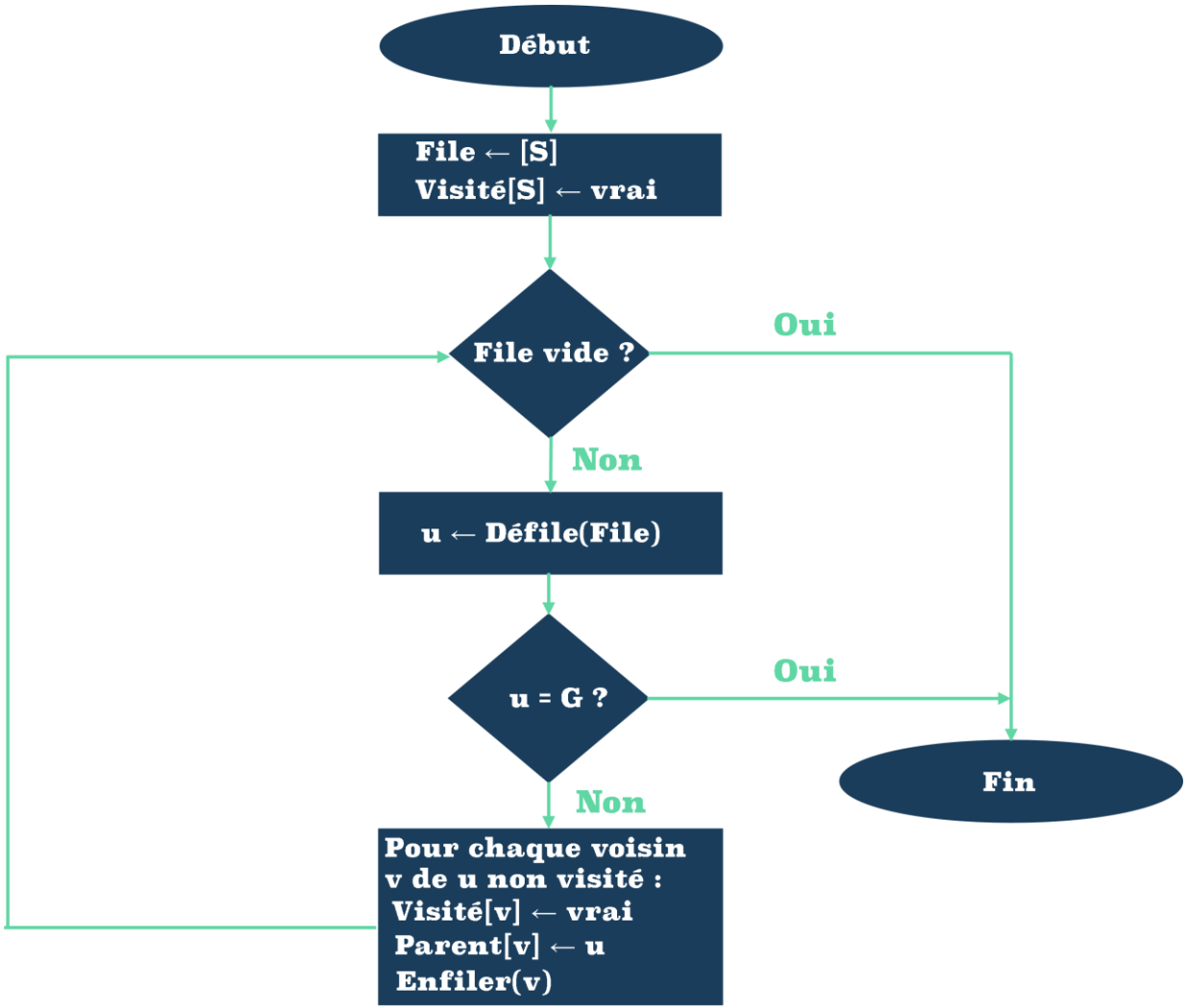


Figure 1.4

Principe de l'algorithme BFS

5. Évolution de la File:

Étape	u dépilé	File avant	File après	Nouveaux parents
0	—	—	[D]	—
1	D	[D]	[4, 1]	4←D, 1←D
2	4	[4,1]	[1, 7, 8]	7←4, 8←4
3	1	[1, 7, 8]	[7, 8, 5, 2]	5←1, 2←1
4	7	[7, 8, 5, 2]	[8, 5, 2, 11, 12]	11←7, 12←7
5	8	[8, 5, 2, 11, 12]	[5, 2, 11, 12, 13, 9]	13←8, 9←8
6	5	[5, 2, 11, 12, 13, 9]	[2, 11, 12, 13, 9, 10, 6, 3]	10←5, 6←5, 3←5
7	2	[2, 11, 12, 13, 9, 10, 6, 3]	[11, 12, 13, 9, 10, 6, 3]	—
8	11	[11, 12, 13, 9, 10, 6, 3]	[12, 13, 9, 10, 6, 3]	—
9	12	[12, 13, 9, 10, 6, 3]	[13, 9, 10, 6, 3]	—
10	13	[13, 9, 10, 6, 3]	[9, 10, 6, 3, G]	G←13
11	9	[9, 10, 6, 3, G]	[10, 6, 3, G]	—
12	10	[10, 6, 3, G]	[6, 3, G]	—
13	6	[6, 3, G]	[3, G]	—
14	3	[3, G]	[G]	—
15	G	[G]	[]	—

Figure 1.5

Principe de l'algorithme BFS

6. Processus de back-tracking:

nœud v	parent[v]
D	—
1	D
4	D
2	1
5	1
7	4
8	4
3	2
6	2
10	5
9	5
13	8
12	8
11	8
G	10

Figure 1.6

Principe de l'algorithme BFS

7. Le chemin le plus court:

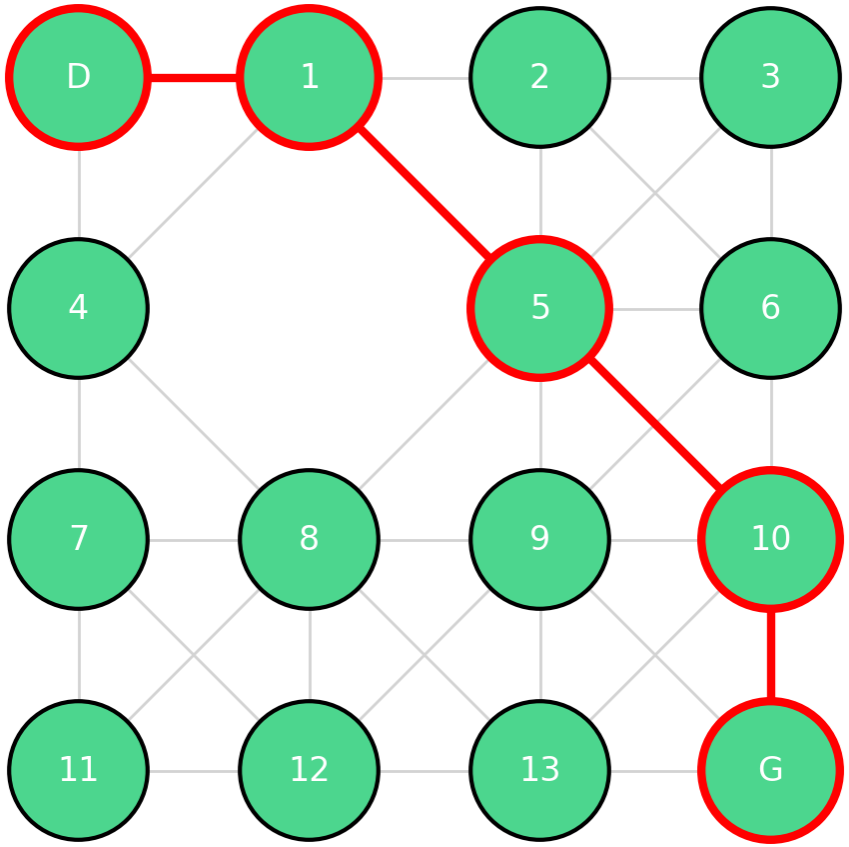


Figure 1.7

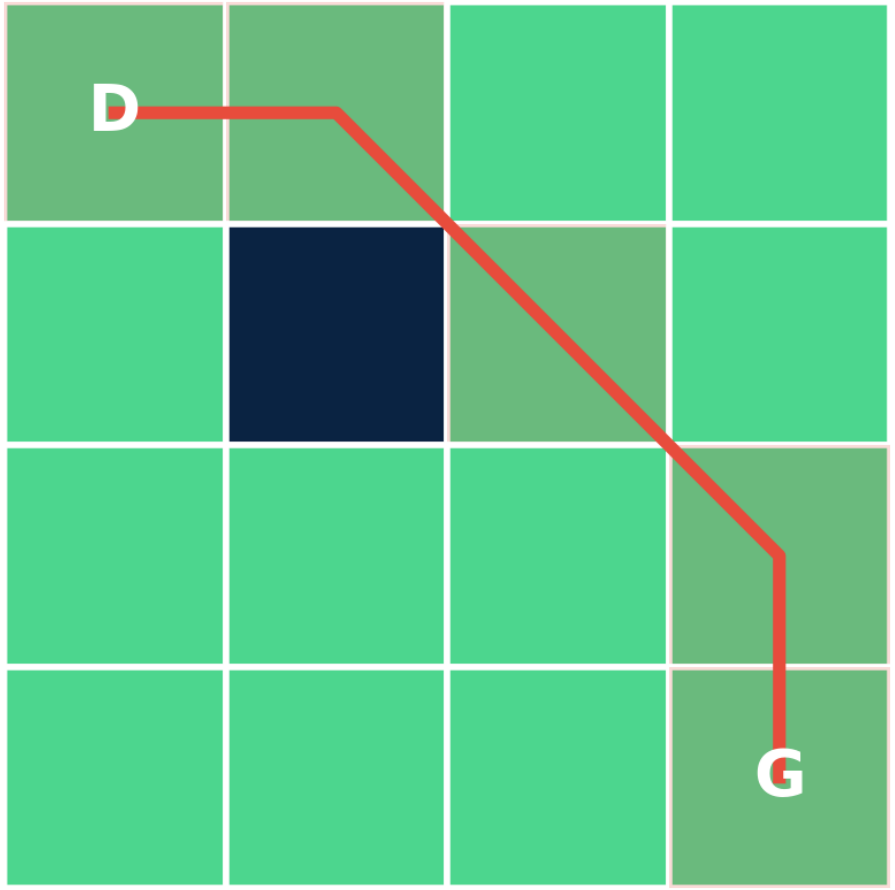


Figure 1.8

Cartographie et optimisation

1. Hypothèses de modélisation:

- Robot = point pour BFS (15×25 cm géré plus tard)
- 8-connectivité (N, NE, E, SE, S, SO, O, NO)
- Carte statique : seuls les obstacles marqués sont pris en compte
- Cellule homogène : chaque case 10×10 cm a un coût identique → pas de pondération.

Cartographie et optimisation

2. Carte brute (24 × 24):

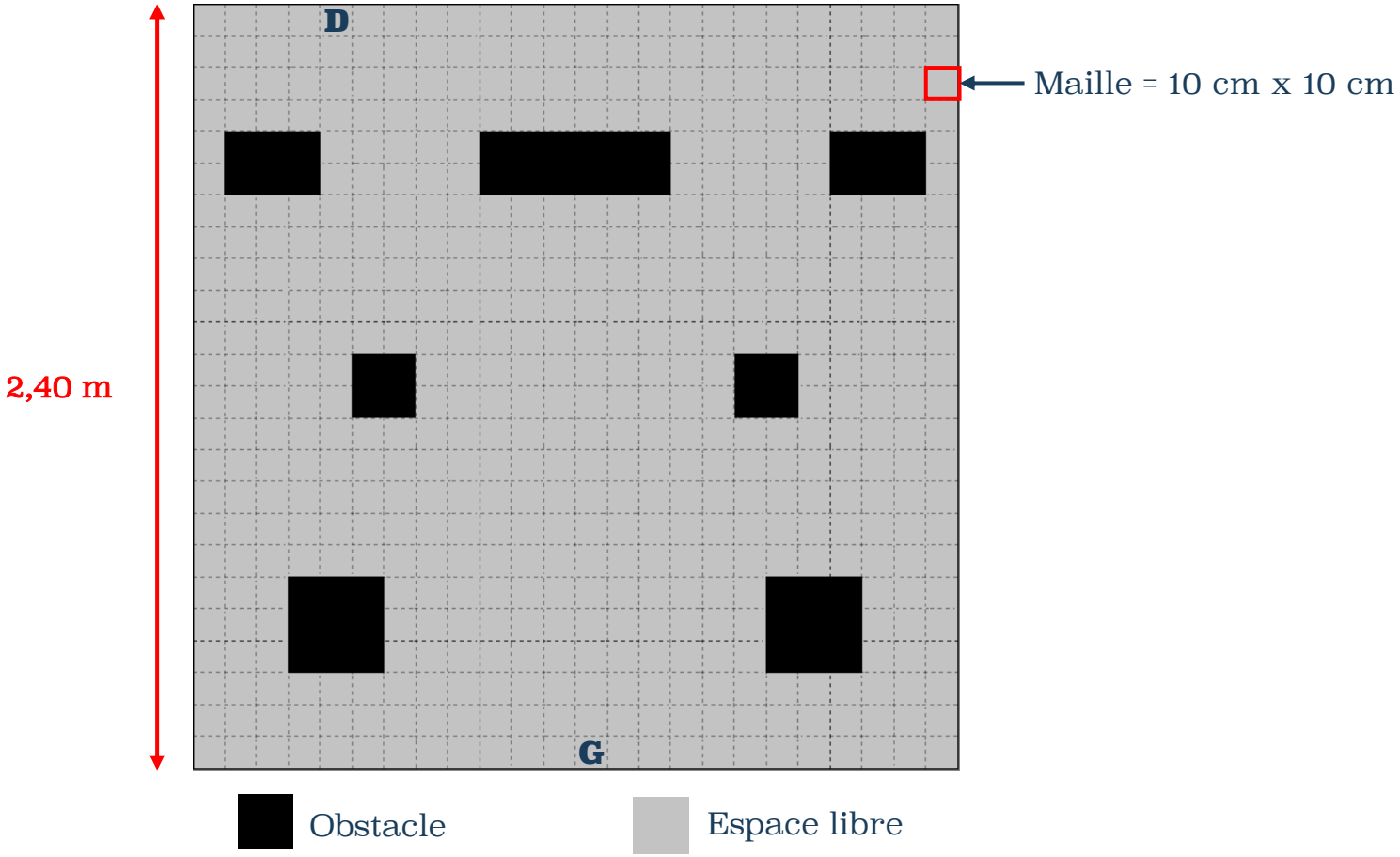
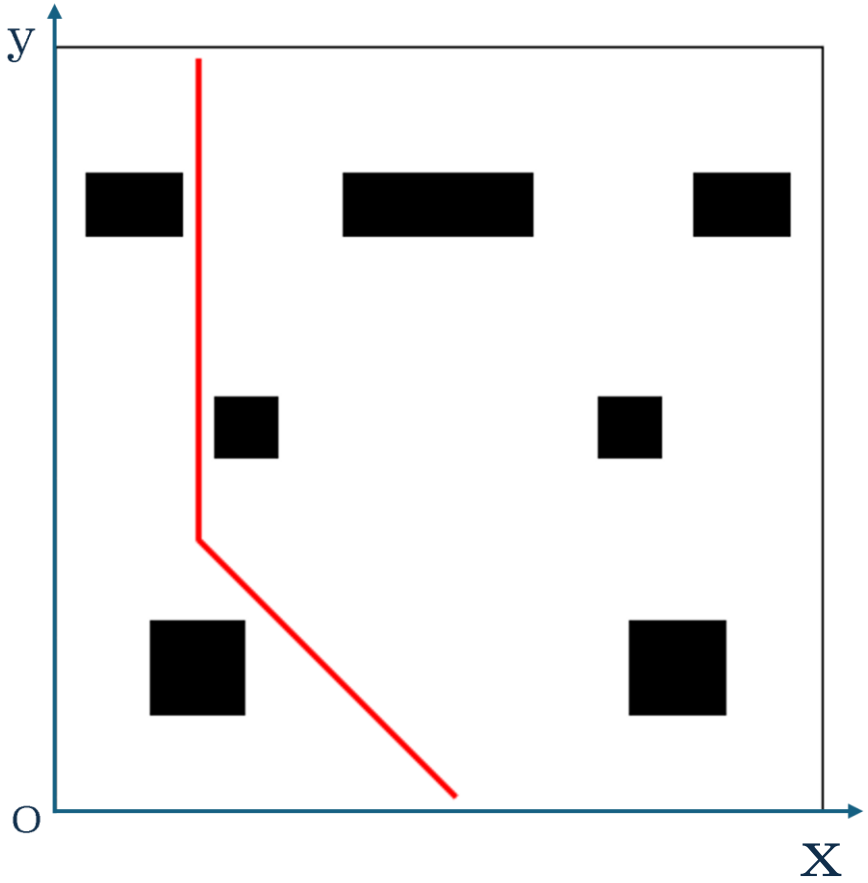


Figure 1.9

Cartographie et optimisation

3. BFS sur carte brute:




```
Waypoints (x, y) en cm :
[(40, 230), (40, 220), (40, 210), (40, 200), (40, 190), (40, 180), (40, 170), (40, 160), (40, 150), (40, 140), (40, 130), (40, 120), (
40, 110), (40, 100), (40, 90), (40, 80), (50, 70), (60, 60), (70, 50), (80, 40), (90, 30), (100, 20), (110, 10), (120, 0)]
```

Figure 1.10

Cartographie et optimisation

4. Limite du chemin brut:

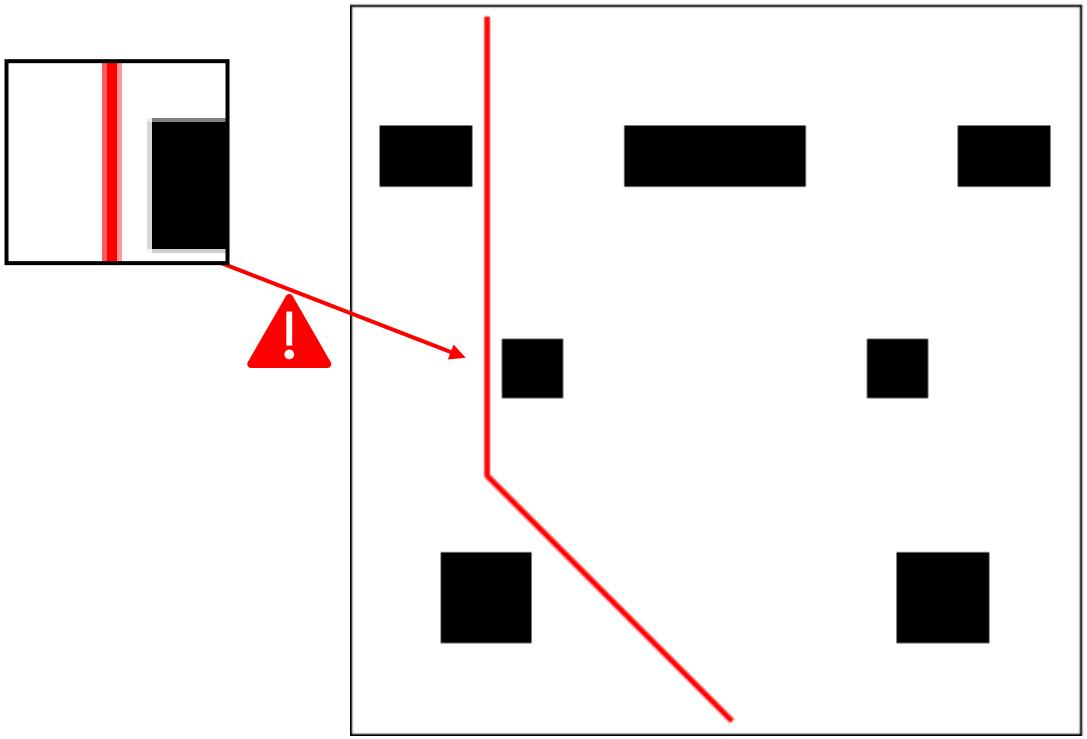


Figure 1.11

- On remarque que le chemin brut frôle dangereusement les obstacles (BFS traite le robot comme un point), or celui-ci a un gabarit réel non négligeable, ce qui crée un risque de collision.

Cartographie et optimisation

5. Avant / Après gonflage:

- demi-largeur robot (7,5 cm) → Gonflage = 1 case (10 cm)
- tolérance (~2,5 cm) pour marge de sécurité

