

# Robot ShapIO

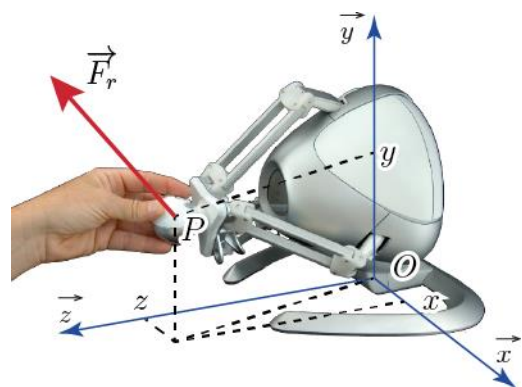
## Livret de présentation pédagogique Applications en CPGE

### Système réel et Technologie connectée

Le robot ShapIO est une Interface Homme Machine pour l'immersion en réalité virtuelle sur ordinateur ou pour la téléopération. Il est le lien entre **le réel et le numérique**. Son architecture mécanique est basée sur une **structure « Delta »** à trois mobilités, équipée de trois codeurs et trois moteurs pilotés en courant. Equipé d'une carte Raspberry Pi et de Python, il devient un Système Haptique pour l'Internet des Objets (ShapIO) - intelligent et connecté.

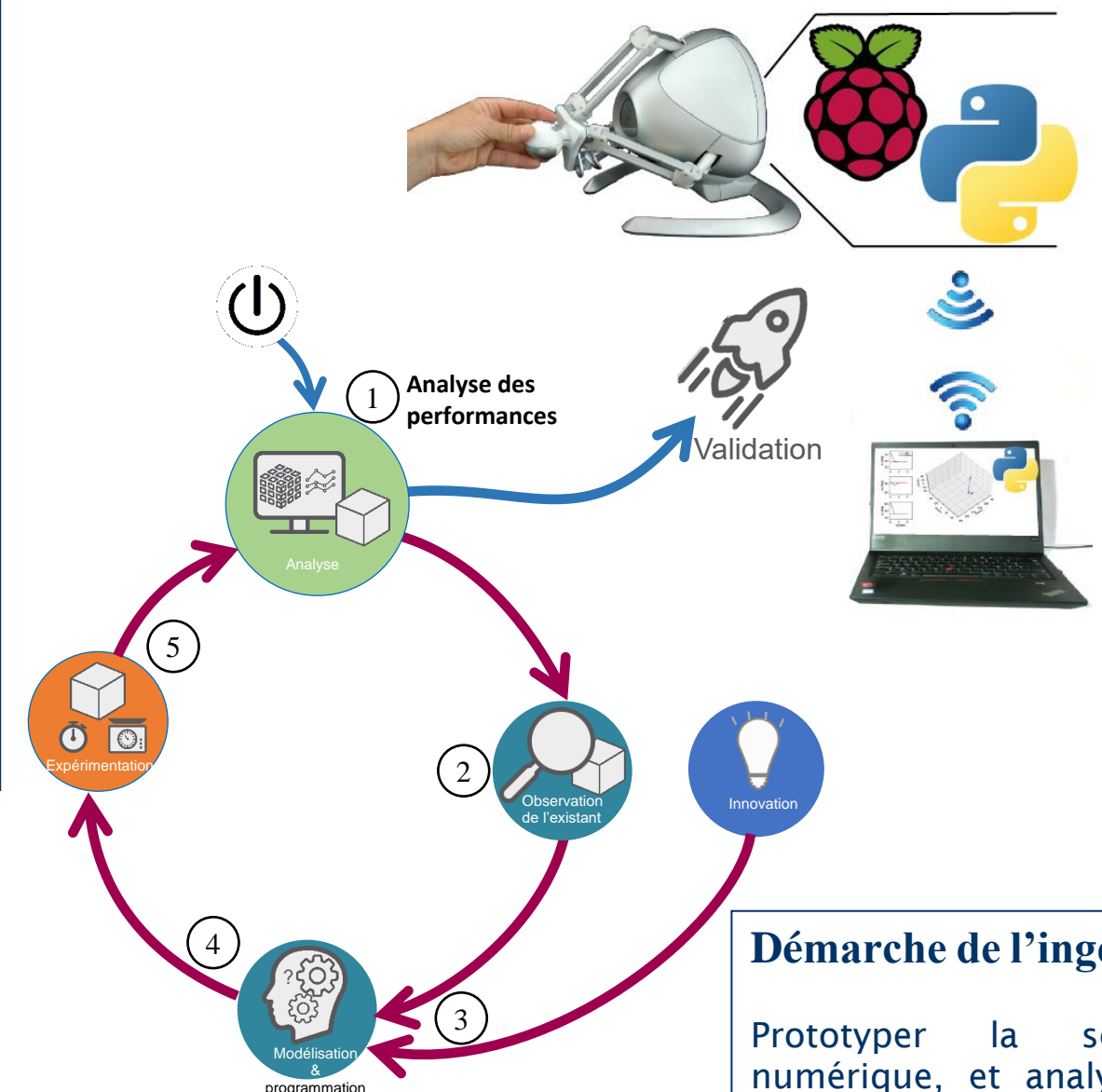
Il est un support d'**activités pratiques** pour l'acquisition des **fondamentaux** de l'informatique au service des Sciences de l'ingénieur : algorithmie & programmation, codage de l'information, conception informationnelle des produits (instrumentation et pilotage) et transmission de l'information (réseaux IP & Bluetooth).

Mais il reste aussi totalement ouvert, connecté et interfaçable pour un usage dans le cadre de **projets** (pilotage d'un drone, téléopération par exemple).



### Haptique ?

Pour réaliser une tâche virtuelle ou distante, l'opérateur indique son intention en déplaçant la poignée. Il ressent aussi un effort restitué : ce retour sensoriel est qualifié d'**haptique**.



### Démarche de l'ingénieur

Prototyper la solution numérique, et analyser le gain de performance



## COMPETENCES ET CONNAISSANCES VISEES

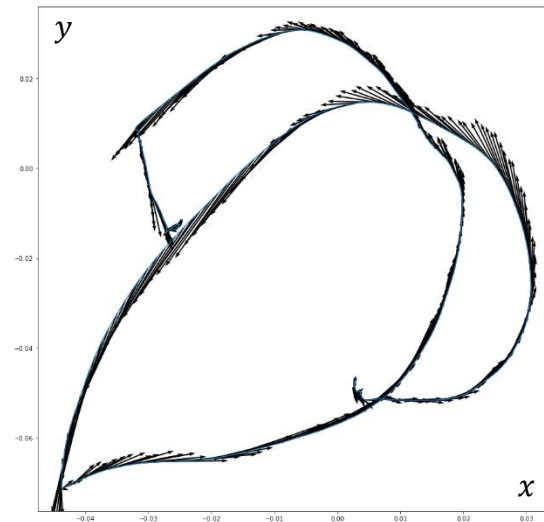
### THEME :

- Modéliser la cinématique
- Analyser un algorithme
- Mettre en œuvre un système en suivant un protocole
- Caractériser des signaux échantillonnés
- Interpréter et vérifier la cohérence des résultats obtenus expérimentalement ou numériquement
- Effectuer des traitements à partir de données
- Implémenter et exécuter un programme (dérivation numérique)
- Intégration et dérivation numérique
- Produire des documents techniques adaptés à l'objectif de la communication

## Exemples d'activités pédagogiques proposées