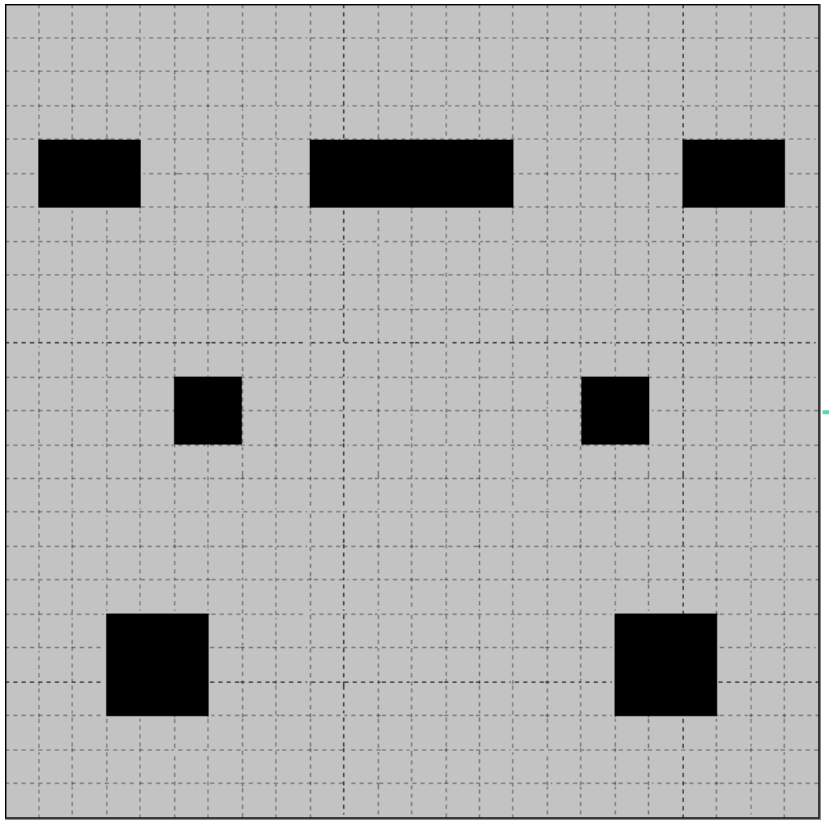


Figure 1.12: Avant gonflage

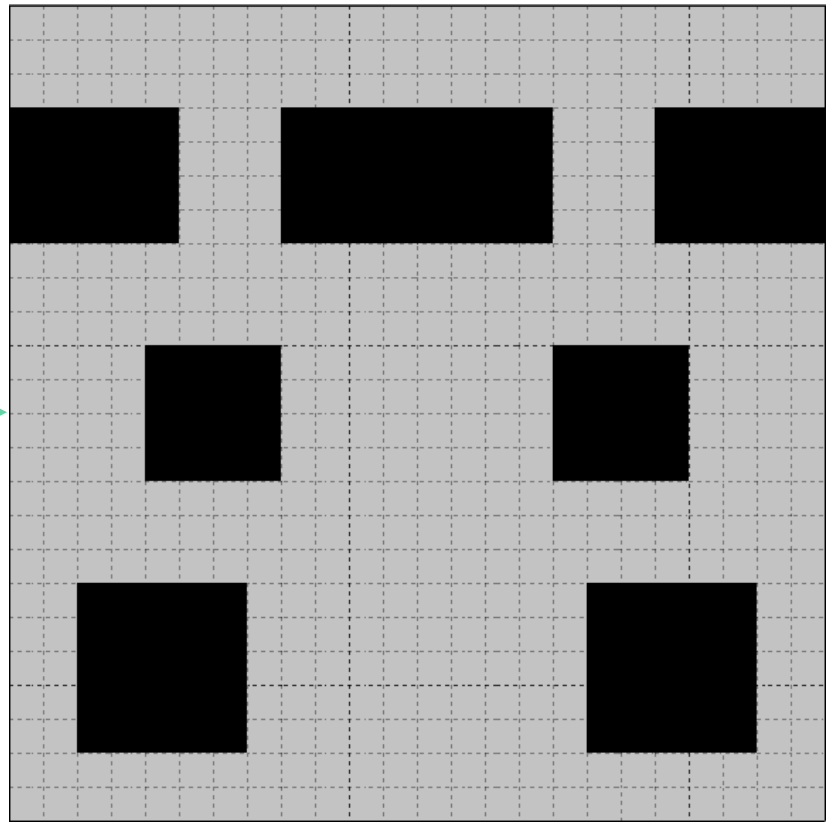


Figure 1.13: Après gonflage

Cartographie et optimisation

6. Impacts du gonflage sur BFS:

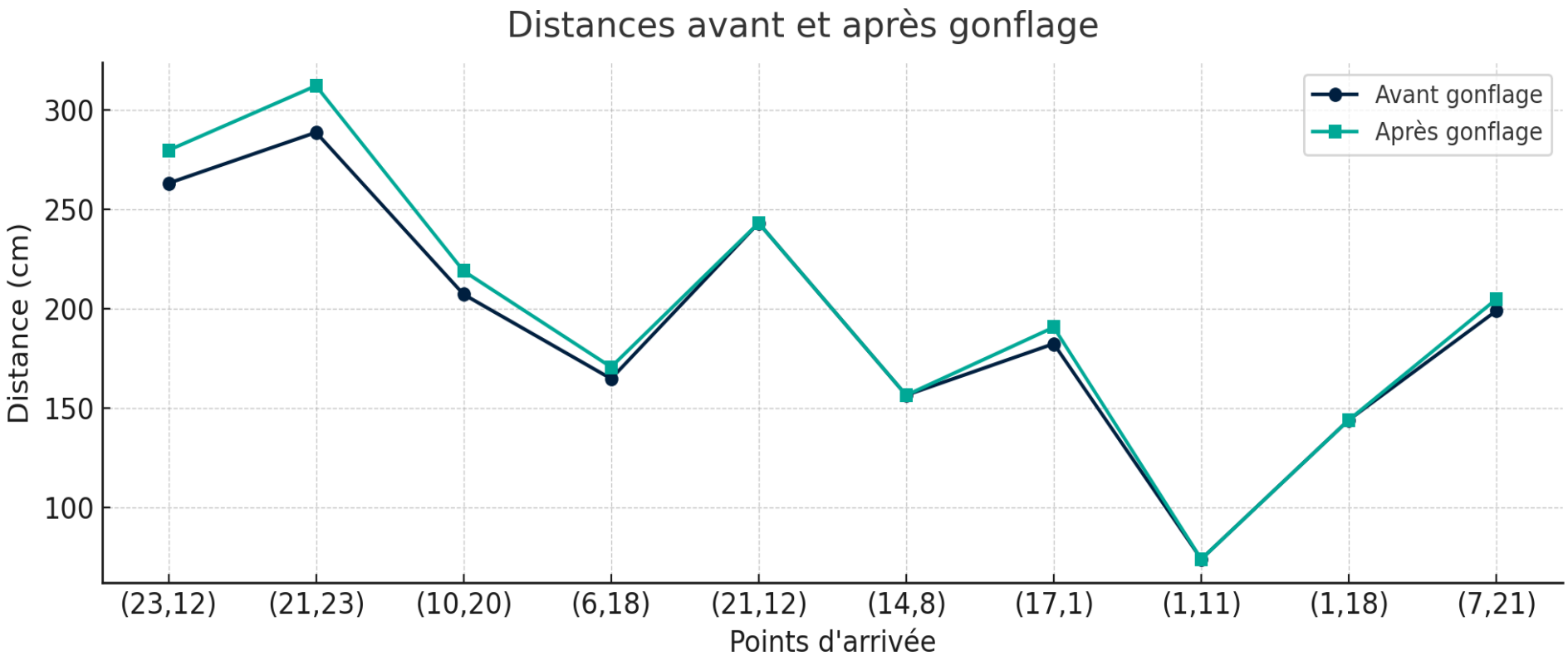


Figure 1.14

Cartographie et optimisation

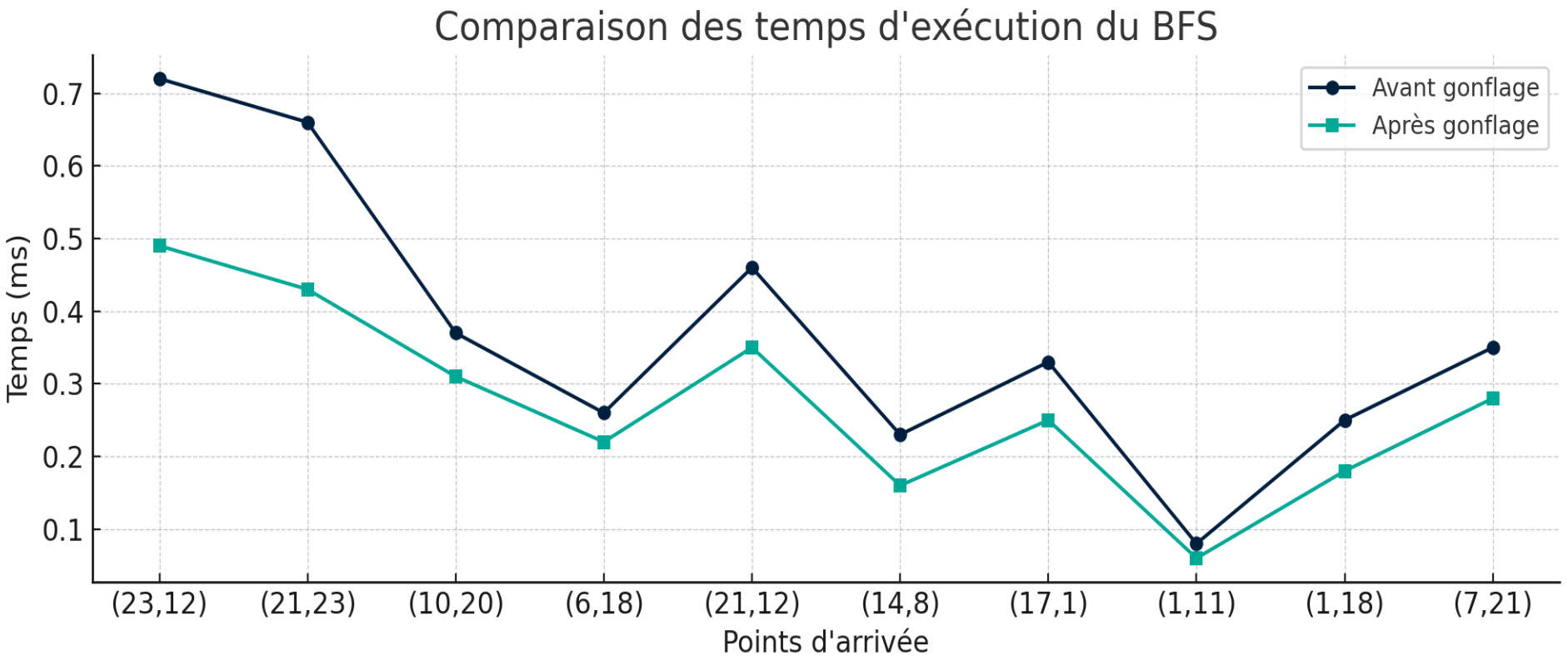


Figure 1.15

7. Fusion de segments rectilignes:

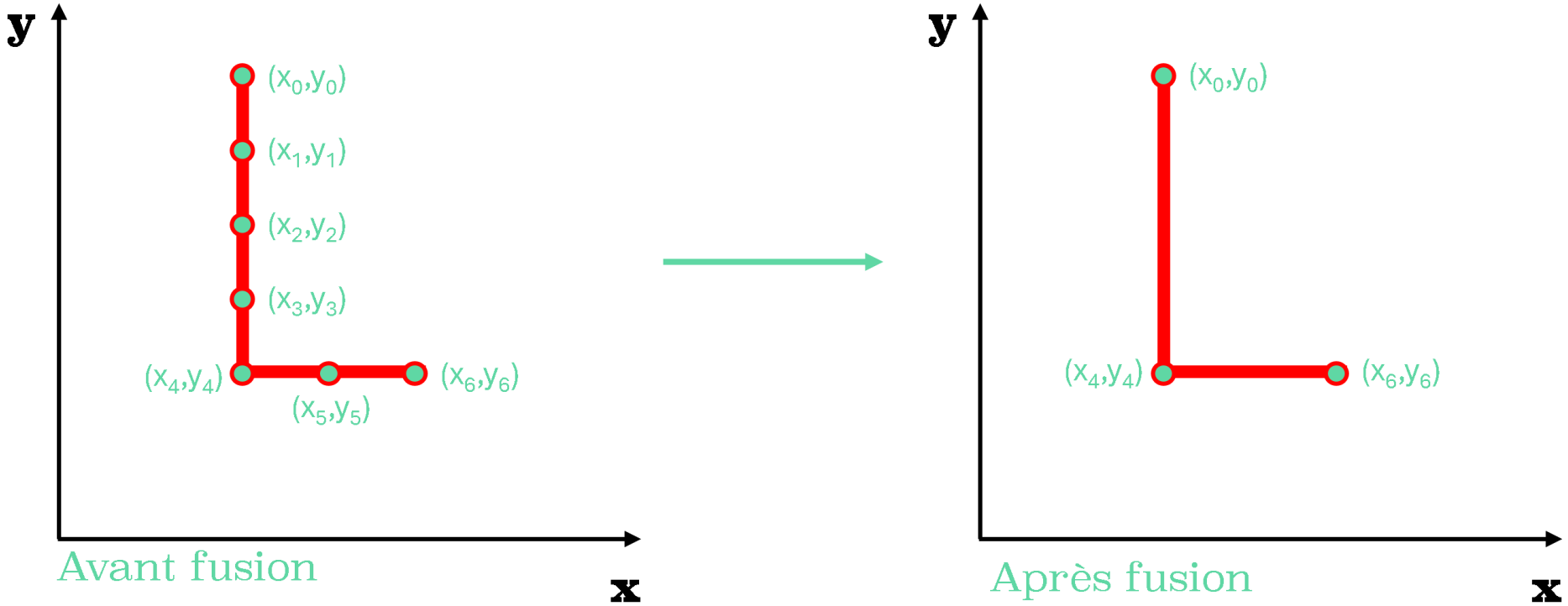


Figure 1.16

Comparaison des algorithmes BFS et Dijkstra

Mes résultats (BFS):

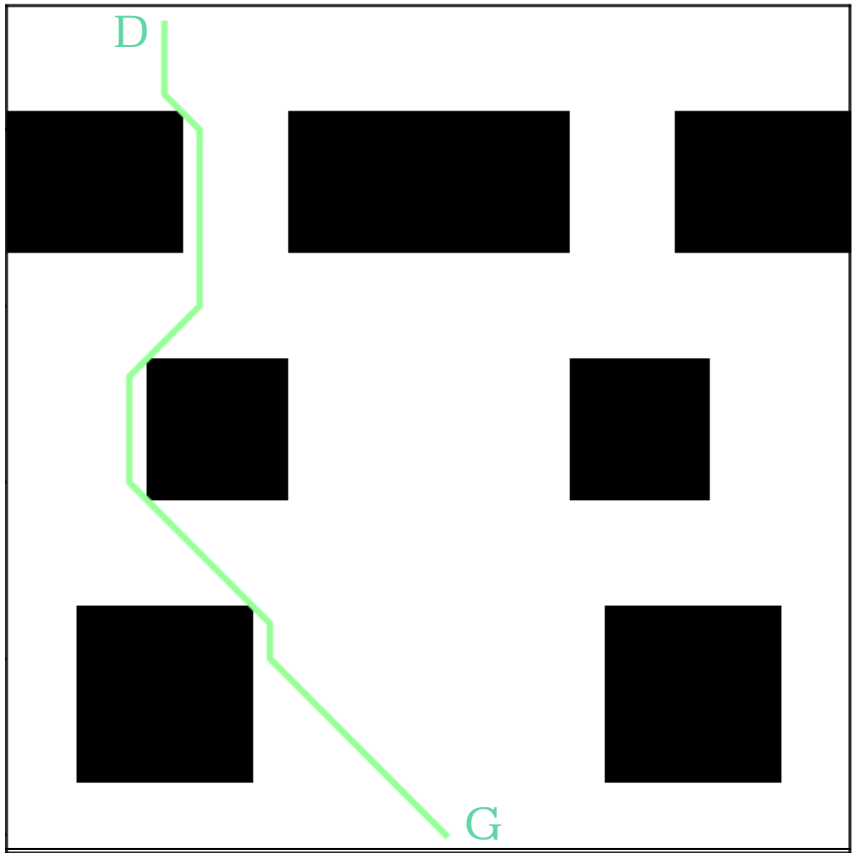


Figure 1.17

Nombre de pas : 22

Distance parcourue : 279.7 cm

Les résultats de mon binôme (Dijkstra):

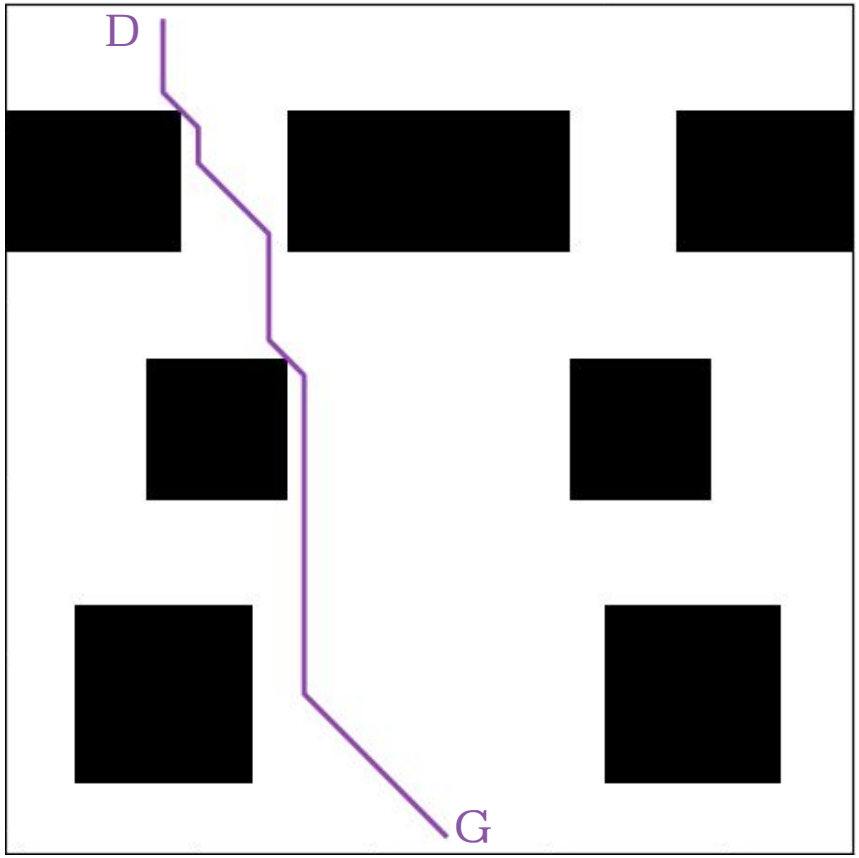


Figure 1.18

Nombre de pas : 23

Distance parcourue : 263.14 cm

Odométrie

1. Principe & Hypothèses:

- Objectif: Estimer la pose (x, y, θ) à partir des mesures des codeurs.

- Hypothèses:
  - Roulement sans glissement.
  - Châssis rigide, mouvement plan (2 D.L).
  - Chaque paire de roues d'un même flanc reçoit une commande identique (modèle différentielle).
  - Rayon de roue constant r,entraxe L.

Odométrie

2. Modèle géométrique du robot:

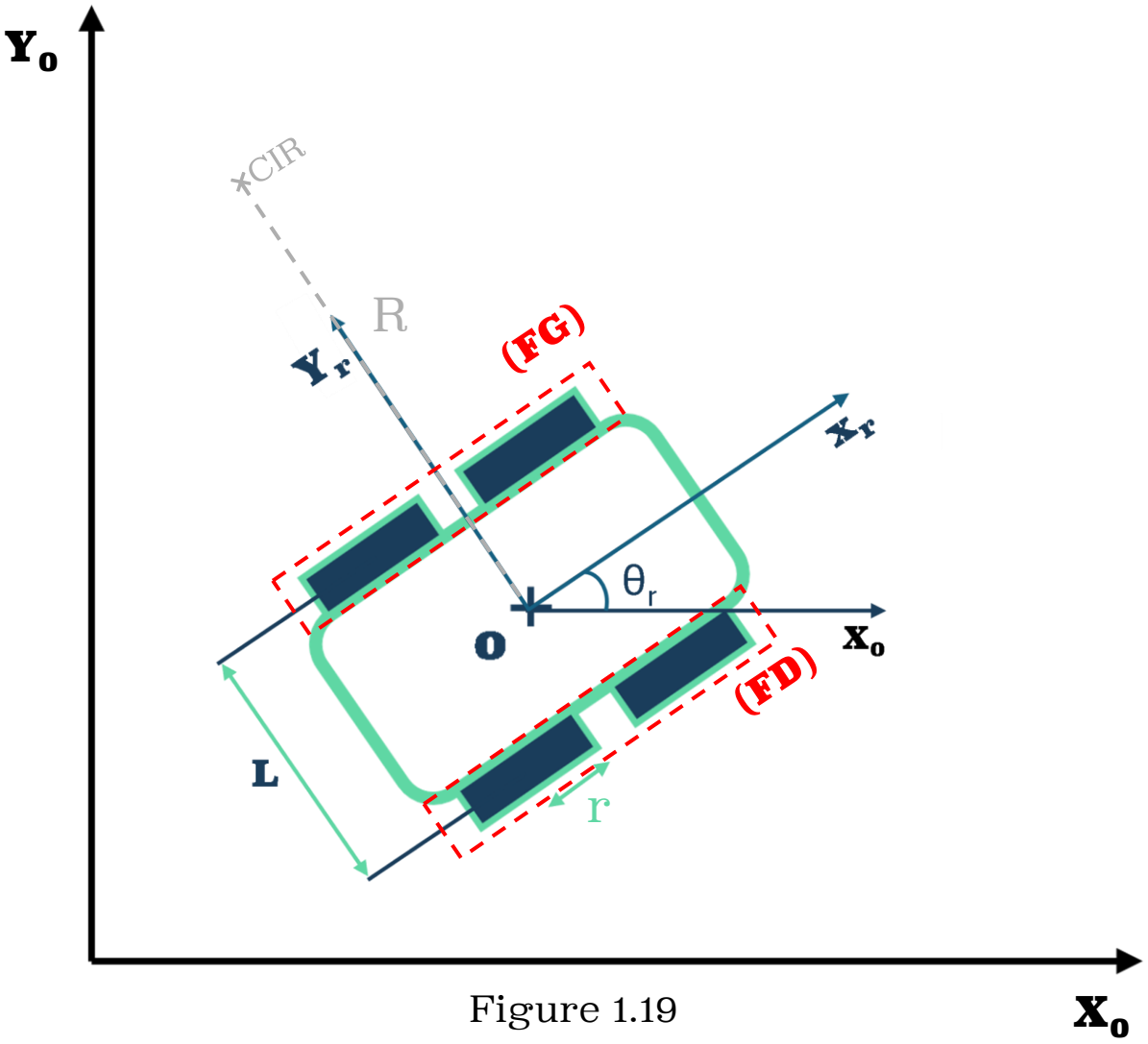


Figure 1.19

- **(FD):** flanc droit
- **(FG):** flanc gauche
- **r:** rayon de roue
- **L:** entraxe flanc gauche / flanc droit
- θ_r: orientation actuelle du robot
- **CIR:** centre instantané de rotation

Odométrie

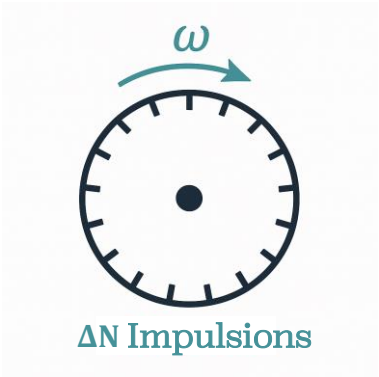
3. De l'encodeur à la distance roulée:

❖ Conversion impulsions → angle:

$$\Delta\varphi = \Delta N_{imp} \times \frac{2\pi}{N_{imp/tour}}$$

❖ Angle → distance parcourue:

$$\begin{cases} \Delta S_G = r\Delta\varphi_G \\ \Delta S_D = r\Delta\varphi_D \end{cases}$$



Odométrie

4. Cinématique différentielle:

❖ Vitesse linéaire & angulaire:

$$\begin{cases} v = \dfrac{v_D + v_G}{2} \\ \omega = \dfrac{v_D - v_G}{\mathrm{L}} \end{cases}$$

❖ Cas particuliers:

- Trajectoire rectiligne: $v_G = v_D$
- Rotation sur place: $v_G = -v_D$

Odométrie

5. Mise à jour de la pose (x, y, θ):

❖ Incréments:

$$\begin{cases} \Delta S = \dfrac{\Delta S_D + \Delta S_G}{2} \\ \Delta\theta = \dfrac{\Delta S_D - \Delta S_G}{L} \end{cases}$$

❖ Suites:

$$\begin{cases} x_{k+1} = x_k + \Delta S \cdot \cos\left(\theta_k + \dfrac{\Delta\theta}{2}\right) \\ y_{k+1} = y_k + \Delta S \cdot sin\left(\theta_k + \dfrac{\Delta\theta}{2}\right) \\ \theta_{k+1} = \theta_k + \Delta\theta \end{cases}$$

Stratégie de commande: Logique floue

1. Qu'est-ce que la logique floue:

- La logique floue est une méthode de raisonnement qui, au lieu de n'évaluer chaque proposition qu'en vrai (1) ou faux (0), lui attribue un degré de vérité continu μ compris entre 0 et 1.

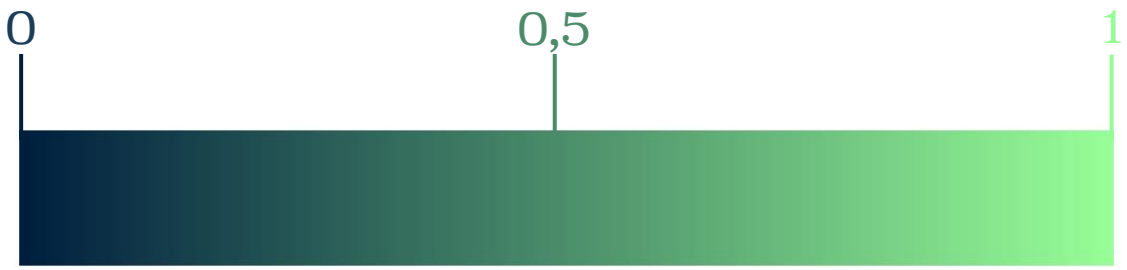


Figure 1.20: Degré d'appartenance μ

Stratégie de commande: Logique floue

2. Architecture d'un contrôleur flou:

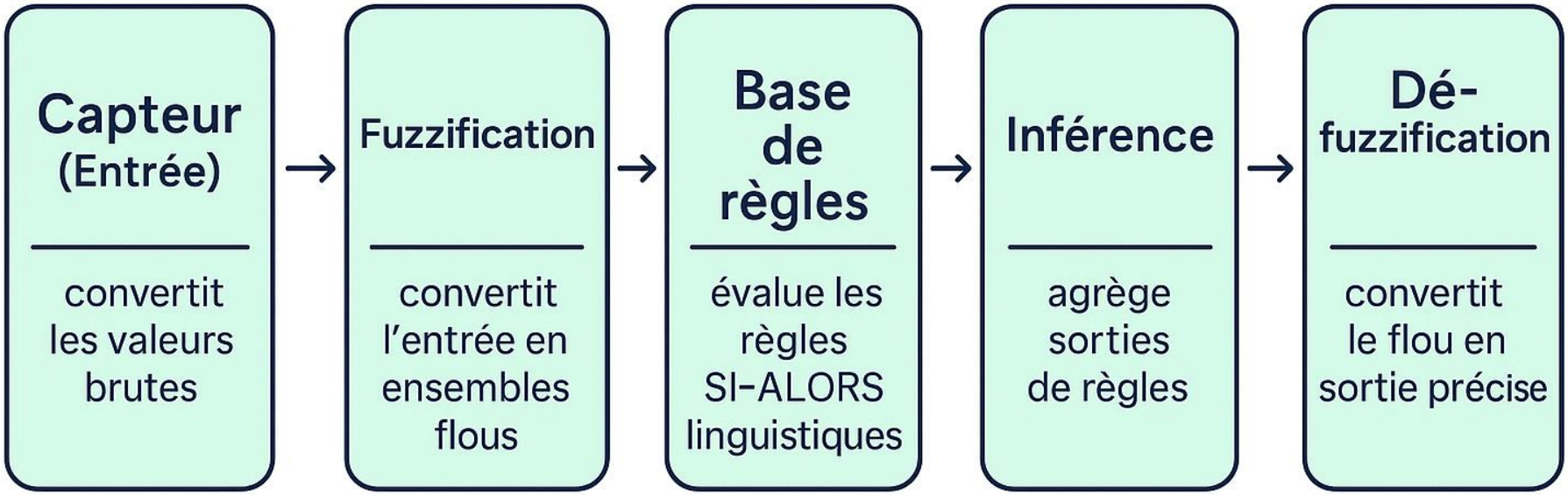


Figure 1.21

Stratégie de commande: Logique floue

3. Calcul d'erreurs:

Orientation absolue du vecteur allant du robot vers la cible:

$$\phi = a\tan 2(y_c - y_a, x_c - x_a)$$

Erreur d'orientation - amplitude et sens du pivot nécessaire:

$$e_\theta = wrap(\phi - \theta_a)$$

Erreur de distance - distance euclidienne séparant le robot de son objectif:

$$e_d = \sqrt{(x_c - x_a)^2 + (y_c - y_a)^2}$$

Stratégie de commande: Logique floue

4. Fonctions d'appartenance:

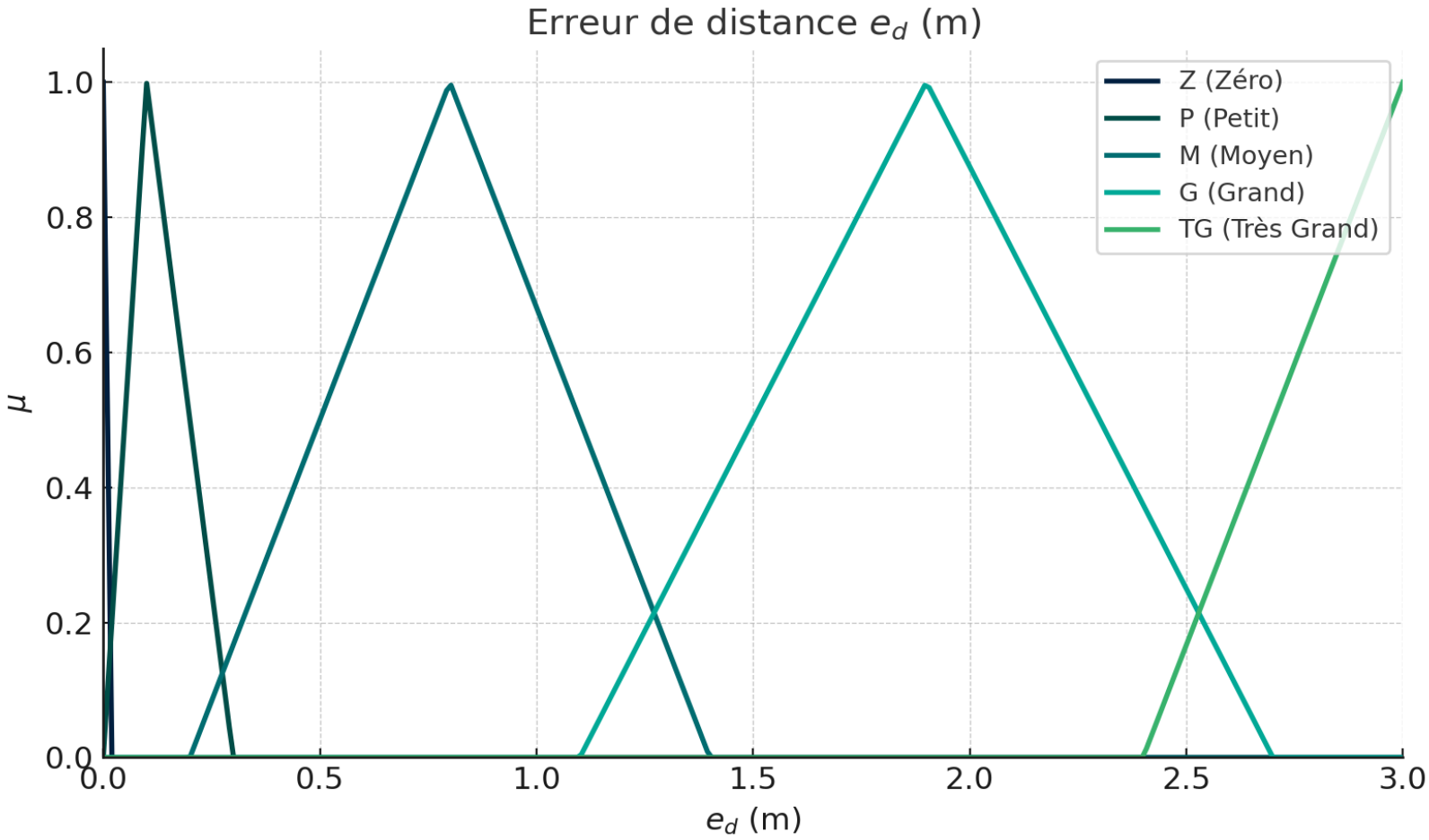


Figure 1.22

Stratégie de commande: Logique floue

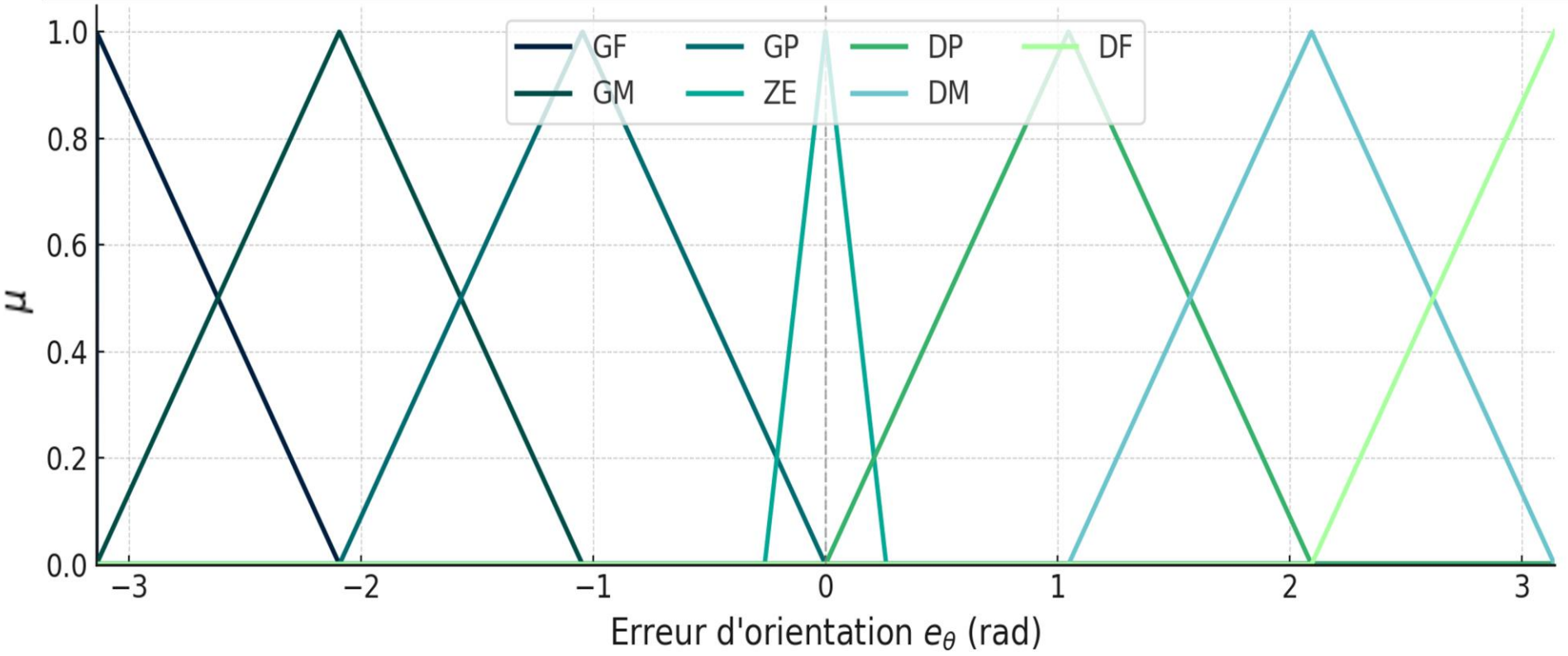


Figure 1.23

Stratégie de commande: Logique floue

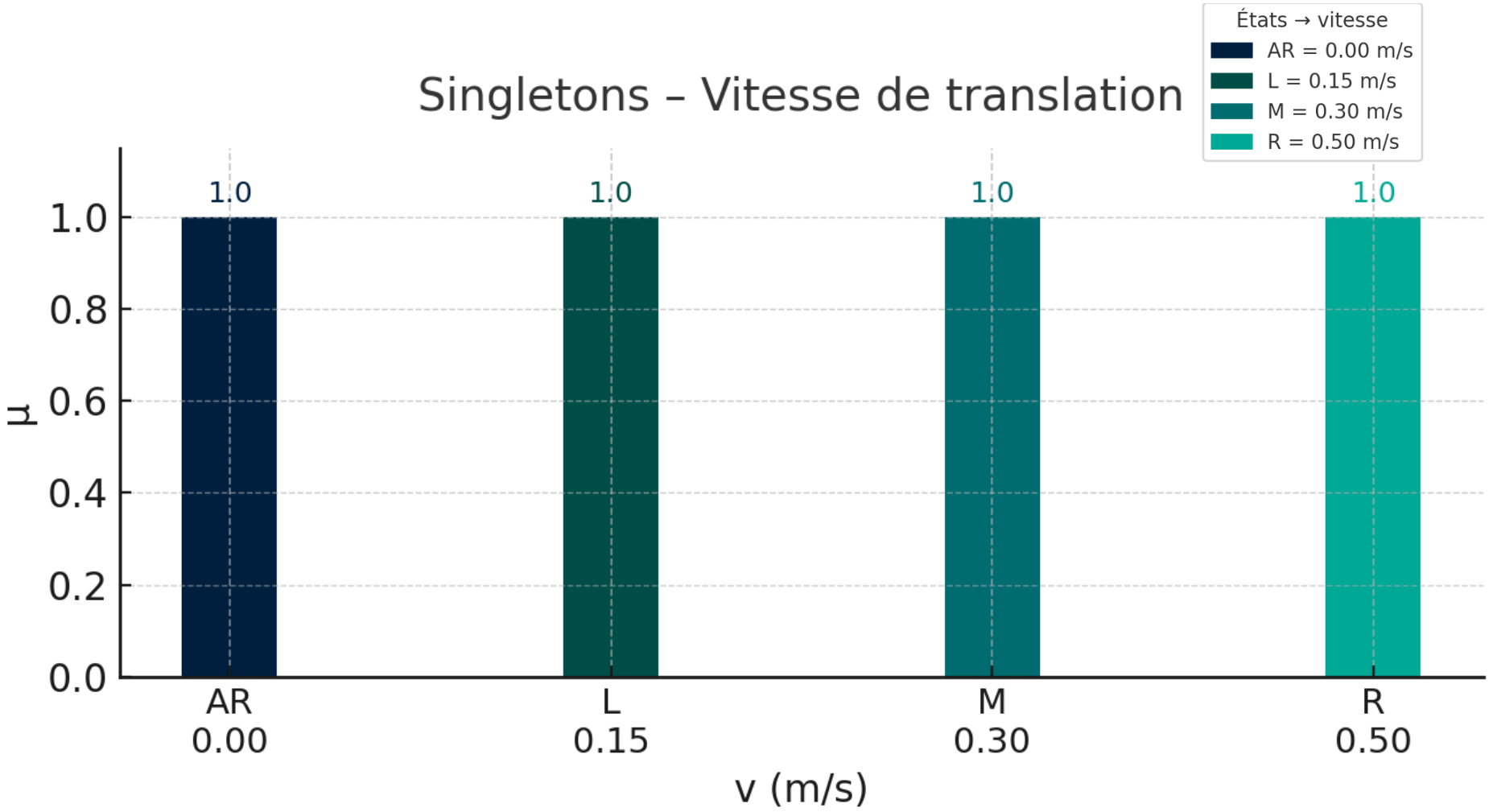


Figure 1.24

Stratégie de commande: Logique floue

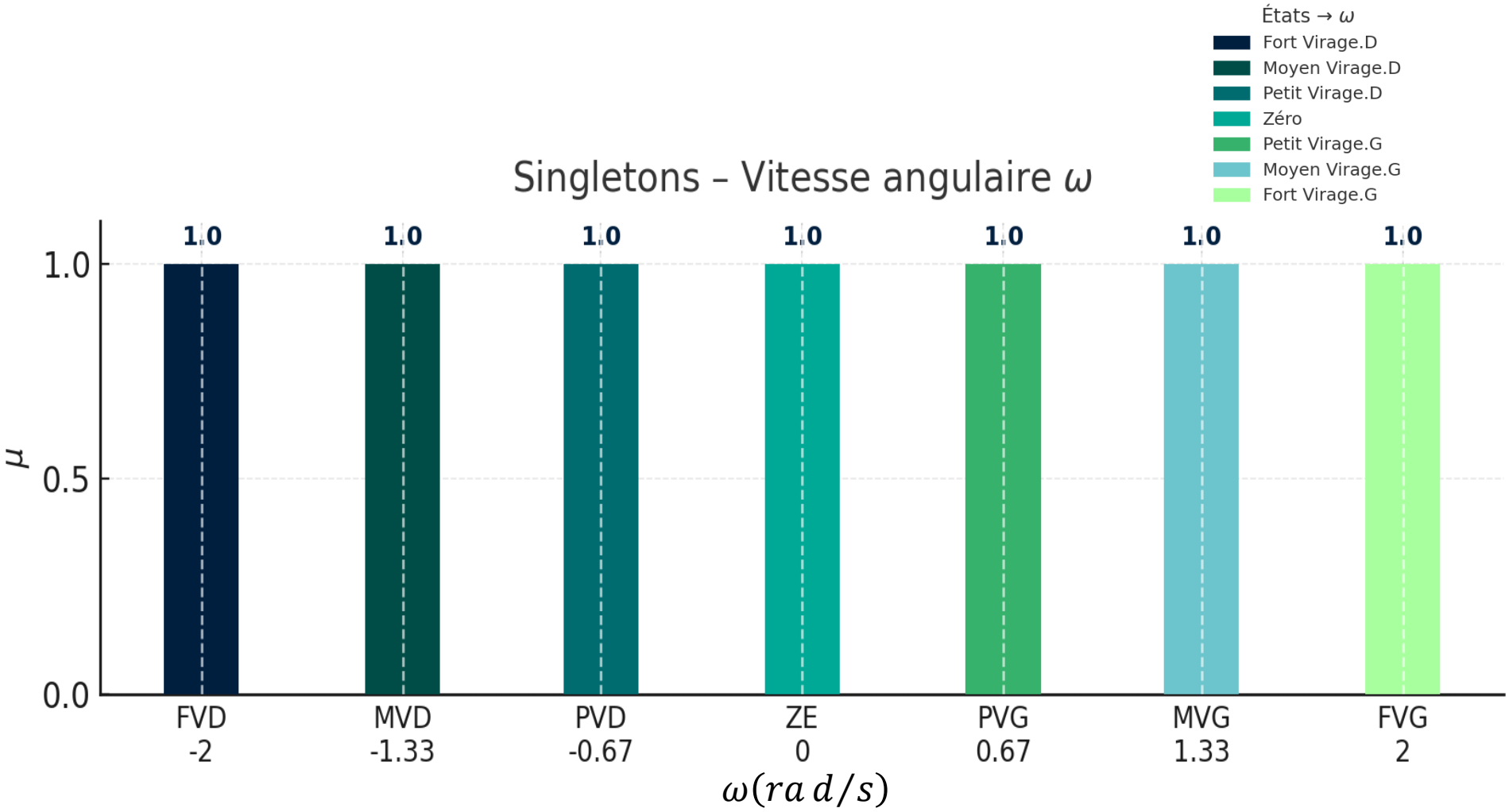


Figure 1.25

Stratégie de commande: Logique floue

5. Tableau de règles:

e_θ \ e_d	Z	P	M	G	TG
GF	AR/ZE	L/FVD	L/FVD	M/FVD	M/FVD
GM	AR/ZE	L/MVD	L/MVD	M/MVD	M/MVD
GP	AR/ZE	L/PVD	M/PVD	R/PVD	R/PVD
ZE	AR/ZE	L/ZE	M/ZE	R/ZE	R/ZE
DP	AR/ZE	L/PVG	M/PVG	R/PVG	R/PVG
DM	AR/ZE	L/MVG	L/MVG	M/MVG	M/MVG
DF	AR/ZE	L/FVG	L/FVG	M/FVG	M/FVG

Figure 1.26

6. Qu'est ce que le degré d'activation?

- On suppose que : $\mu_M(e_d)$=0,6 ; $\mu_{GF}(e_\theta)$=0,8
- R1 : SI e_d est M(moyenne) ET e_θ est GF(gauche forte) Alors avance lentement et Fort virage à droite
- le degré d'activation de la règle R1: α = min(0,6 ; 0,8) = 0,6

Stratégie de commande: Logique floue

7. Exemple - atteindre (1 m, 1 m):

- Point de départ : $(x_r, y_r, \theta) = (0,0,0°)$
- Point cible : $(x_C, y_C) = (1,1)$
- But : calculer en temps réel les commandes (v,ω) pour que le robot rejoigne (1, 1)

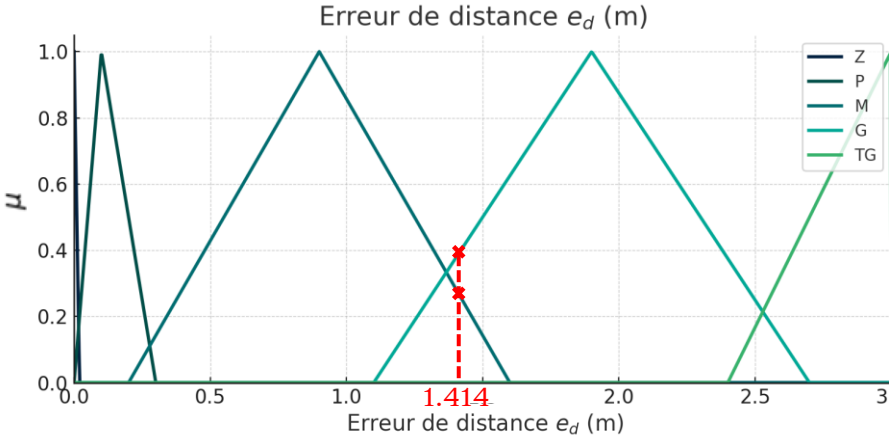
❖ Étape 1 : Calcul des erreurs

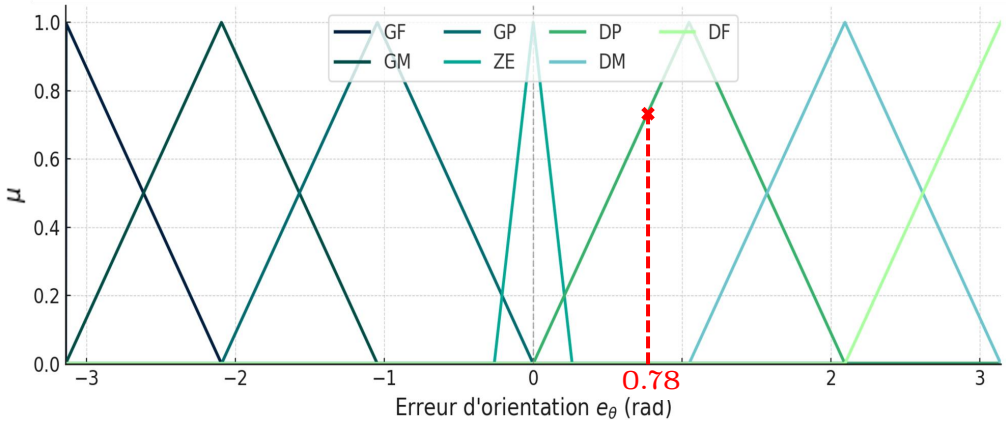
Erreur de distance: $e_d = \sqrt{(1-0)^2 + (1-0)^2} = \sqrt{2} \approx 1.414\ m$

Erreur d'orientation: $\begin{cases} \phi = a\tan 2(1 - 0{,}1\ - 0) = 45° \\ e_\theta = wrap(\phi - \theta_a) = 45° = 0.78\ rad \end{cases}$

Stratégie de commande: Logique floue

❖ Étape 2 : Fuzzification





Ensemble flou	μ
Z	0.0000
P	0.0000
M	0.2657
G	0.3925
TG	0.0000

Ensemble flou	μ
GF	0.000
GM	0.000
GP	0.000
ZE	0.000
DP	0.750
DM	0.000
DF	0.000

Figure 1.27

Stratégie de commande: Logique floue

❖ Étape 3 : Base de règles.

- On a $\mu_M(e_d)$=0.2657, $\mu_G(e_d)$=0.3925 et $\mu_{DP}(e_\theta)$=0.750

Règles possibles :

- R1: SI e_d est Grande ET e_θ est Droite-Petite ALORS v_r est rapide ET ω_r est petit virage à à gauche.

- R2: SI e_d est Moyenne ET e_θ est Droite-Petite ALORS v_r est moyenne ET ω_r est petit virage à à gauche.

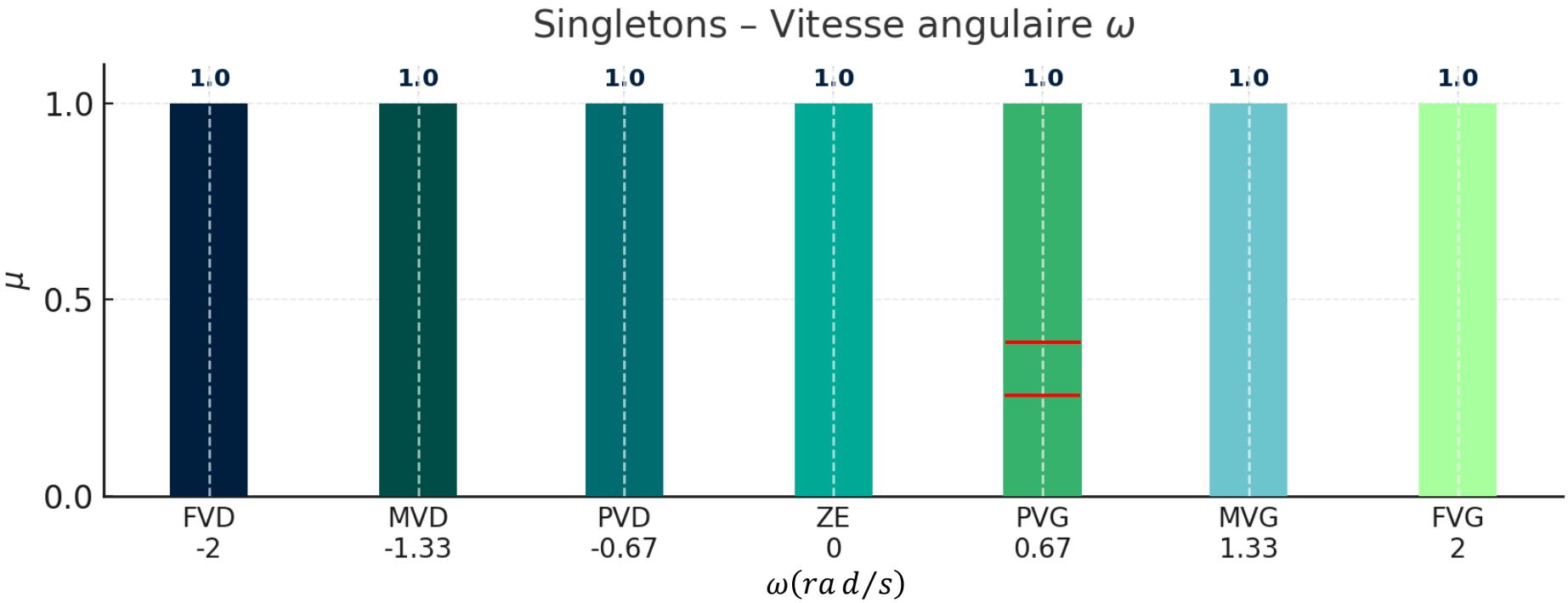
❖ Étape 4 :Inférence.

- $\alpha(R_1) = \min\left(\mu_G(e_d), \mu_{DP}(e_\theta)\right) = 0.3925$

- $\alpha(R_2) = \min\left(\mu_M(e_d), \mu_{DP}(e_\theta)\right) = 0.2657$

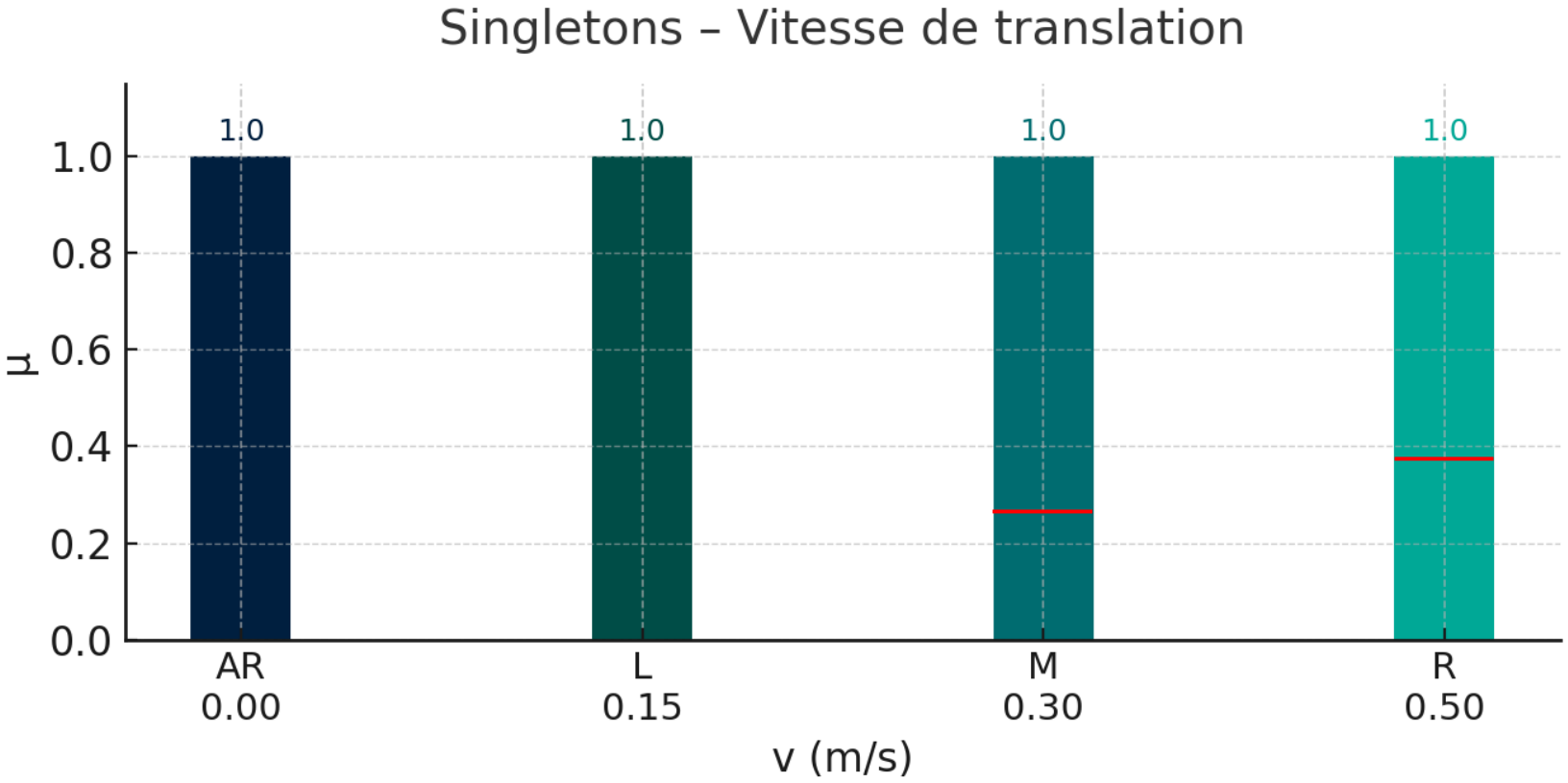
Stratégie de commande: Logique floue

❖ Étape 5 : Défuzzification



$$\omega = \frac{0{,}3925 \times 0{,}67 + 0{,}2657 \times 0{,}67}{0{,}3925 + 0{,}2657} = 0{,}67\ rad/s$$

Stratégie de commande: Logique floue



$$v_r = \frac{0{,}3925 \times 0{,}50 + 0{,}2657 \times 0{,}30}{0{,}3925 + 0{,}2657} \approx 0.42\ m/s$$

Stratégie de commande: Logique floue

8. Création du contrôleur flou (type Sugeno):

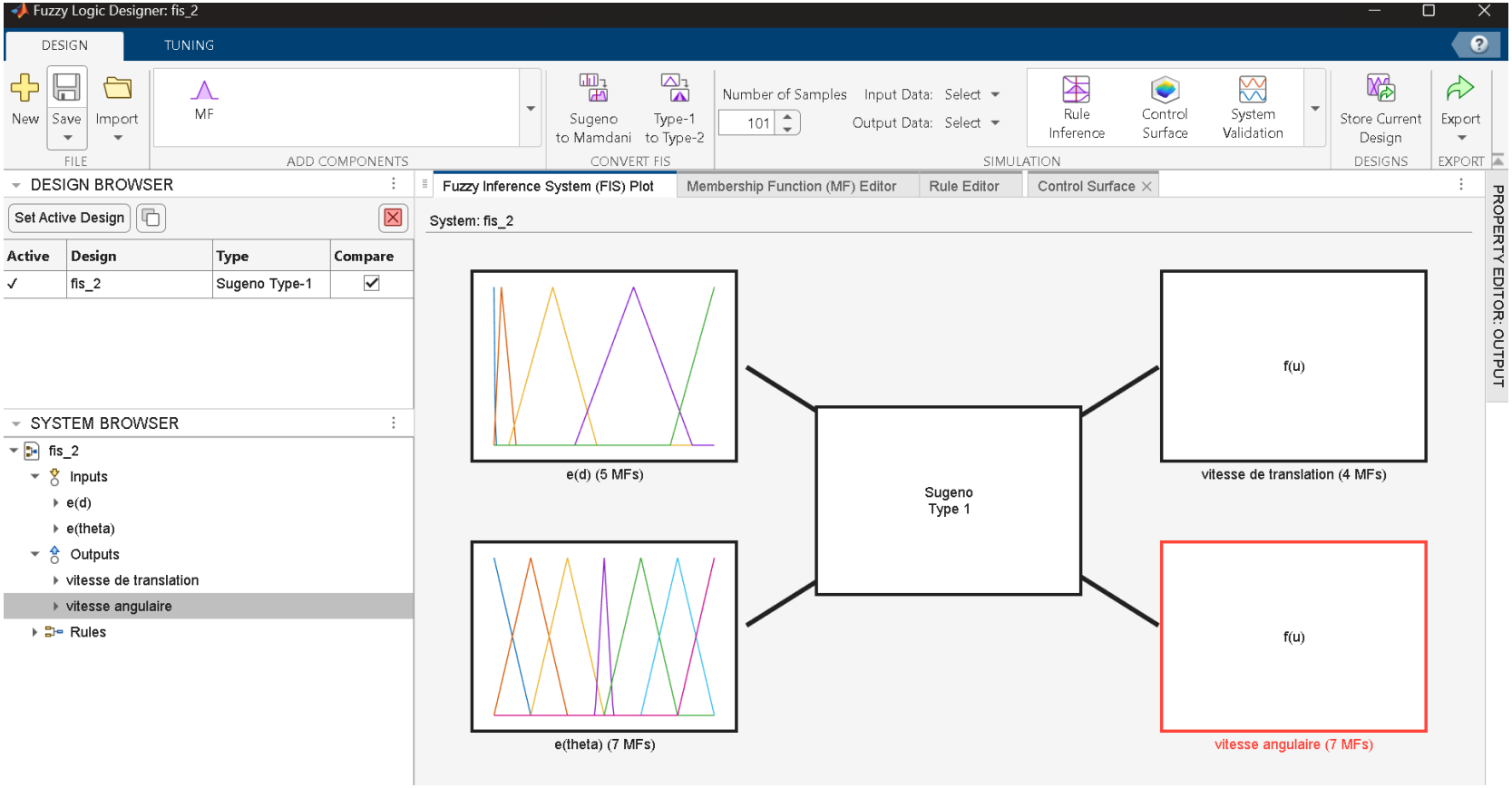


Figure 1.28

Stratégie de commande: Logique floue

9. Validation de la commande par Matlab:

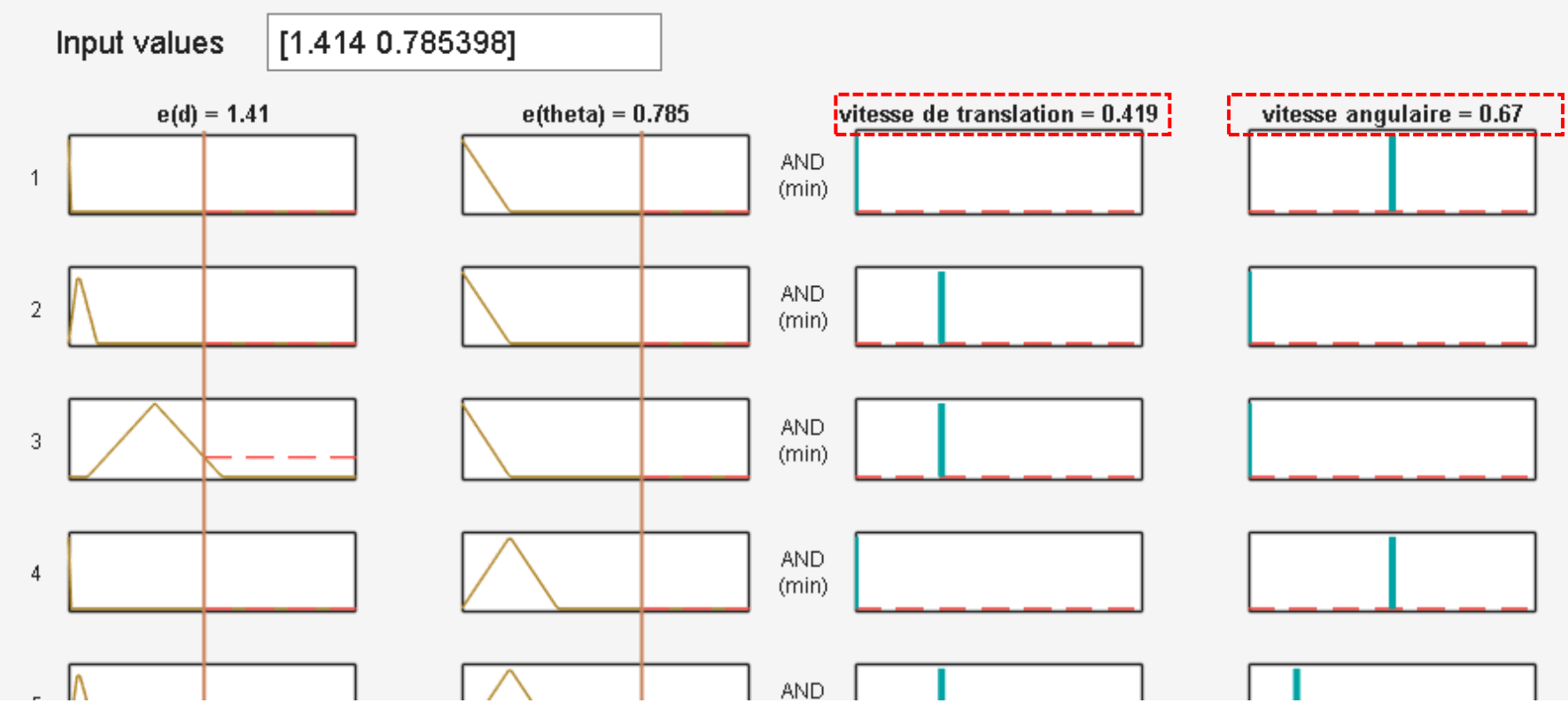


Figure 1.29

Stratégie de commande: Logique floue

10. Surface de contrôle (vitesse de translation):

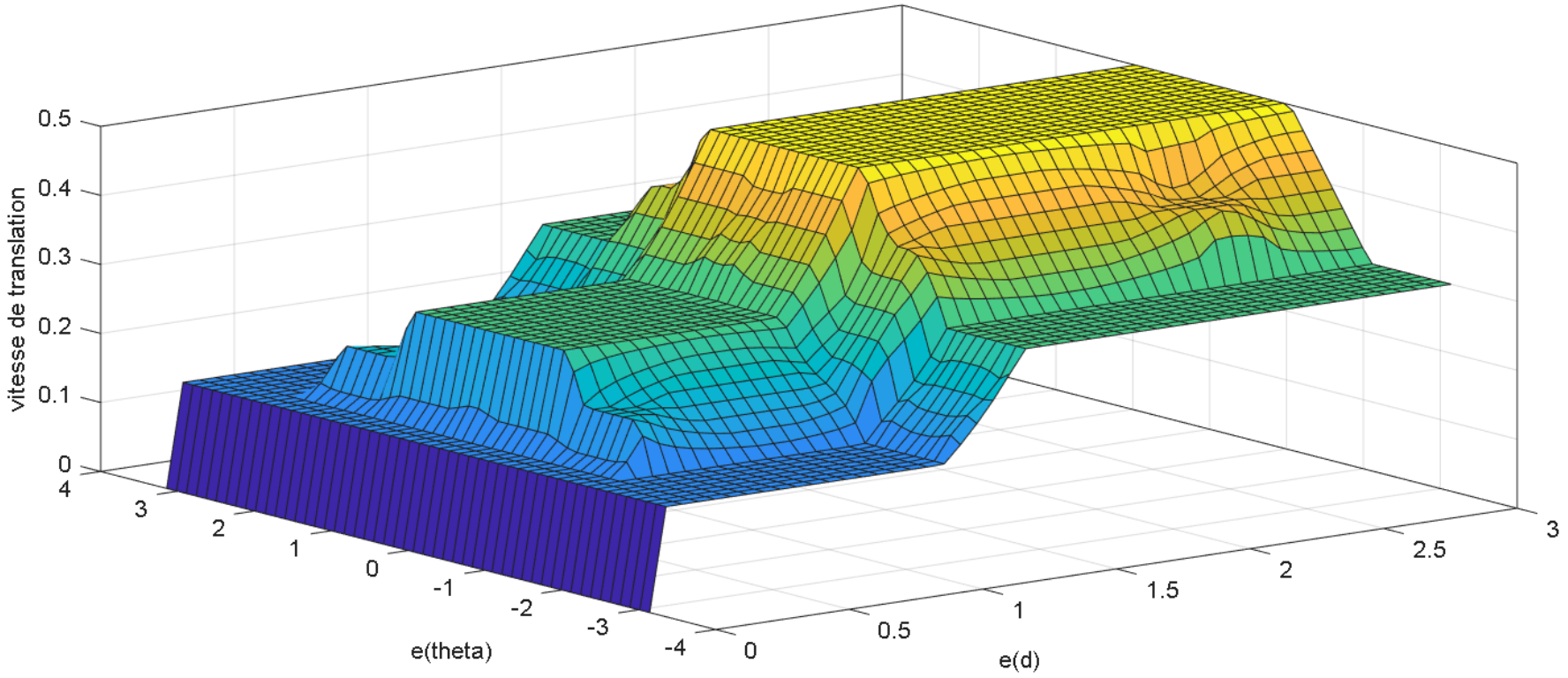


Figure 1.30

- **Translation** : la vitesse est automatiquement réduite quand l'écart d'orientation est important, évitant ainsi que le robot n'avance trop vite s'il n'est pas aligné.

Asservissement en vitesse

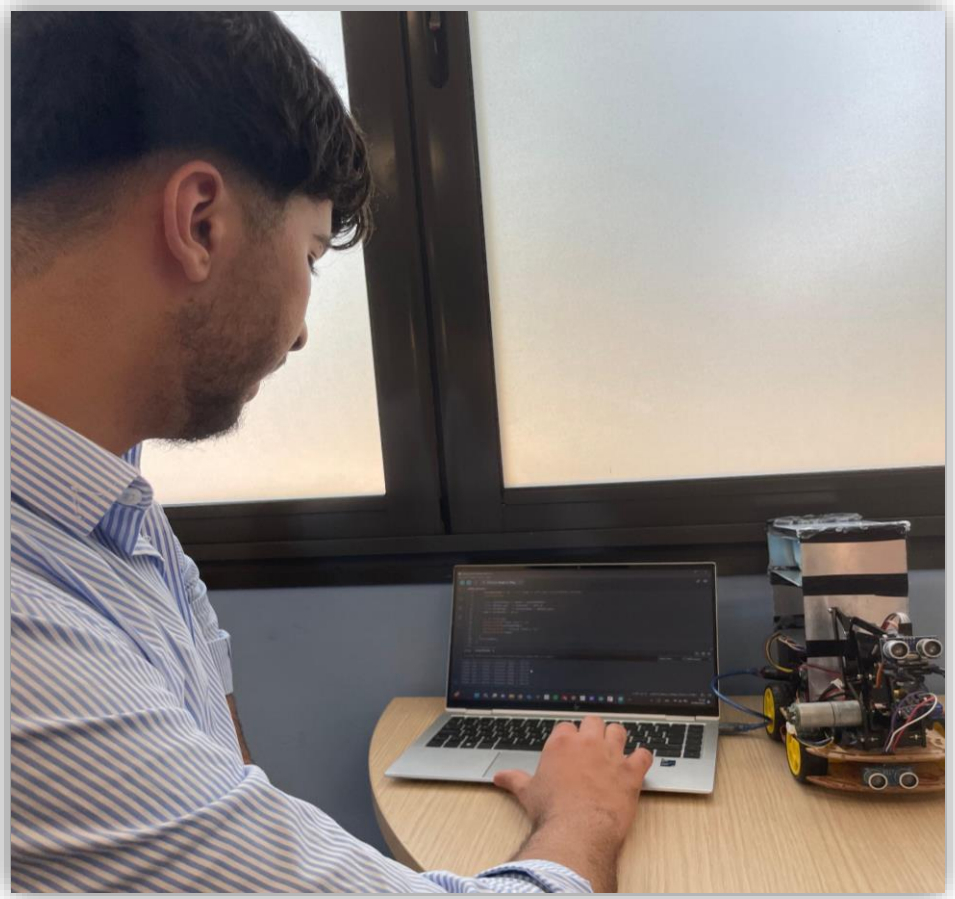
1. Exigences:

Critère	Valeur cible
Rapidité	Un temps de réponse Tr5% ≤ 0,5 s
Stabilité	Mφ ≥ 45° et aucun dépassement
Précision	$\varepsilon_s = 0$

Figure 1.31

Asservissement en vitesse

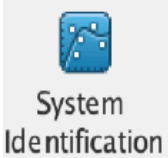
2. Modèle de comportement des moteurs sous charge:

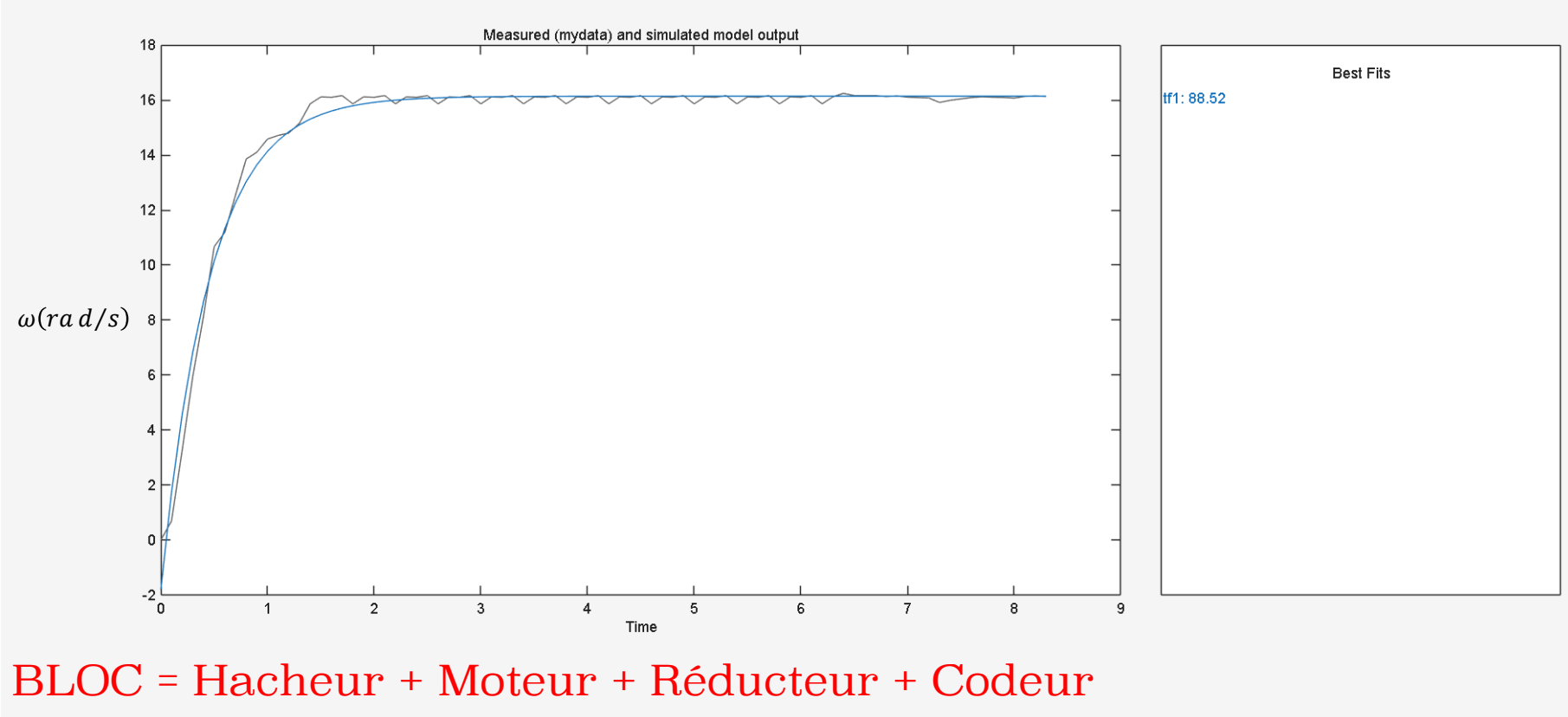


Expérience : mesure de la vitesse de l'arbre réducteur d'un moteur pour PWM= 255

Asservissement en vitesse

Identification du BLOC comme système du 1er ordre à l'aide de :





BLOC = Hacheur + Moteur + Réducteur + Codeur

Figure 1.32

Asservissement en vitesse

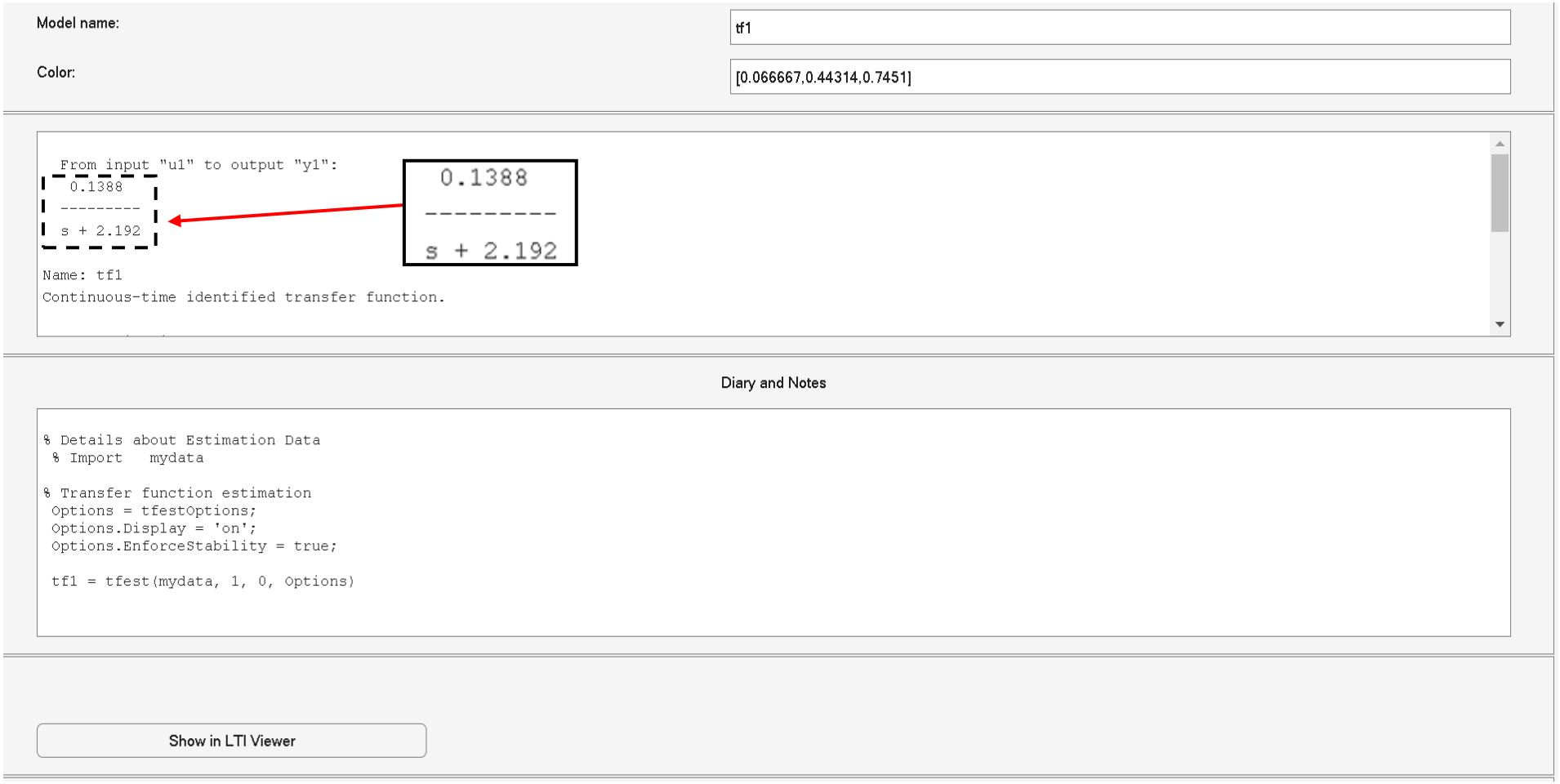


Figure 1.33

Asservissement en vitesse

3. Performances du système avant correction:

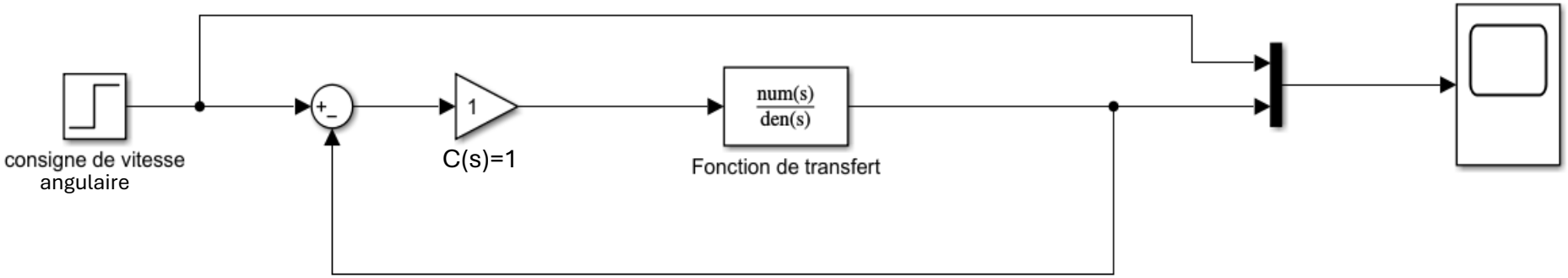
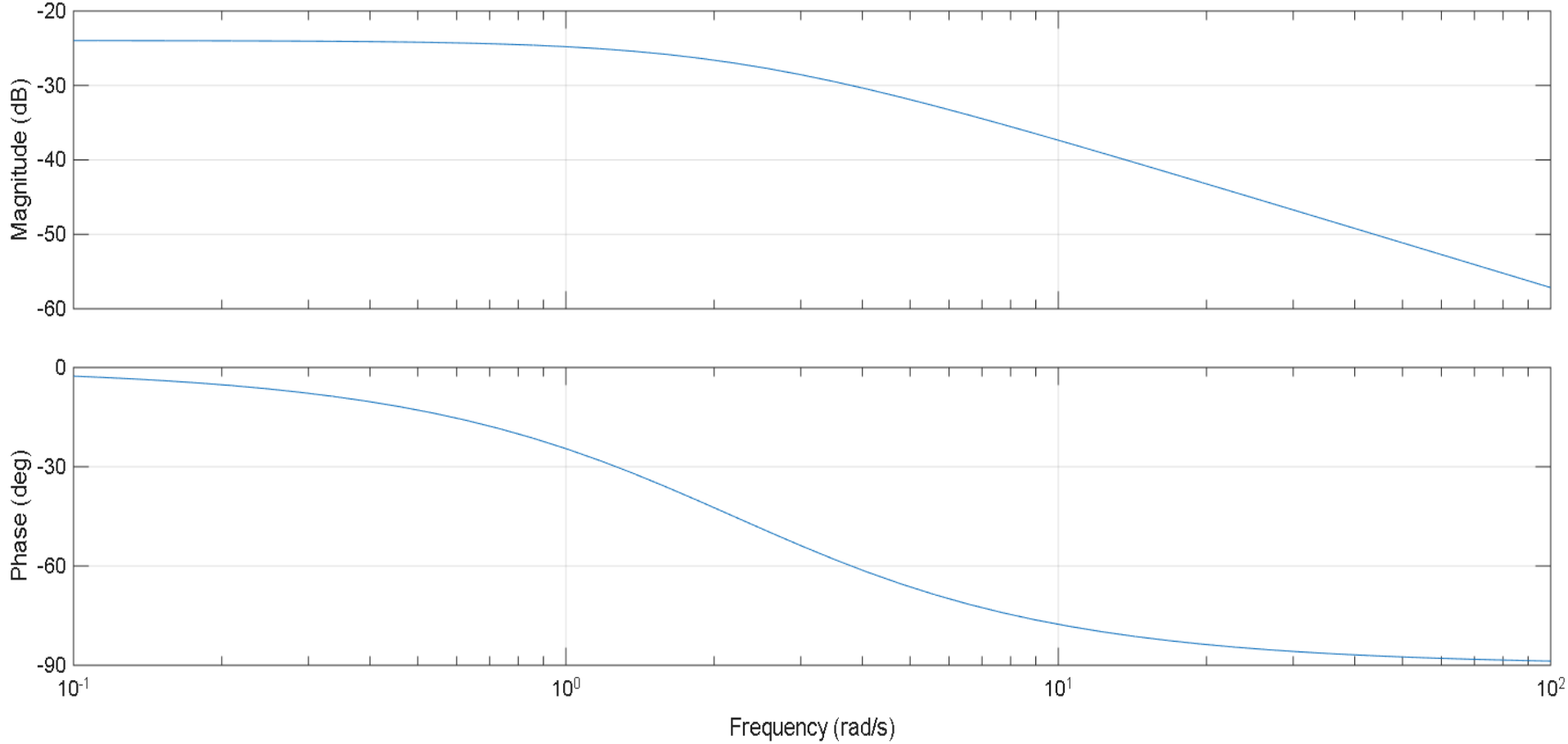


Figure 1.34: Schéma bloc du système non corrigé

Asservissement en vitesse

Figure 1.35: Diagrammes de Bode de la FTBO du système non corrigé



- MG: $+\infty$, Mφ : $+\infty$ $\Rightarrow$ Le système est stable

Asservissement en vitesse

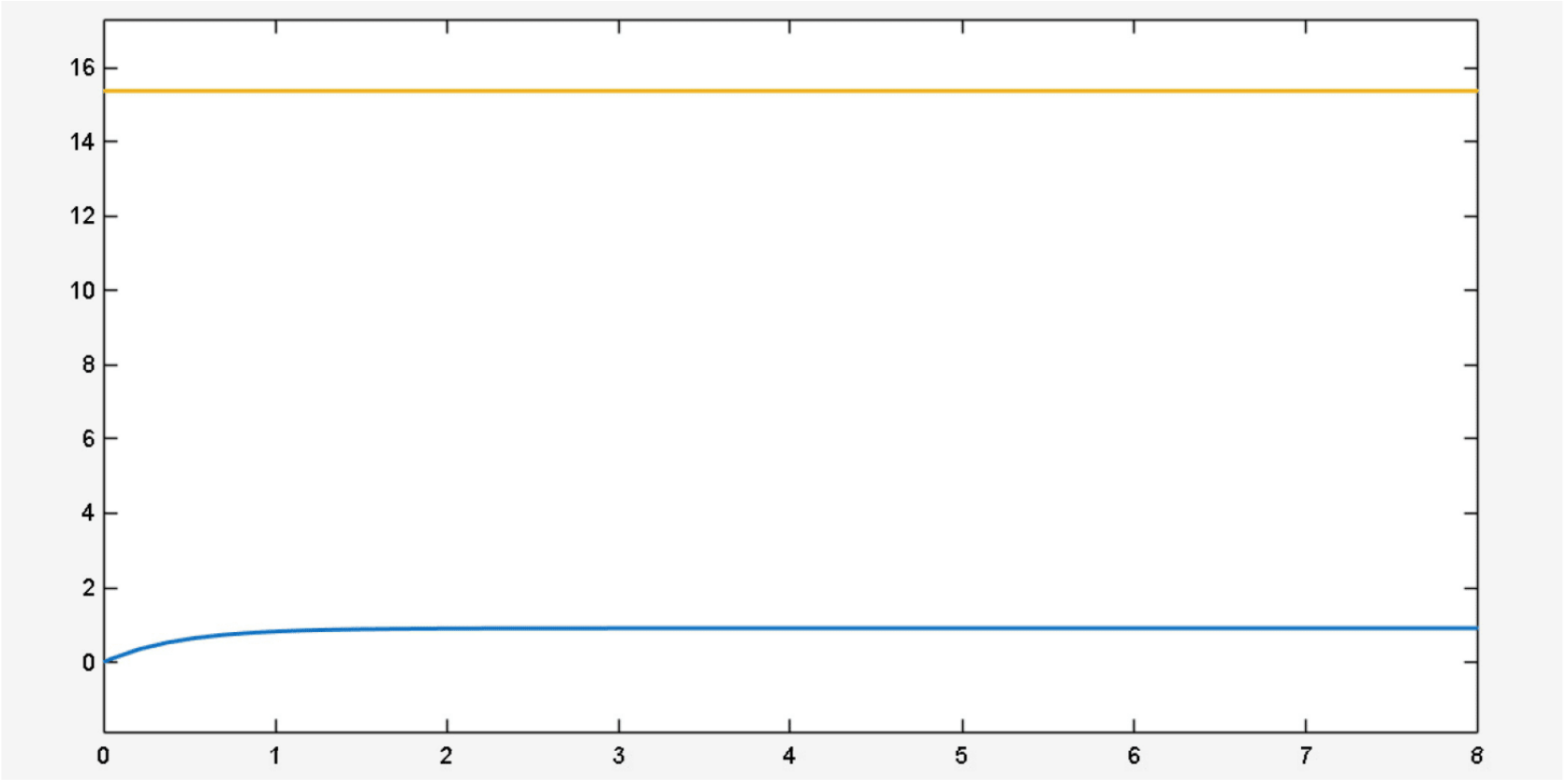


Figure 1.36: Réponse à un échelon de vitesse angulaire de valeur : 15.38 rad/s

Asservissement en vitesse

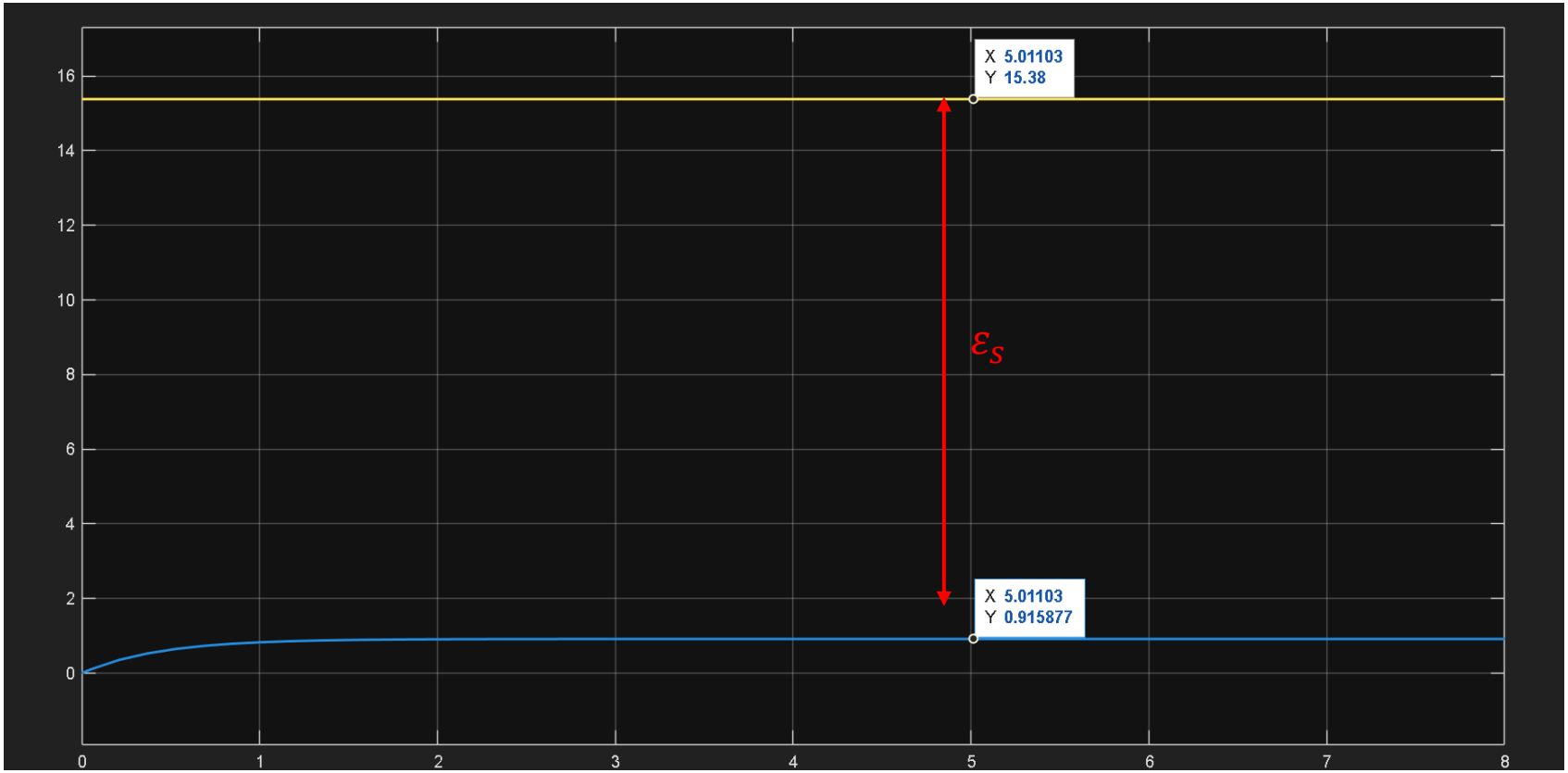


Figure 1.37

$$\varepsilon_s = \frac{15{,}38 - 0{,}91}{15{,}38} \times 100 = 94\ \% \Rightarrow \varepsilon_s$$ ne respecte pas l'exigence sur la précision

Asservissement en vitesse

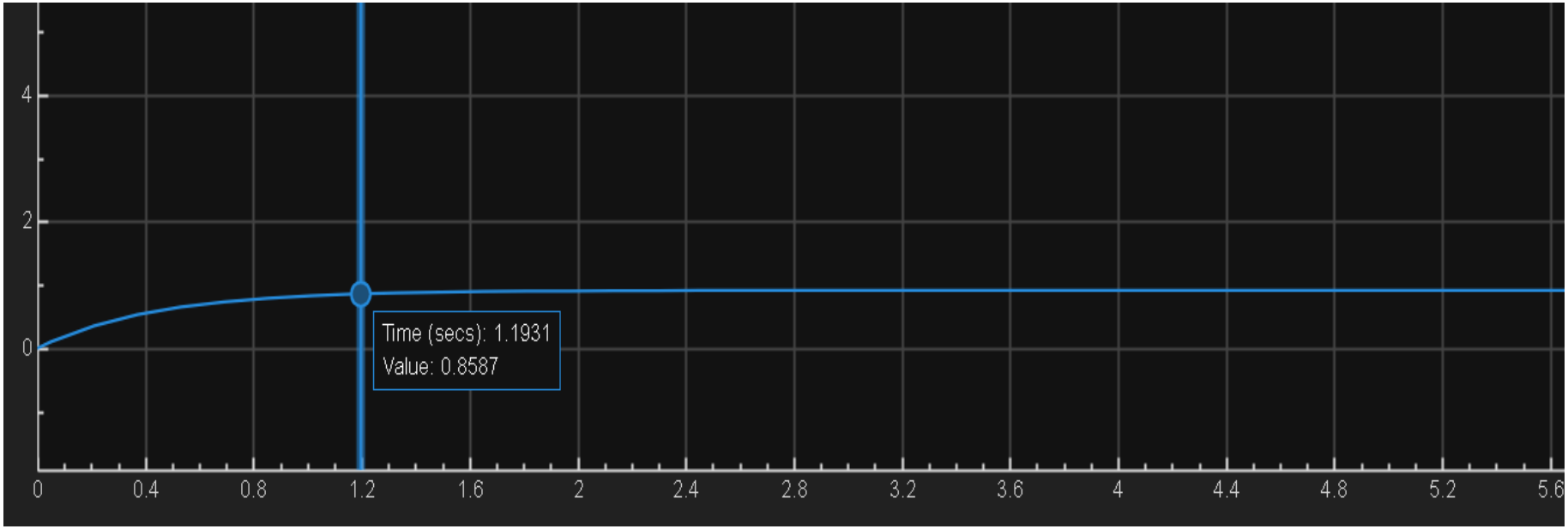


Figure 1.38

On a :$0{,}95 \times 0{,}91 = 0.86$ rad/s $\Rightarrow t_{r5\%} \approx 1.20$ s

$\Rightarrow t_{r5\%}$ ne respecte pas l'exigence sur la rapidité

Asservissement en vitesse

4. Correction:

La fonction de transfert de notre système est : $H(P) = \frac{0,1388}{P+2,192}$

Nous optons pour un correcteur PI : $C(P) = K_p\left(1 + \frac{I}{P}\right)$

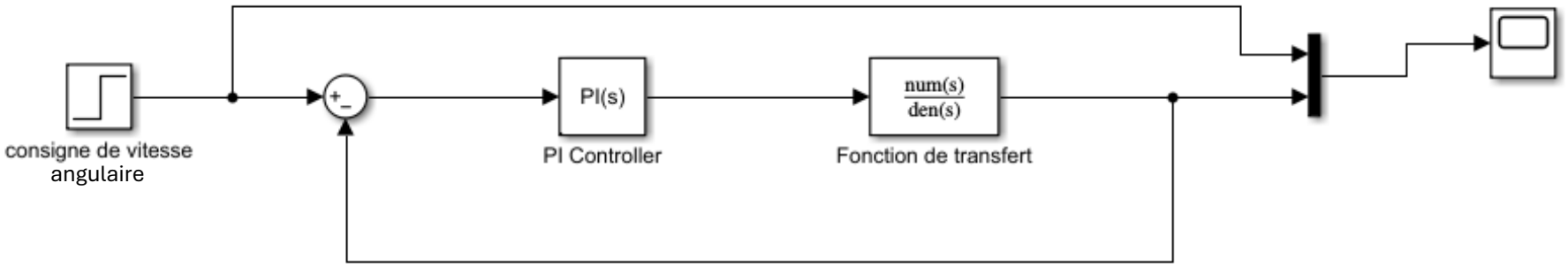


Figure 1.39

Asservissement en vitesse

La valeur de I :

On utilise la méthode de compensation du pôle dominant: $I = 2{,}192$

$$\Rightarrow FTBO(P) = H(P) \times C(P) = \frac{0{,}1388}{P+2{,}192} \times K_p \left(\frac{P+2{,}192}{P}\right) = \frac{0{,}1388 \times K_p}{P}$$

La présence d'une action intégrale dans la FTBO supprime l'erreur statique ε_s

$\Rightarrow$ Notre système corrigé respecte le cahier des charges en termes de précision.

Asservissement en vitesse

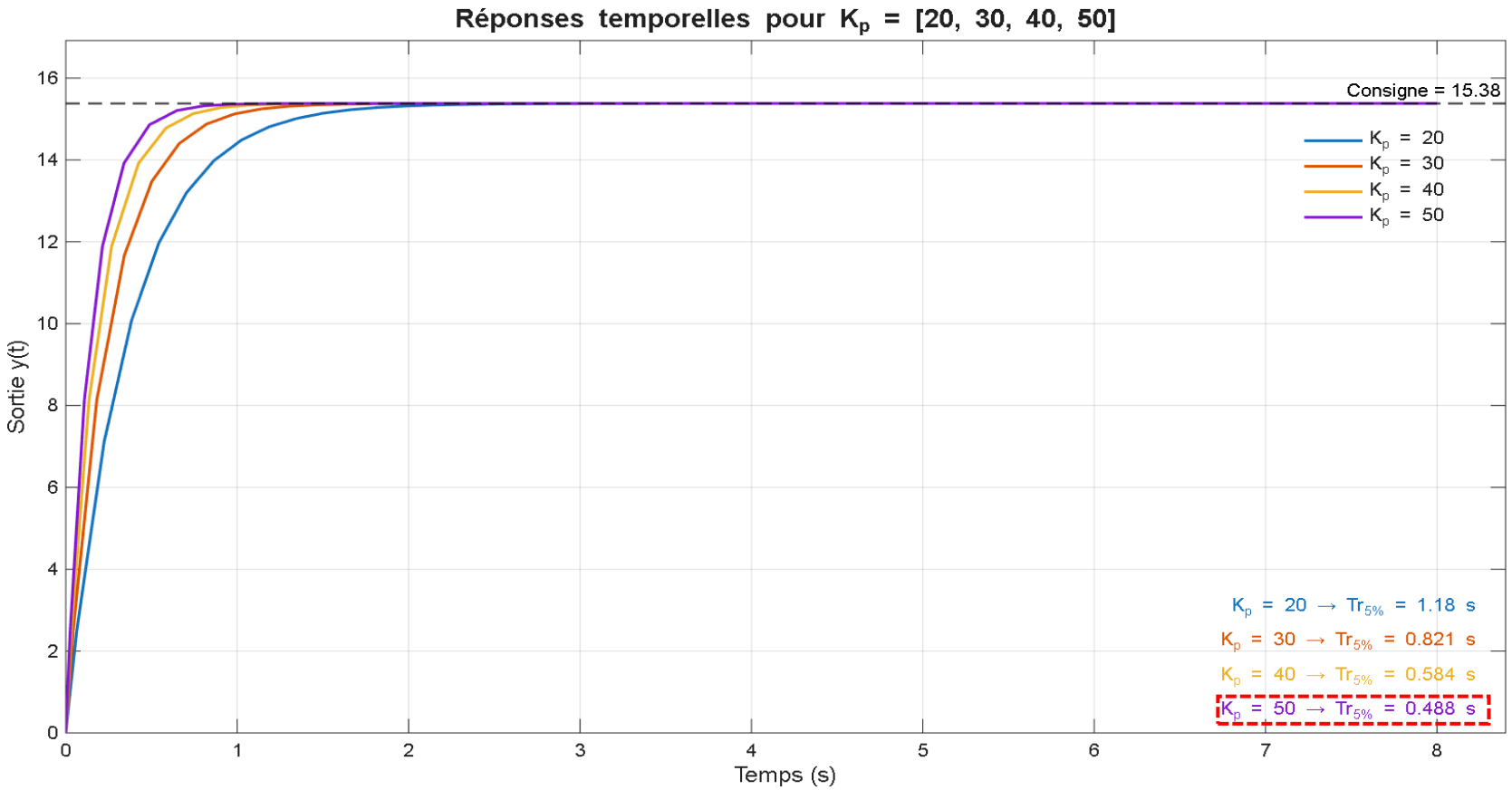


Figure 1.40

Pour K_p=50 : $t_{r5\%} = 0.488\ s < 0.5\ s$ $\Rightarrow$ On choisit K_p=50

$\Rightarrow$ Notre système corrigé respecte le cahier des charges en termes de rapidité.

Asservissement en vitesse

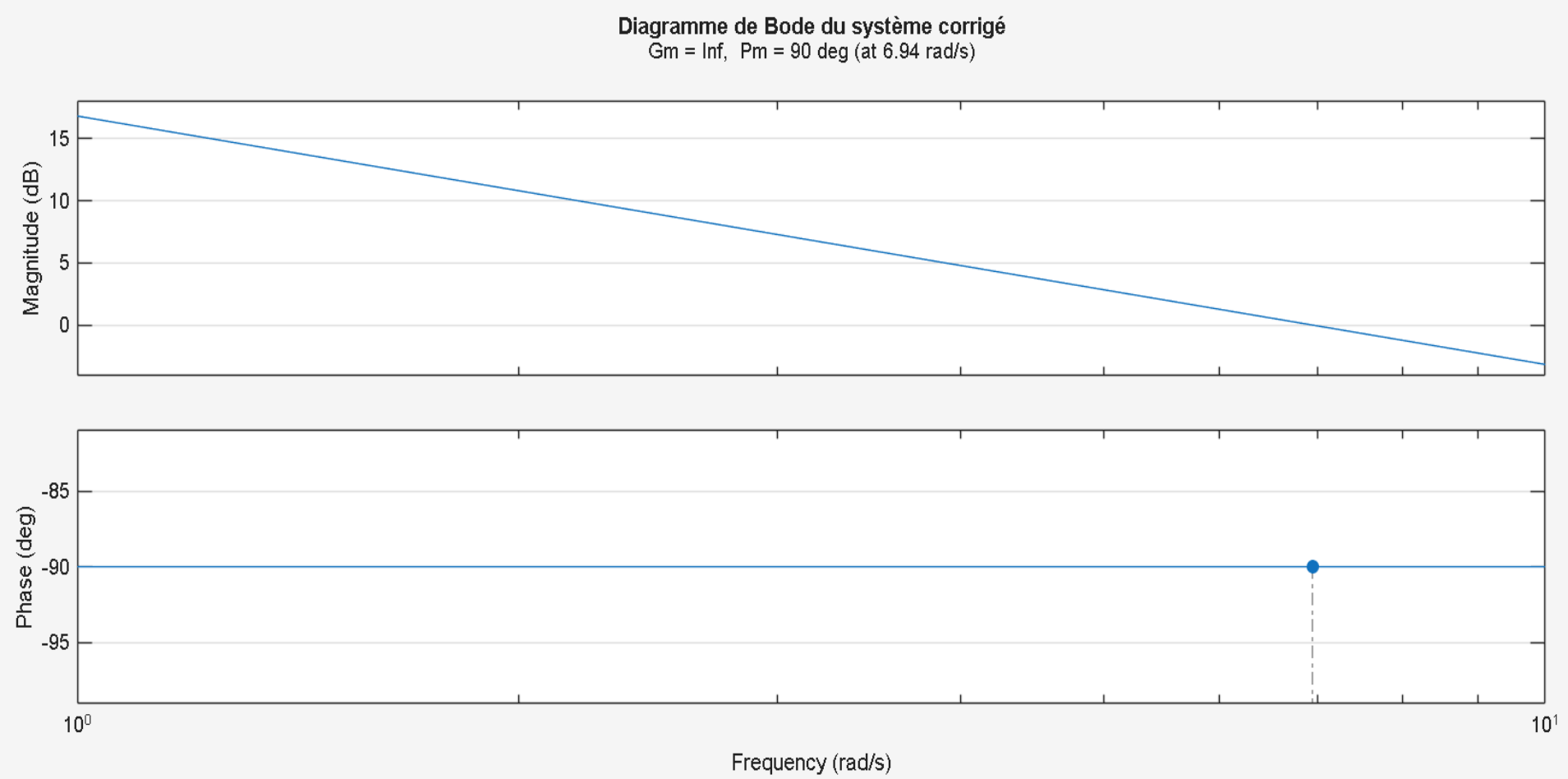


Figure 1.41

$M\varphi = 90^{\circ} > 45°$ $\Rightarrow$ Notre système corrigé respecte le cahier des charges en termes de stabilité.

Simulation sous MATLAB

1. Schéma sur Simulink:

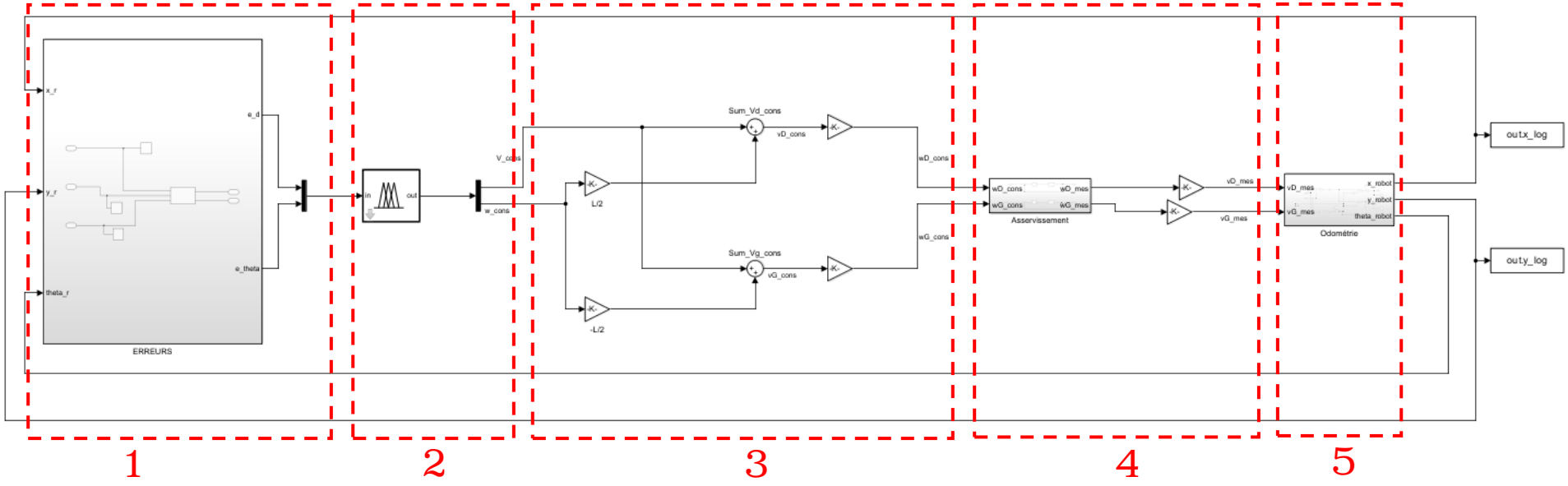


Figure 1.42

Simulation sous MATLAB

2. Résultats de la simulation:

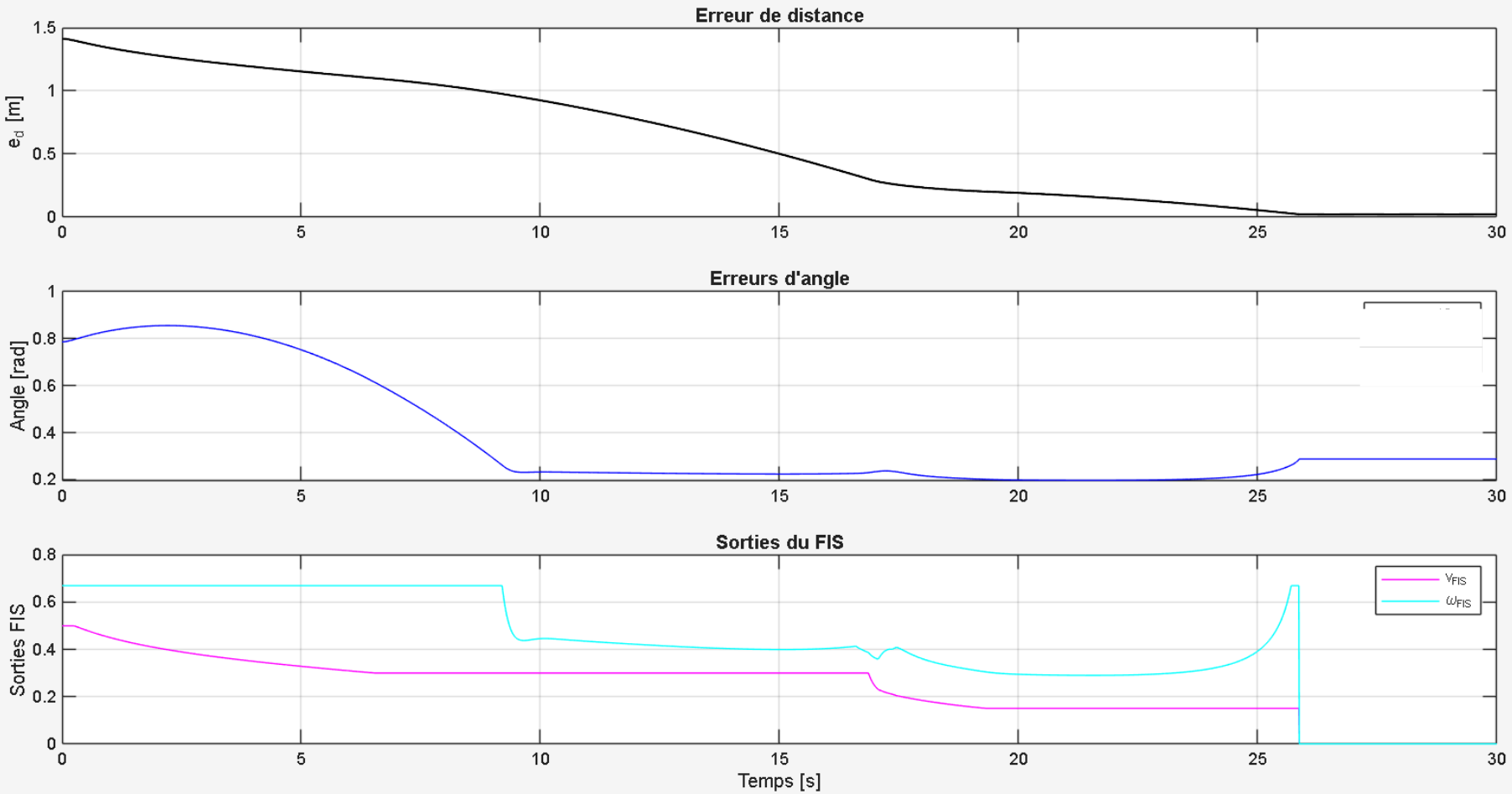


Figure 1.43

Simulation sous MATLAB

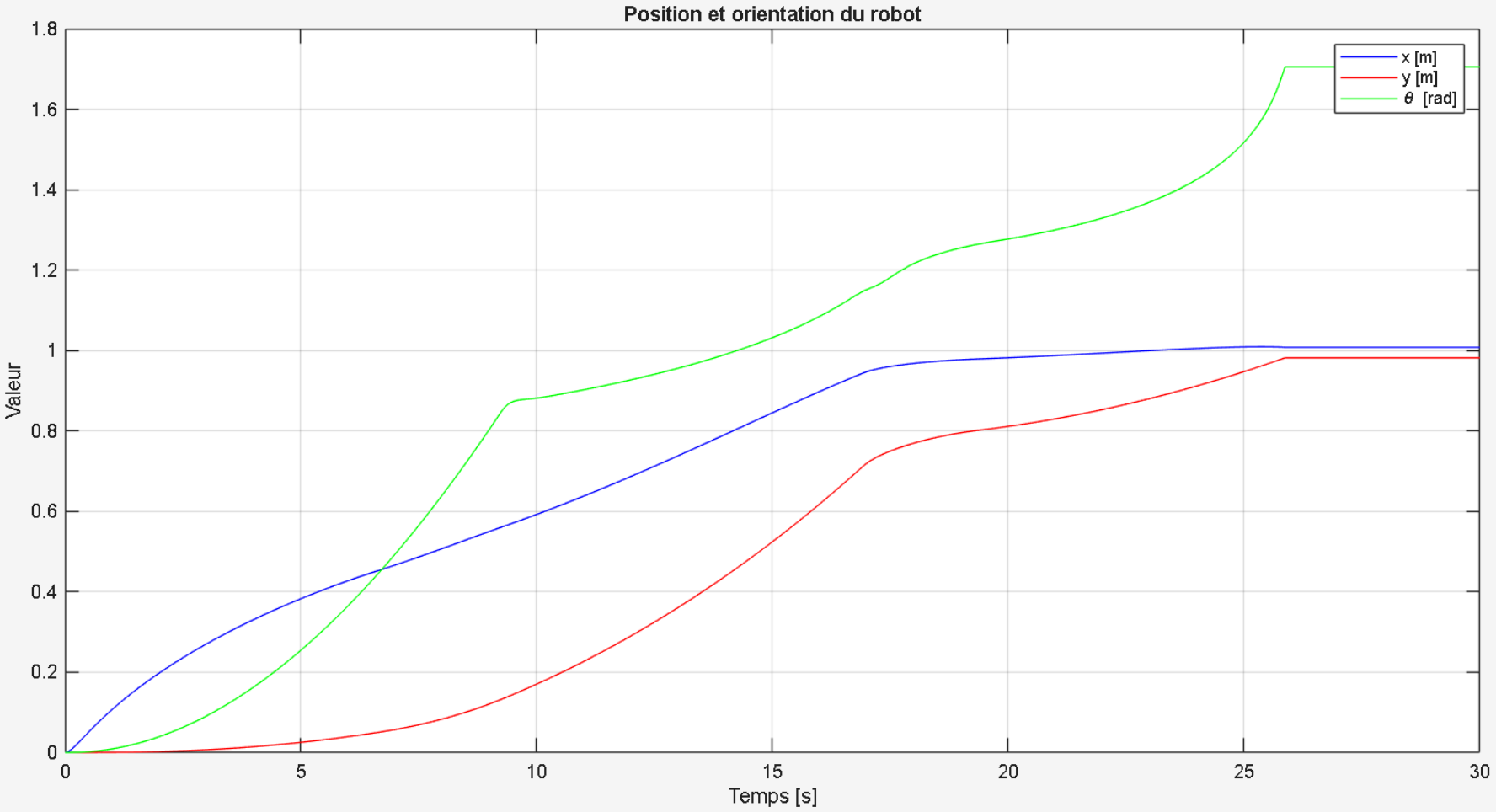


Figure 1.44

Simulation sous MATLAB

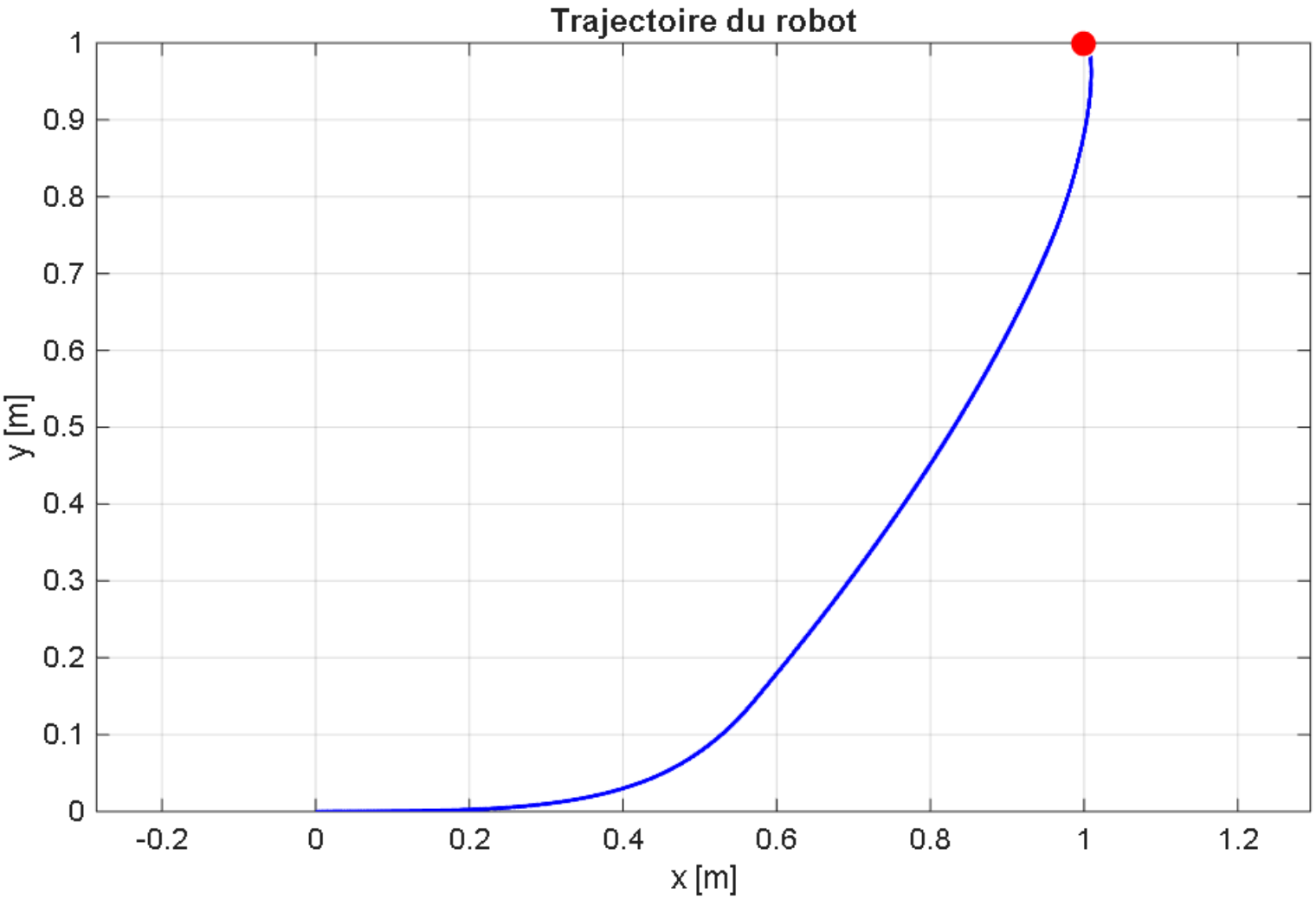


Figure 1.45

2ème objectif:

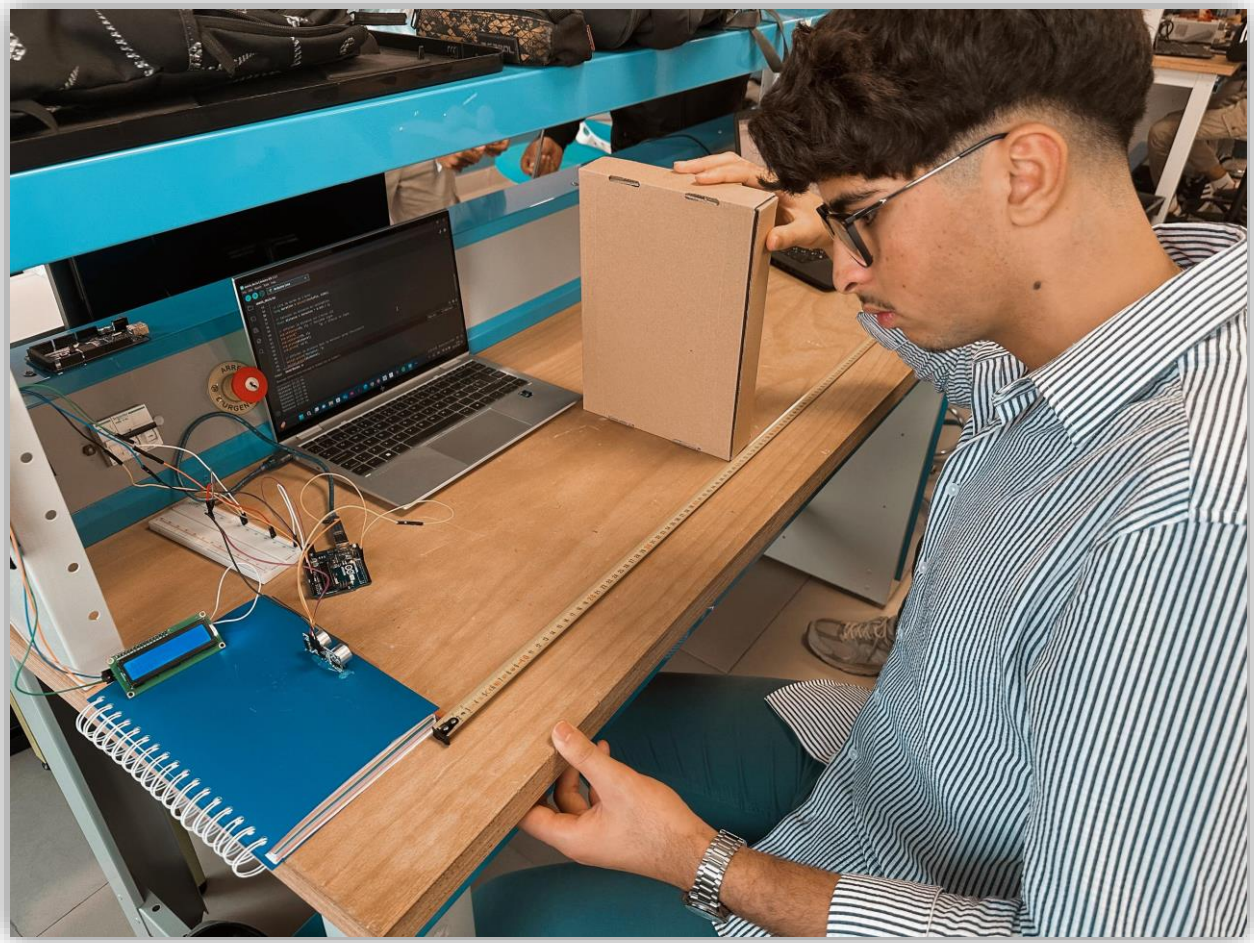
Choix du capteur de détection d'obstacles.

HC-SR04

ou

Sharp GP2Y0A02YK0F

HC-SR04



Expérience : mesure de la distance à un obstacle à l'aide du capteur ultrasonique HC-SR04.

HC-SR04

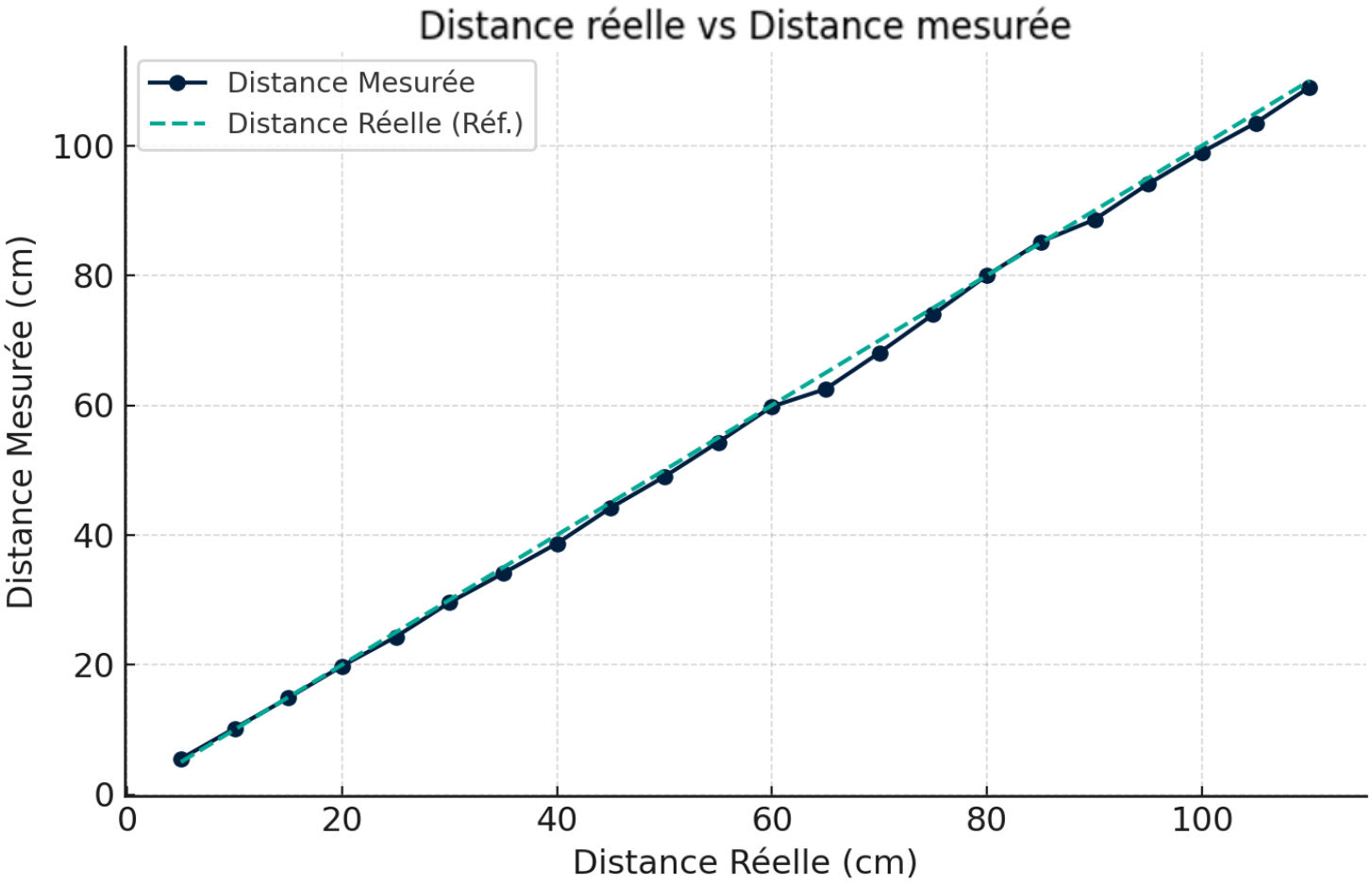


Figure 2.1

⇒ La superposition quasi parfaite des mesures et de la droite de référence montre la linéarité du capteur entre 5 et 110 cm.

HC-SR04

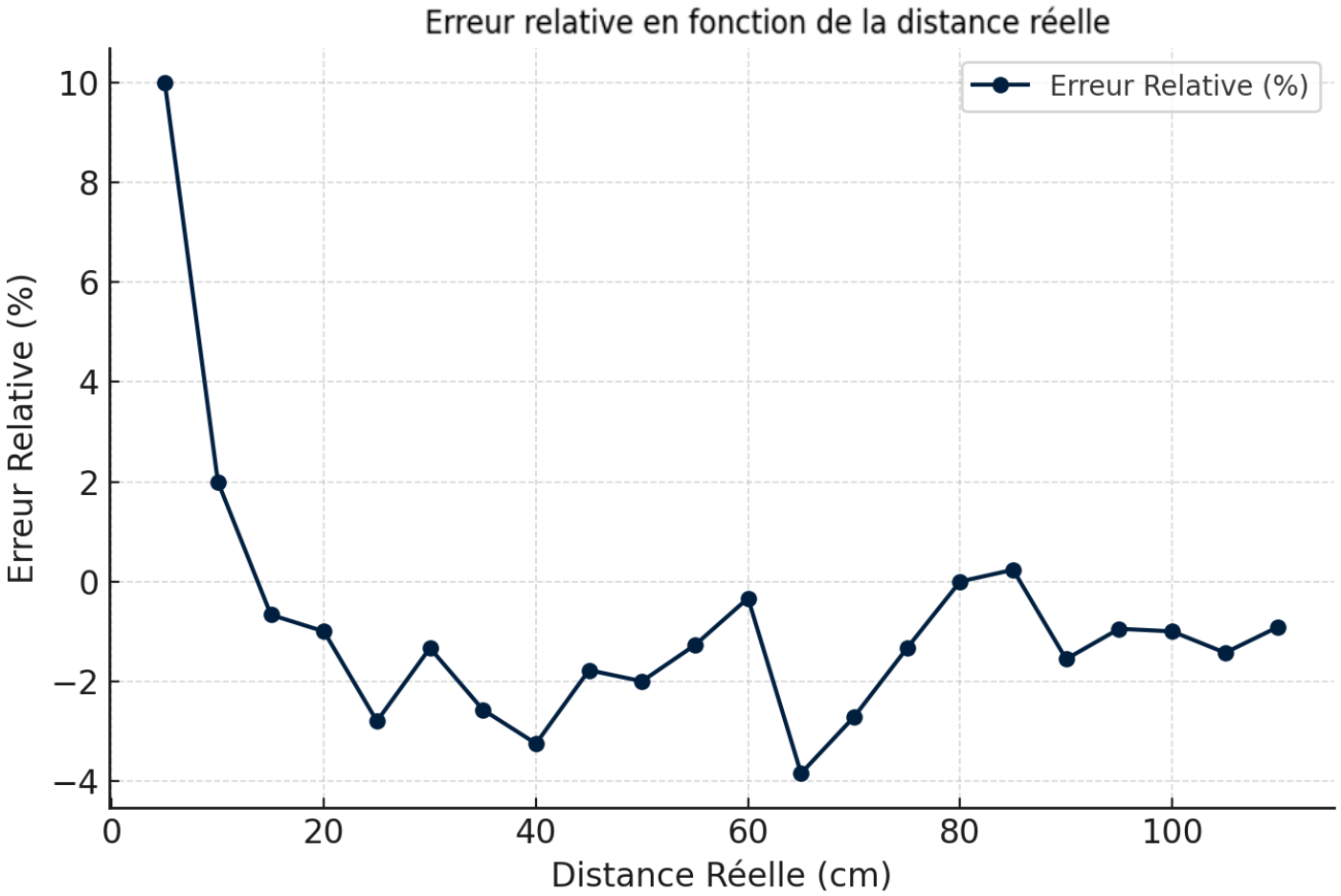


Figure 2.2

⇒ Au-delà de 10 cm, l'erreur relative ne dépasse pas 4 % en valeur absolue

Sharp

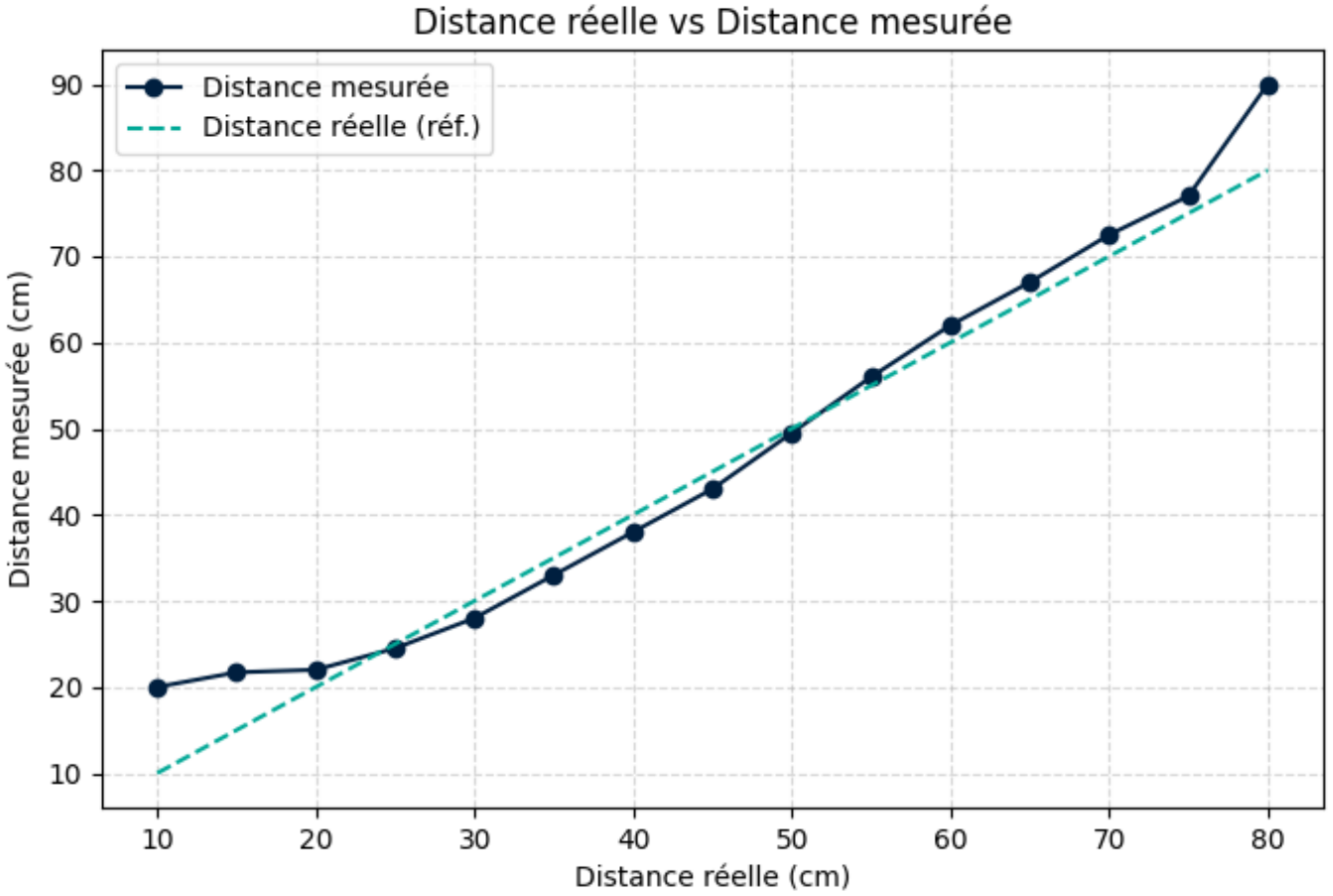


Figure 2.3

⇒ Le capteur surévalue nettement sous 20 cm, présente une légère sous-estimation (erreur <5 %) entre 25 et 50 cm, puis surestime de nouveau au-delà de 55 cm.

Sharp

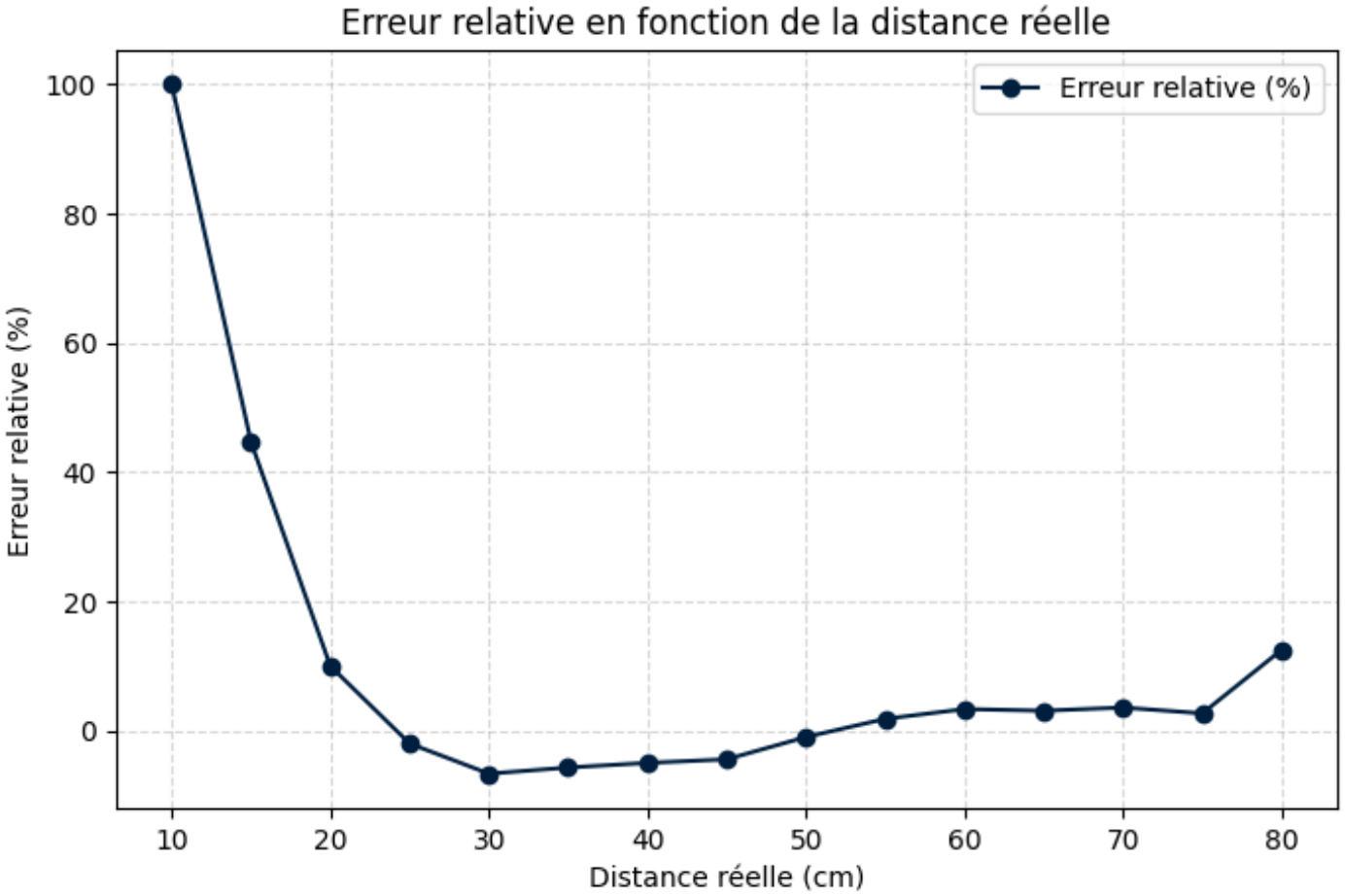


Figure 2.4

⇒ L'erreur passe de 100 % à 10 cm à –6 % à 30 cm, reste dans ±5 % de 25 à 75 cm, puis remonte à +12 % à 80 cm.

Bilan

Critère	HC-SR04	Sharp GP2Y0A02YK0F
Linéarité	Quasi colinéaire à la droite d'identité	Déviation notable en début (<25 cm) et fin (>60 cm)
Amplitude de l'erreur relative	Pic à 10 cm (~+10 %), puis toujours <±4 % dès 5	Extrême à 10 cm (+100 %), min à 30 cm (-7 %), remonte à +13 % à 80 cm
Zone de stabilité	5-110 cm : erreur contenue dans ±4 %	25-75 cm : erreur dans ±5 %

Figure 2.5

⇒ Pour une détection d'obstacle robuste, linéaire et précise sur une large plage (> 20 cm), le **HC-SR04** s'impose :

- Linéarité quasi parfaite de 20 à 110 cm
- Erreur relative maîtrisée (< 4 %)

3ème objectif:

Valider le diamètre et le matériau de l'arbre moteur.

1. Hypothèses:

- Masse de 2 kg répartie uniformément sur les 4 roues motrices.
- Réducteurs 48 : 1 modélisés avec rendement η = 0,70.
- Arbres homogènes: cylindres pleins Ø 2 mm, L = 9 mm, sans concentration de contraintes.

2. Données:

Grandeur	Valeur
Masse totale m	2 kg
Roues motrices	4
Coef. de roulement μ_{rr}	0,02
Rayon de roue r	32,5 mm
Rapport de réduction	48:1
Matériau de l'arbre	Acier S235
Ø réel de l'arbre	2 mm
Longueur L de l'arbre	9 mm
Limite élastique τ_e	235 MPa
Facteur s visé	3

Figure 3.1

3. Couple induit par la charge:

- Force de roulement par roue:

$$F_{rr} = \frac{m}{4}\mu_{rr}g = \frac{2}{4} \times 0{,}02 \times 9{,}81 \approx 0{,}098\ \text{N}$$

- Couple à la roue:

$$M_{roue} = F_{rr} \times \text{r} = 0{,}098 \times 0{,}0325 \approx 2{,}94 \times 10^{-3}\ N \cdot m$$

- Couple sur l'arbre (entrée réducteur):

$$M_{arbre} = \frac{M_{roue} \times k}{\eta} = \frac{2{,}94 \times 10^{-3} \times 1}{0{,}7 \times 48} \approx 8{,}8 \times 10^{-5}\ N \cdot m$$

4. Modélisation en torsion:

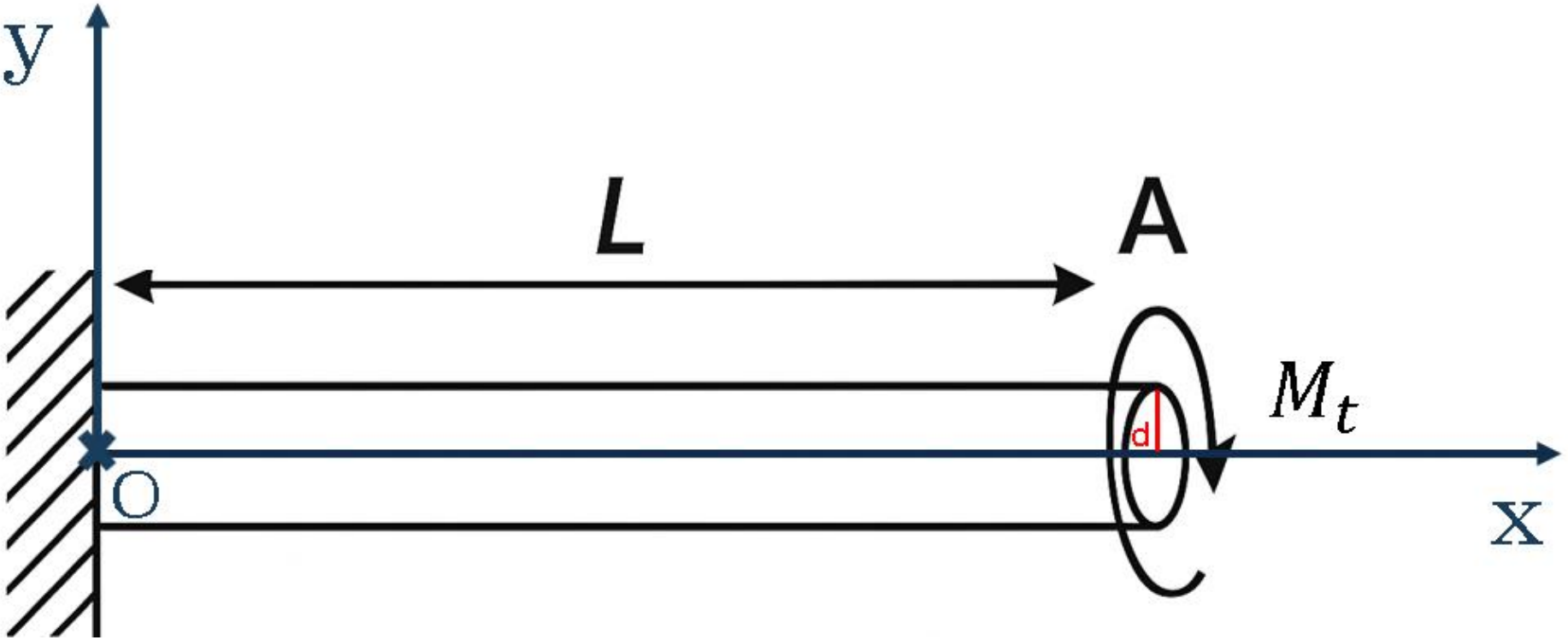


Figure 3.2

5. Formules:

- Section:

$$S = \frac{\pi d^2}{4}$$

- Moment polaire d'inertie:

$$I_0 = \frac{\pi d^4}{32}$$

- Contrainte tangentielle:

$$\tau_{max} = \frac{M_{arbre} \times \frac{d}{2}}{I_0} = \frac{M_{arbre} \times 16}{\pi \times d^3} \quad (1)$$

6. Diamètre minimal requis:

- Condition de résistance:

$$\tau_{max} \leq \frac{\tau_e}{s} \qquad (2)$$

- Isolement (on remplace par (1) dans (2)):

$$d_{min} = \sqrt[3]{\frac{16 \times M_{\text{arbre}} \times s}{\pi \times \tau_e}}$$

- Valeur numérique:

$$d_{min} = \sqrt[3]{\frac{16 \times 6{,}8 \times 10^{-5} \times 3}{\pi \times 235 \times 10^{6}}} \approx 1{,}8 \times 10^{-4}\ m = 0{,}18\ mm$$

- On a: $d_{réel}$= 2 mm ≥ 0,18 mm ⇒ Le diamètre de l'arbre moteur est valide

7. Facteur de sécurité réel:

- Formule :

$$s_{réel} = \frac{\tau_e}{\tau_{max}(d_{réel})} = \frac{\tau_e \times \pi \times d_{réel}^3}{16 \times M_{arbre}}$$

- Valeur numérique :

$$s_{réel} = \frac{235 \times 10^6 \times \pi \times 0{,}002^3}{16 \times 6{,}8 \times 10^{-5}} \approx 6000$$

- On a: $s_{réel} \gg 3$ $\Rightarrow$ Acier largement dimensionné offrant une marge de sécurité significative.

4ème objectif:

Analyse les contraintes appliquées sur les étagères et le choix du matériau adapté (AISI 304,aluminium 6061-T6 ou polycarbonate).

Remarque : toutes les images utilisées dans cette partie proviennent de mes simulations sous SolidWorks.

1. Création du système de plateaux sur Solidworks :

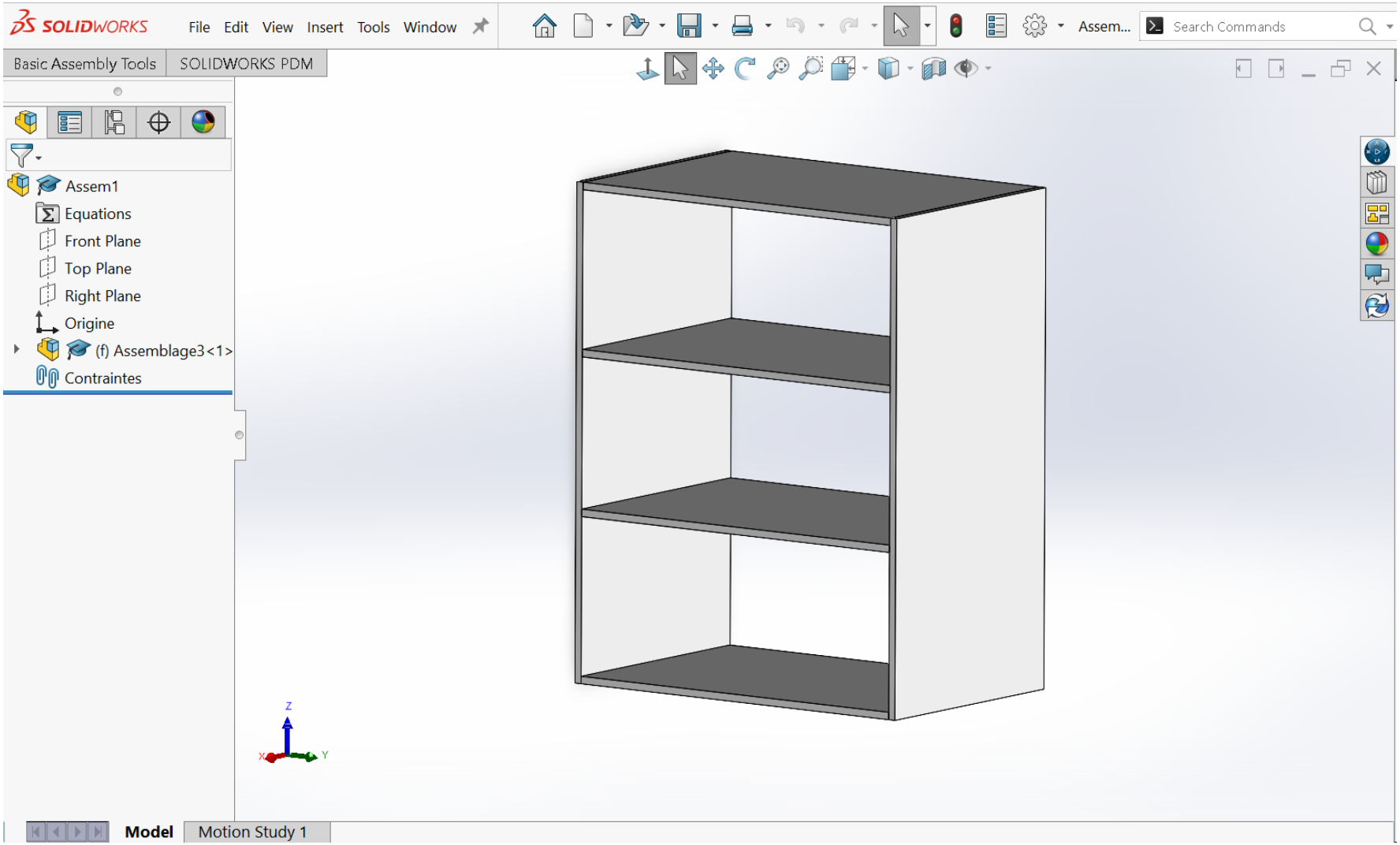


Figure 4.1

2. Mise en plan d'ensemble:

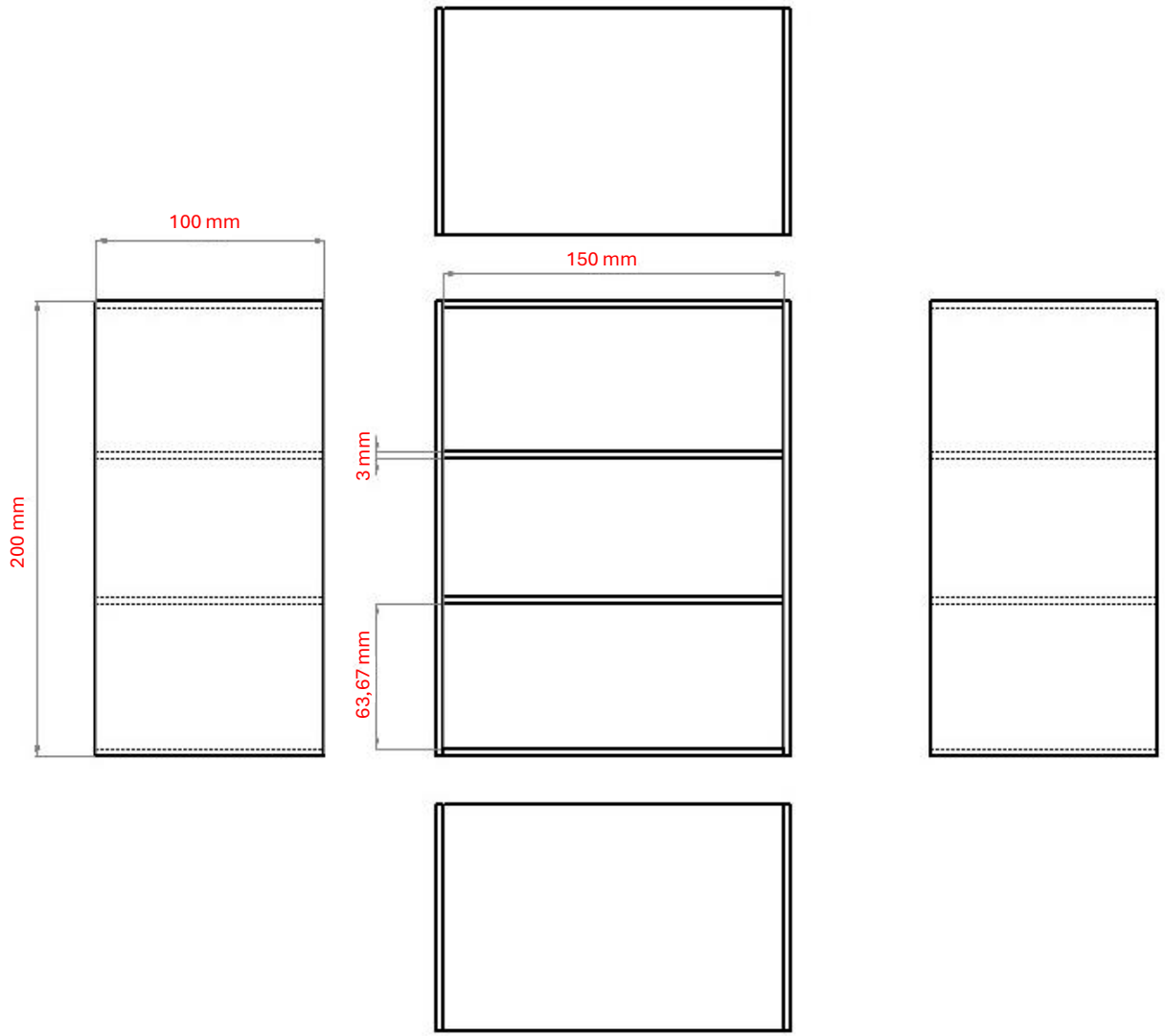


Figure 4.2

3. Cahier des charges:

Exigence	Spécification
Supporter la charge utile	0,2 kg par étagère (≃ 1,96 N)
Rigidité (flèche)	≤ 0,01 mm
Capacité mécanique et sécurité	$\sigma_{max} \leq \frac{R_e}{2}$ (CS ≥ 2)
Légèreté (masse totale)	≤ 1 kg
Coût matière	< 5 € par assemblage

Figure 4.3

AISI 304

1. Propriétés de l' AISI 304:

Material Properties

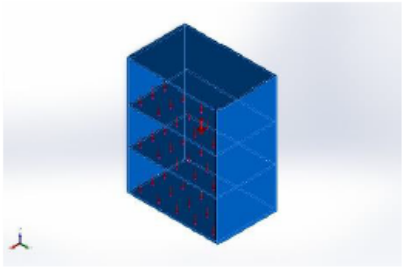
Model Reference	Properties		Components
	Name:	AISI 304	SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-1), SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-2), SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-3), SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-4), SolidBody 1(Boss.-Extru.1)(Assemblage3-1/plaque verticale 200x100x3 en mm-1), SolidBody 1(Boss.-Extru.1)(Assemblage3-1/plaque verticale 200x100x3 en mm-2)
	Model type:	Linear Elastic Isotropic	
	Default failure criterion:	Max von Mises Stress	
limite d'élasticité Re →	Yield strength:	2,06807e+08 N/m^2	
	Tensile strength:	5,17017e+08 N/m^2	
E →	Elastic modulus:	1,9e+11 N/m^2	
	Poisson's ratio:	0,29	
ρ →	Mass density:	8 000 kg/m^3	
	Shear modulus:	7,5e+10 N/m^2	
	Thermal expansion coefficient:	1,8e-05 /Kelvin	
Curve Data:N/A			

Figure 4.4

AISI 304

2. Charges:

- Chaque plateau supporte une charge de 0,2 kg (soit un poids de 1,96 N). Ses dimensions sont de 150 mm × 100 mm (surface = 0,015 m²). La pression surfacique est donc: 130,8 Pa

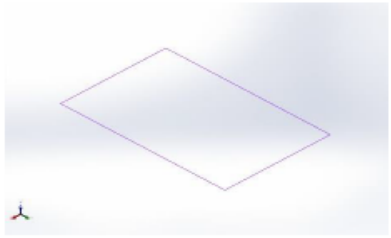
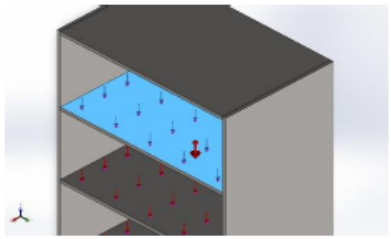
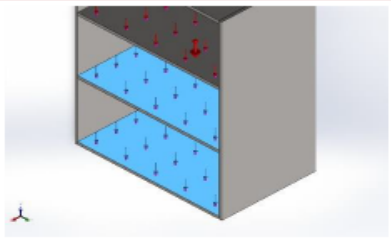
Load name	Load Image	Load Details
Gravity-1		**Reference:** Top Plane **Values:** 0 0 -9,81 **Units:** m/s^2
Pressure-1		**Entities:** 1 face(s) **Type:** Normal to selected face **Value:** 130,8 **Units:** N/m^2 **Phase Angle:** 0 **Units:** deg
Pressure-2		**Entities:** 2 face(s) **Type:** Normal to selected face **Value:** 130,8 **Units:** N/m^2 **Phase Angle:** 0 **Units:** deg

Figure 4.5

AISI 304

3. Contrainte maximale:

Name	Type	Min	Max
Stress1	VON: von Mises Stress	7,721e+01N/m^2 Node: 180939	5,046e+05N/m^2 Node: 590773

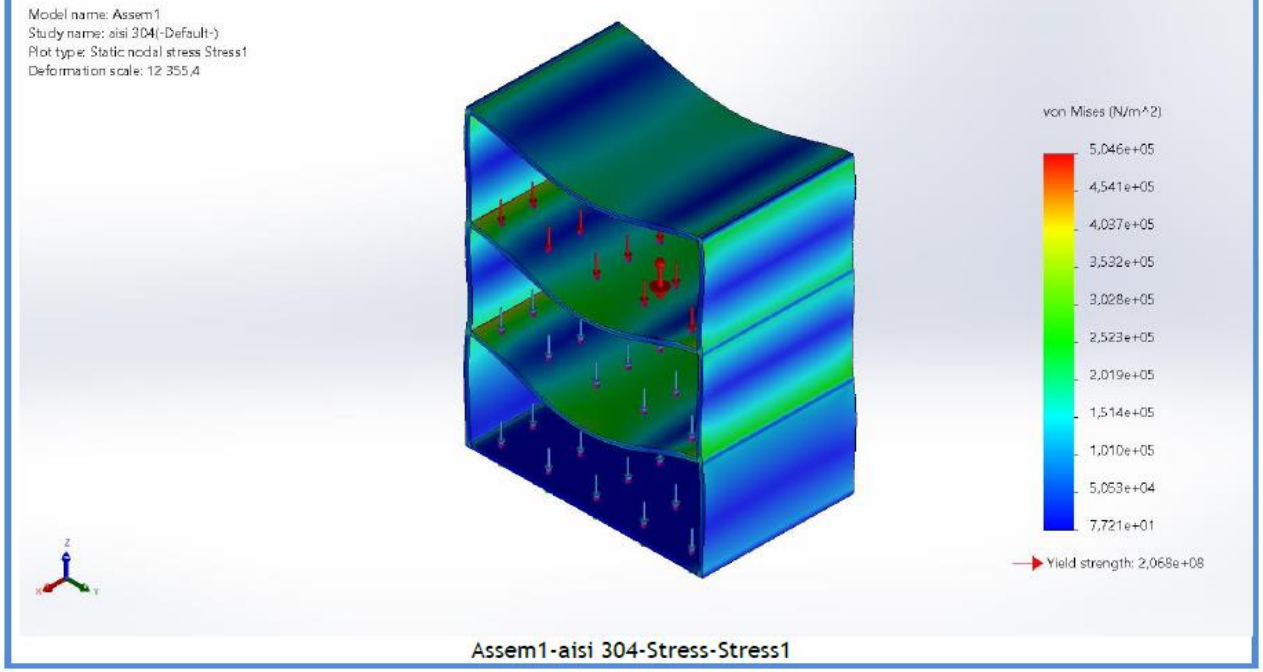


Figure 4.6

- On a $R_e = 2{,}068 \times 10^8\ N/m^2$ et $\sigma_{max} = 5{,}056 \times 10^5\ N/m^2$

$$\Rightarrow CS = \frac{R_e}{\sigma_m} = \frac{2{,}068 \times 10^8}{5{,}056 \times 10^5} \approx 410$$

AISI 304

4. Flèche maximale :

Name	Type	Min	Max
Displacement1	URES: Resultant Displacement	0,000e+00mm Node: 76542	1,619e-03mm Node: 587068

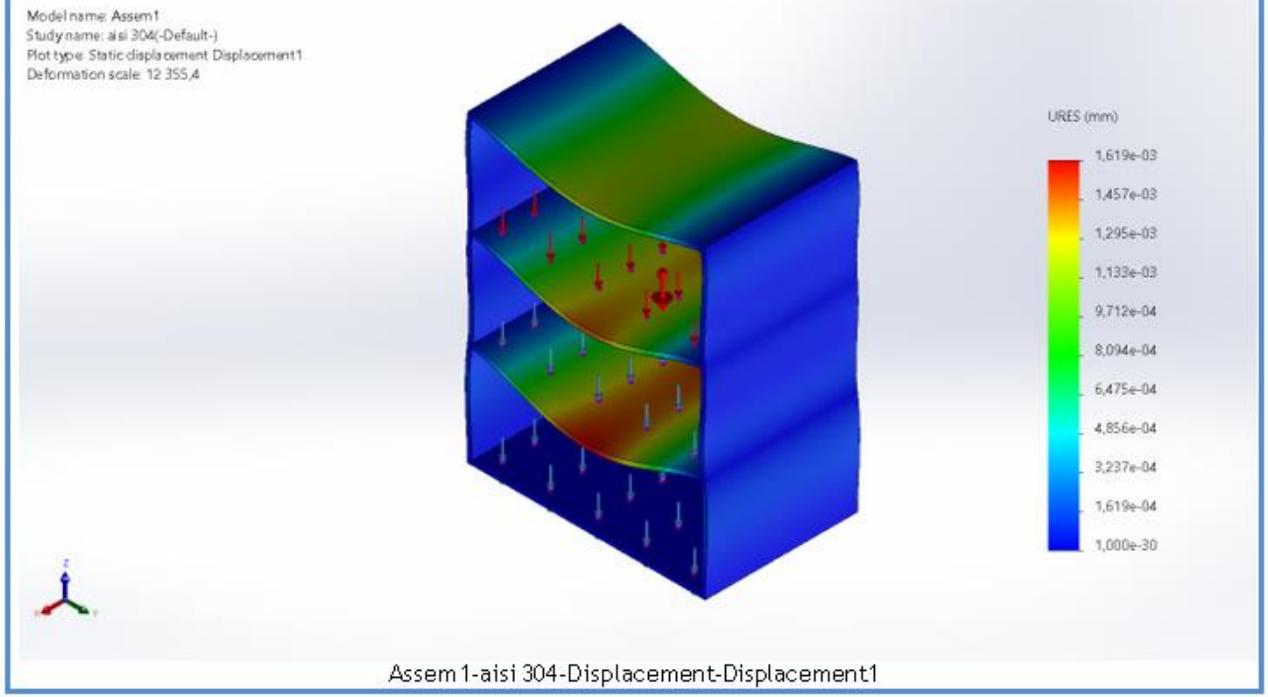


Figure 4.7

- La flèche maximale est : $1{,}619 \times 10^{-3}$ mm

Aluminium 6061-T6

1. Propriétés de l' Aluminium 6061-T6 :

Material Properties

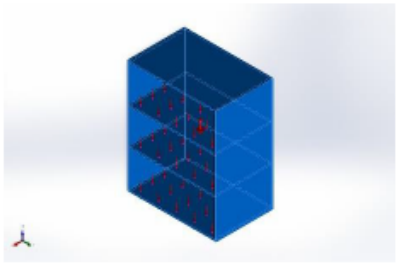
Model Reference	Properties		Components
	Name:	6061-T6 (SS)	SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-1), SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-2), SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-3), SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-4), SolidBody 1(Boss.-Extru.1)(Assemblage3-1/plaque verticale 200x100x3 en mm-1), SolidBody 1(Boss.-Extru.1)(Assemblage3-1/plaque verticale 200x100x3 en mm-2)
	Model type:	Linear Elastic Isotropic	
	Default failure criterion:	Max von Mises Stress	
limite d'élasticité Re →	Yield strength:	2,75e+08 N/m^2	
	Tensile strength:	3,1e+08 N/m^2	
E →	Elastic modulus:	6,9e+10 N/m^2	
	Poisson's ratio:	0,33	
ρ →	Mass density:	2 700 kg/m^3	
	Shear modulus:	2,6e+10 N/m^2	
	Thermal expansion coefficient:	2,4e-05 /Kelvin	
Curve Data:N/A			

Figure 4.8

Aluminium 6061-T6

2. Contrainte maximale:

Name	Type	Min	Max
Stress1	VON: von Mises Stress	6,630e+01N/m^2 Node: 181028	2,833e+05N/m^2 Node: 590773

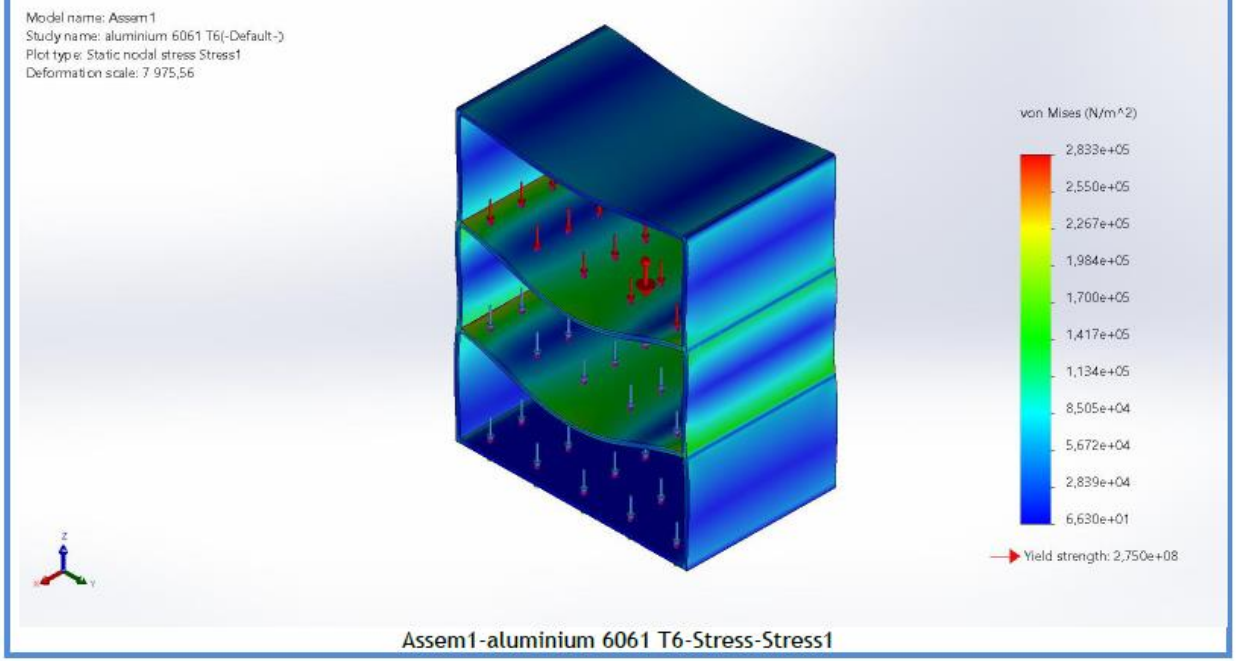


Figure 4.9

- On a $R_e = 2{,}750 \times 10^8\ N/m^2$ et $\sigma_{max} = 2{,}833 \times 10^5\ N/m^2$

$$\Rightarrow CS = \frac{R_e}{\sigma_m} = \frac{2{,}750 \times 10^8}{2{,}833 \times 10^5} \approx 970$$

Aluminium 6061-T6

3. Flèche maximale :

Name	Type	Min	Max
Displacement1	URES: Resultant Displacement	0,000e+00mm Node: 76542	2,508e-03mm Node: 587068

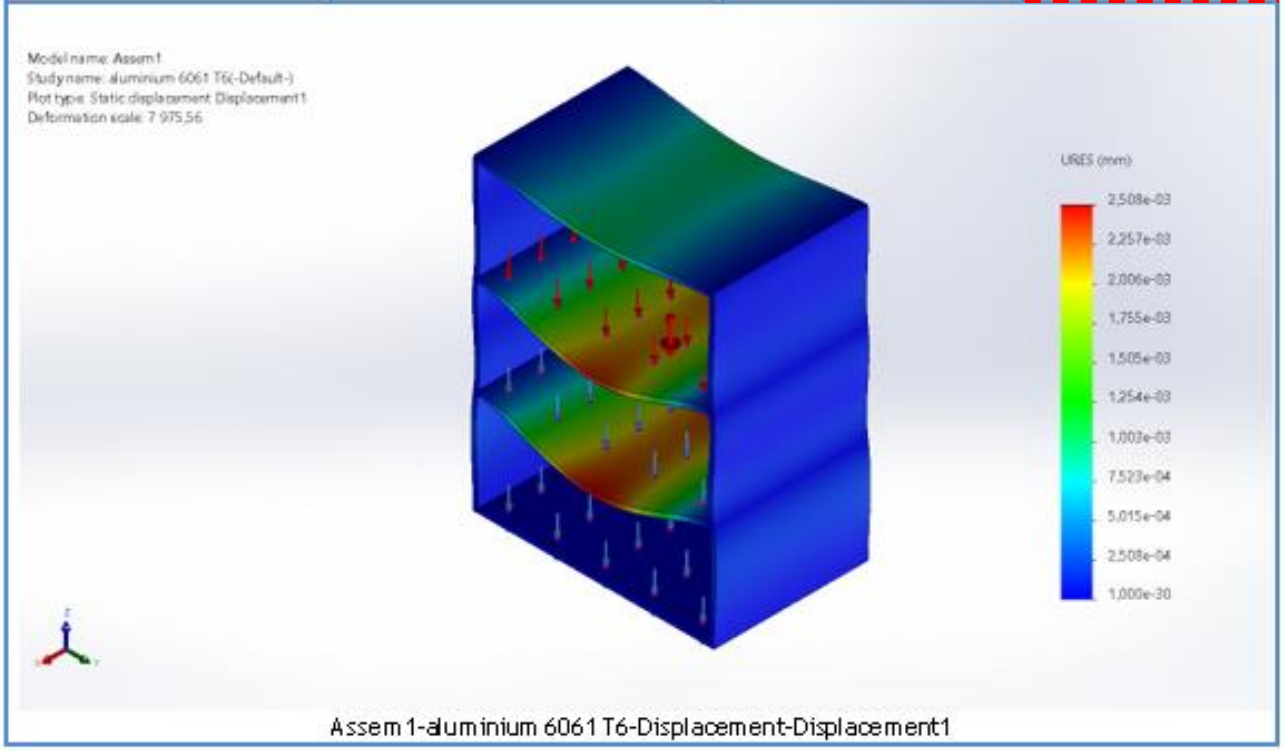


Figure 4.10

- La flèche maximale est : $2{,}508 \times 10^{-3}$ mm

Polycarbonate

1. Propriétés du Polycarbonate :

Material Properties

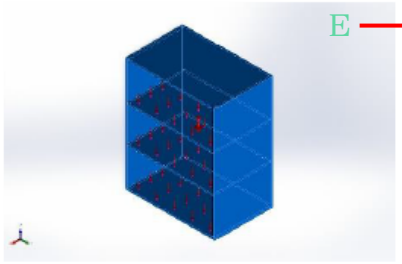
Model Reference	Properties		Components
limite d'élasticité Re →	Name:	Polycarbonate	SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-1), SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-2), SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-3), SolidBody 1(Boss.-Extru.2)(Assemblage3-1/Pièce de base 150x100x3 mm-4), SolidBody 1(Boss.-Extru.1)(Assemblage3-1/plaque verticale 200x100x3 en mm-1), SolidBody 1(Boss.-Extru.1)(Assemblage3-1/plaque verticale 200x100x3 en mm-2)
	Model type:	Linear Elastic Isotropic	
	Default failure criterion:	Max von Mises Stress	
	Yield strength:	7e+07 N/m^2	
	Tensile strength:	7e+07 N/m^2	
	Compressive strength:	1e+08 N/m^2	
 E →	Elastic modulus:	2,2e+09 N/m^2	
	Poisson's ratio:	0,37	
ρ →	Mass density:	1 200 kg/m^3	
	Shear modulus:	8,02e+08 N/m^2	
Curve Data:N/A			

Figure 4.11

Polycarbonate

2. Contrainte maximale:

Name	Type	Min	Max
Stress1	VON: von Mises Stress	5,391e+01N/m^2 Node: 78726	2,164e+05N/m^2 Node: 590773

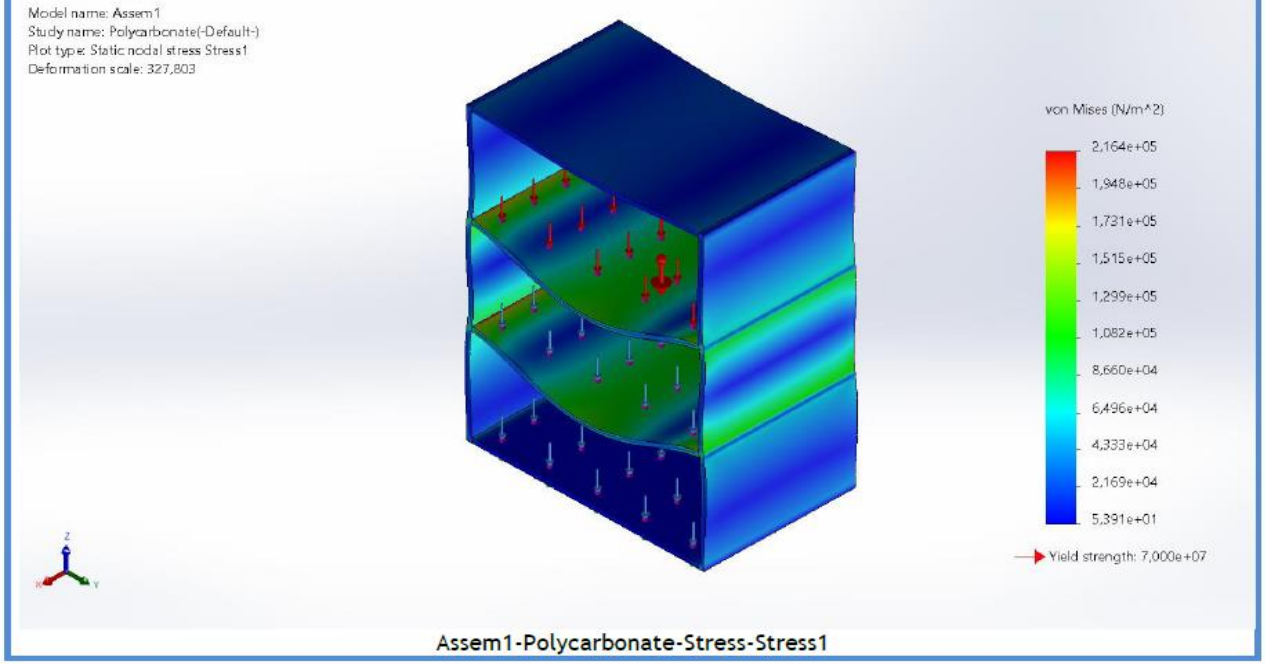


Figure 4.12

- On a $R_e = 7 \times 10^7\ N/m^2$ et $\sigma_{max} = 2{,}164 \times 10^5\ N/m^2$

$$\Rightarrow CS = \frac{R_e}{\sigma_m} = \frac{7 \times 10^7}{2{,}164 \times 10^5} \approx 323$$

Polycarbonate

3. Flèche maximale :

Name	Type	Min	Max
Displacement1	URES: Resultant Displacement	0,000e+00mm Node: 76542	6,101e-02mm Node: 587068

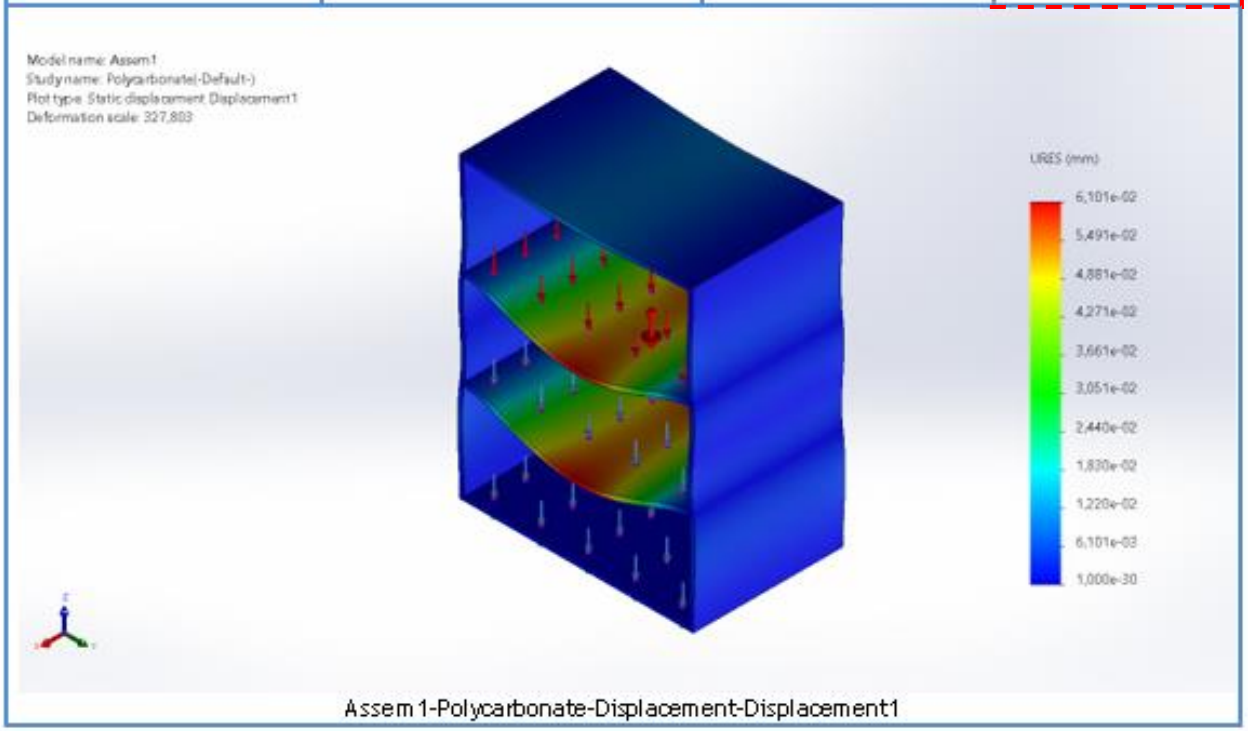


Figure 4.13

- La flèche maximale est : $6{,}101 \times 10^{-2}$ mm

Bilan

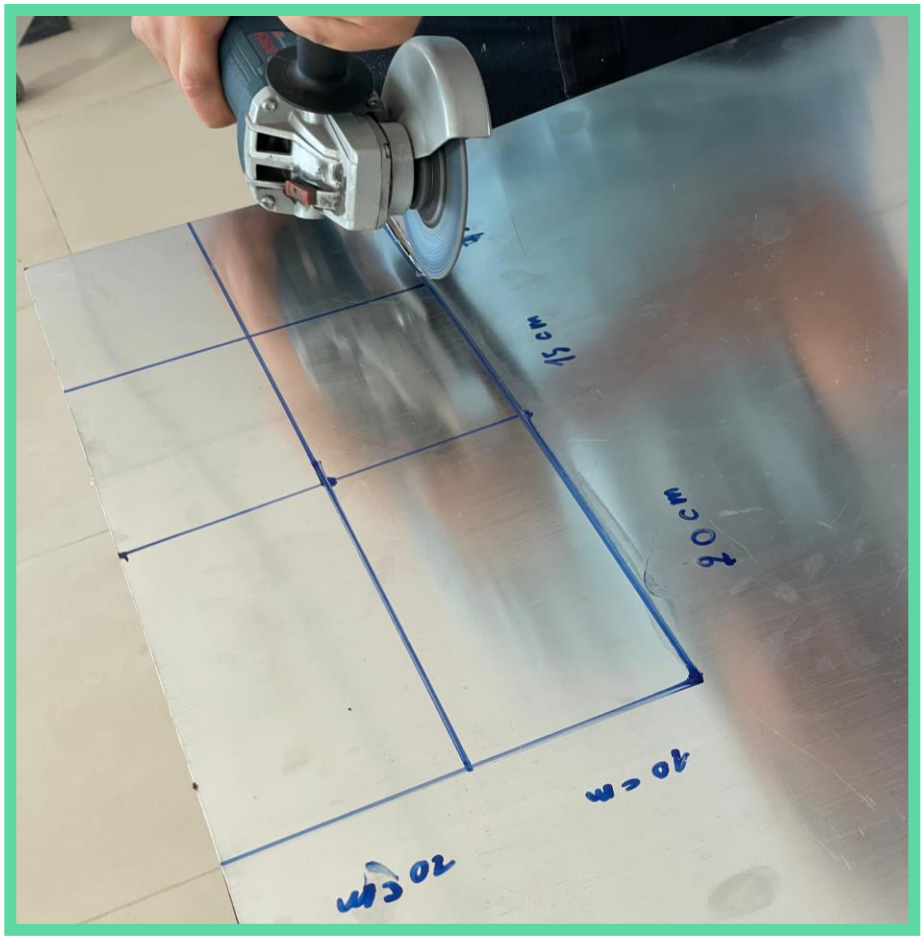
Matériau	Masse	Flèche max	CS: s	Prix	C.d.C
AISI 304	2,40 Kg ❌	$1,619 \times 10^{-3}$ mm ✅	410 ✅	≈2,5 €/Kg ✅	❌
Aluminium 6061-T6	0,81 Kg ✅	$2,508 \times 10^{-3}$ mm ✅	970 ✅	≈1,8 €/Kg ✅	✅
Polycarbonate	0,36 Kg ✅	$6,101 \times 10^{-2}$ mm ❌	323 ✅	≈20 €/Kg ❌	❌

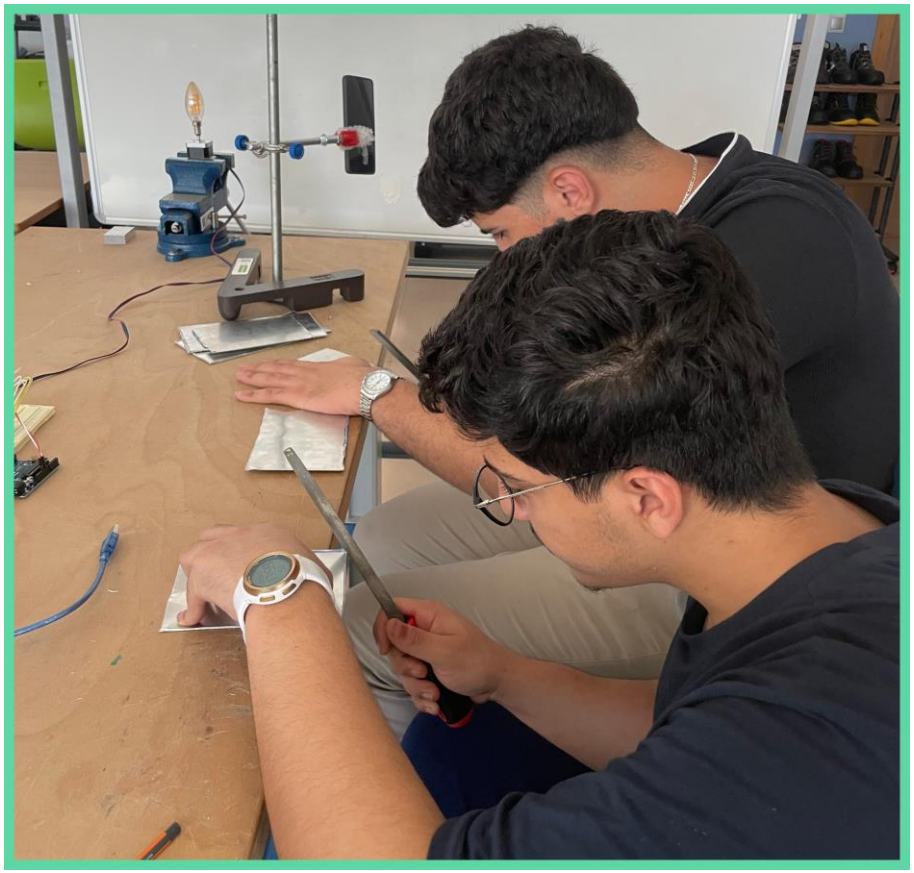
Figure 4.14

⇒ Nous optons pour l'aluminium 6061-T6 en raison de ses excellentes caractéristiques de légèreté, de rigidité et de coût.

5ème objectif:

Construire un prototype pour valider la mise en œuvre du système.


15cm
20cm
40cm
20cm




Young Leaders
19 81

Conclusion

Merci pour votre attention!

BFS et fusion de segments

Code Python

```python
from collections import deque
import matplotlib.pyplot as plt
import numpy as np
import time

# Chargement + inversion + miroir

def charger_carte_sym(csv_path):
    mat = []
    with open(csv_path, "r") as f:
        for ligne in f:
            mat.append([1 - int(x) for x in ligne.strip().split(",")])
    return mat[::-1]

DELTA = [(-1,0),(1,0),(0,-1),(0,1),(-1,-1),(-1,1),(1,-1),(1,1)]

def reconstituer(parent, lf, cf):
    ch, cur = [], (lf, cf)
    while cur:
        ch.append(cur)
        cur = parent[cur[0]][cur[1]]
    return ch[::-1]


def bfs(mat, dep, arr):
    (ld, cd), (la, ca) = dep, arr
    n, m = len(mat), len(mat[0])
    if mat[ld][cd] == 0 or mat[la][ca] == 0:
        return []
    dist = [[-1]*m for _ in range(n)]
    parent = [[None]*m for _ in range(n)]
    q = deque([(ld, cd)])
    dist[ld][cd] = 0
    while q:
        l, c = q.popleft()
        if (l, c) == (la, ca):
            return reconstituer(parent, l, c)
        for dl, dc in DELTA:
            ln, cn = l + dl, c + dc
            if 0 <= ln < n and 0 <= cn < m and mat[ln][cn] == 1 and dist[ln][cn] == -1:
                dist[ln][cn] = dist[l][c] + 1
                parent[ln][cn] = (l, c)
                q.append((ln, cn))
    return []


def fusion(chemin):
    if len(chemin) <= 2:
        return chemin
    optim = [chemin[0]]
    dir_prev = (chemin[1][0]-chemin[0][0], chemin[1][1]-chemin[0][1])
    for i in range(1, len(chemin)-1):
        dir_curr = (chemin[i+1][0]-chemin[i][0], chemin[i+1][1]-chemin[i][1])
        if dir_curr != dir_prev:
            optim.append(chemin[i])
            dir_prev = dir_curr
    optim.append(chemin[-1])
    return optim


def montrer(mat, ch_fuse):
    plt.imshow(np.array(mat), cmap="gray")
    xs = [c for (l, c) in ch_fuse]
    ys = [l for (l, _) in ch_fuse]
    plt.plot(xs, ys, 'r', lw=2)
    plt.gca().invert_yaxis()
    plt.show()


def main():
    mat = charger_carte_sym(r"C:\Users\hp\Desktop\avant.csv")
    H = len(mat)
    dep_orig = (0, 4)
    arr_orig = (23, 12)
    dep = (H - 1 - dep_orig[0], dep_orig[1])
    arr = (H - 1 - arr_orig[0], arr_orig[1])
    t0 = time.perf_counter()
    chemin_brut = bfs(mat, dep, arr)
    duree = (time.perf_counter() - t0) * 1000
    print(f"Dur e BFS : {duree:.2f} ms")
    if not chemin_brut:
        print("Aucun chemin trouv .")
        return
    chemin_brut_xy = [(c * 10, l * 10) for (l, c) in chemin_brut]
    print(chemin_brut_xy)
    chemin_fusion = fusion(chemin_brut)
    print(chemin_fusion)
    montrer(mat, chemin_fusion)

if __name__ == "__main__":
    main()
```

Algorithme de fusion des segments

Entrée: *chemin* : liste ordonnée de points $P_0, \ldots, P_k$
Sortie: Liste compactée conservant uniquement les changements de direction

if $chemin \leq 2$ **then**
  return *chemin* ; `# rien à fusionner`
$fused \leftarrow [P_0]$;
$dir\_prev \leftarrow P_1 - P_0$; `# vecteur directeur`
for $i \leftarrow 1$ **to** $k - 1$ **do**
  $dir\_curr \leftarrow P_{i+1} - P_i$;
  if $dir\_curr \neq dir\_prev$ **then**
    $fused.ajouter(P_i)$;
    $dir\_prev \leftarrow dir\_curr$;
  end
end
$fused.ajouter(P_k)$; `# toujours garder l'arrivée`
return *fused*;

La fonction atan2

Définition de la fonction atan2

Soit $(x, y) \in \mathbb{R}^2 \setminus \{(0,0)\}$. La fonction $\text{atan2}(y, x)$ renvoie l'angle $\alpha \in (-\pi, \pi]$ tel que

$$(\cos\alpha, \sin\alpha) = \frac{1}{\sqrt{x^2+y^2}}(x, y).$$

On peut l'écrire de façon équivalente :

$$\text{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{si } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{si } x < 0 \text{ et } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{si } x < 0 \text{ et } y < 0, \\ +\frac{\pi}{2} & \text{si } x = 0 \text{ et } y > 0, \\ -\frac{\pi}{2} & \text{si } x = 0 \text{ et } y < 0. \end{cases}$$

Par convention, $\text{atan2}(0,0)$ est souvent défini comme 0.

Mesure de distance à un obstacle

```ino
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

const int trigPin = 9;
const int echoPin = 10;

void setup() {
    Serial.begin(9600);
    lcd.init();
    lcd.backlight();
    lcd.print("Distance:");
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
}

void loop() {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    long duration = pulseIn(echoPin, HIGH);
    float distance = duration * 0.034 / 2;

    lcd.setCursor(0, 1);
    lcd.print("                ");
    lcd.setCursor(0, 1);
    lcd.print(distance);
    lcd.print(" cm");

    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");

    delay(500);
}
```

HC-SR04 et LCD 16x2

Mesure de distance à un obstacle

```ino
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);

const int sharpPin = A0;
int rawValue;
float distance;

void setup() {
    lcd.begin(20, 4);
    lcd.backlight();
    lcd.setCursor(0, 0);
    lcd.print("Initializing...");
    delay(2000);
    lcd.clear();
    Serial.begin(9600);
}

void loop() {
    rawValue = analogRead(sharpPin);
    distance = 60.495 * pow(rawValue * (5.0 / 1023.0), -1.190);

    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");

    lcd.setCursor(0, 0);
    lcd.print("Distance: ");
    lcd.print(distance, 2);
    lcd.print(" cm   ");

    delay(500);
}
```

Capteur Sharp et LCD 20x4

Asservissement en vitesse

```ino
#include <Arduino.h>

/* -------- PARAM TRES -------- */
const int   PPR     = 20;
const float r       = 0.0325f;    // rayon [m]
const float dt      = 0.05f;      // p riode boucle (50 ms)
const float Kp      = 50.0f;
const float Ki      = 2.192f;

/* -------- BROCHAGE -------- */
#define ENC_FL_PIN 2
#define ENC_BL_PIN 3
#define ENC_FR_PIN 4
#define ENC_BR_PIN 5
#define DIR_L1_PIN 6
#define DIR_L2_PIN 7
#define PWM_L_PIN 9
#define DIR_R1_PIN 10
#define DIR_R2_PIN 11
#define PWM_R_PIN 12

/* -------- VARIABLES -------- */
volatile long pulsesFL = 0, pulsesBL = 0;
volatile long pulsesFR = 0, pulsesBR = 0;
long lastFL = 0, lastBL = 0;
long lastFR = 0, lastBR = 0;
float sumErrFL = 0, sumErrBL = 0;
float sumErrFR = 0, sumErrBR = 0;
int8_t dirFL = 1, dirBL = 1;
int8_t dirFR = 1, dirBR = 1;

/* -------- ISR -------- */
void ISR_FL() { ++pulsesFL; }
void ISR_BL() { ++pulsesBL; }
void ISR_FR() { ++pulsesFR; }
void ISR_BR() { ++pulsesBR; }

/* -------- SETUP -------- */
```

```ino
void setup() {
  pinMode(ENC_FL_PIN, INPUT_PULLUP);
  pinMode(ENC_BL_PIN, INPUT_PULLUP);
  pinMode(ENC_FR_PIN, INPUT_PULLUP);
  pinMode(ENC_BR_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ENC_FL_PIN), ISR_FL, RISING);
  attachInterrupt(digitalPinToInterrupt(ENC_BL_PIN), ISR_BL, RISING);
  attachInterrupt(digitalPinToInterrupt(ENC_FR_PIN), ISR_FR, RISING);
  attachInterrupt(digitalPinToInterrupt(ENC_BR_PIN), ISR_BR, RISING);

  pinMode(DIR_L1_PIN, OUTPUT);
  pinMode(DIR_L2_PIN, OUTPUT);
  pinMode(PWM_L_PIN, OUTPUT);
  pinMode(DIR_R1_PIN, OUTPUT);
  pinMode(DIR_R2_PIN, OUTPUT);
  pinMode(PWM_R_PIN, OUTPUT);
}

/* -------- PI  D UNE  ROUE -------- */
void updateWheel(float    vRef,       // consigne m/s
                 long     dP_signed,  //    impulsions sign
                 float &  sumErr,     // int grale erreur
                 int8_t & dirCmd,     // sens appliqu
                 uint8_t  DIR1,
                 uint8_t  DIR2,
                 uint8_t  PWMpin) {
  float dS    = (2 * PI * r) * dP_signed / PPR;
  float vMeas = dS / dt;
  float e     = vRef - vMeas;
  sumErr    += e * dt;
  float u     = constrain(Kp * e + Ki * sumErr, -1.0f, 1.0f);
  dirCmd      = (u >= 0) ? 1 : -1;
  int pwm     = (int)(fabs(u) * 255);
  digitalWrite(DIR1, dirCmd > 0);
  digitalWrite(DIR2, dirCmd < 0);
  analogWrite(PWMpin, pwm);
}
```

Asservissement en vitesse

```
77  /* -------- BOUCLE PRINCIPALE -------- */
78  void loop() {
79    static unsigned long tLast = micros();
80    const unsigned long dt_us = (unsigned long)(dt * 1e6);
81    if ((long)(micros() - tLast) < (long)dt_us) return;
82    tLast += dt_us;
83
84    noInterrupts();
85    long pFL = pulsesFL, pBL = pulsesBL;
86    long pFR = pulsesFR, pBR = pulsesBR;
87    interrupts();
88
89    long dFL = pFL - lastFL; lastFL = pFL;
90    long dBL = pBL - lastBL; lastBL = pBL;
91    long dFR = pFR - lastFR; lastFR = pFR;
92    long dBR = pBR - lastBR; lastBR = pBR;
93
94    long sFL = dFL * dirFL;
95    long sBL = dBL * dirBL;
96    long sFR = dFR * dirFR;
97    long sBR = dBR * dirBR;
98
99    float vRefL = 0.20f;
100   float vRefR = 0.20f;
101
102   updateWheel(vRefL, sFL, sumErrFL, dirFL, DIR_L1_PIN, DIR_L2_PIN,
        PWM_L_PIN);
103   updateWheel(vRefL, sBL, sumErrBL, dirBL, DIR_L1_PIN, DIR_L2_PIN,
        PWM_L_PIN);
104   updateWheel(vRefR, sFR, sumErrFR, dirFR, DIR_R1_PIN, DIR_R2_PIN,
        PWM_R_PIN);
105   updateWheel(vRefR, sBR, sumErrBR, dirBR, DIR_R1_PIN, DIR_R2_PIN,
        PWM_R_PIN);
106 }
```

Cinématique différentielle et odométrie

- Formules de la cinématique différentielle:
  https://en.wikipedia.org/wiki/Differential_wheeled_robot

- Formules de l'odométrie:
  https://automaticaddison.com/calculating-wheel-odometry-for-a-differential-drive-robot/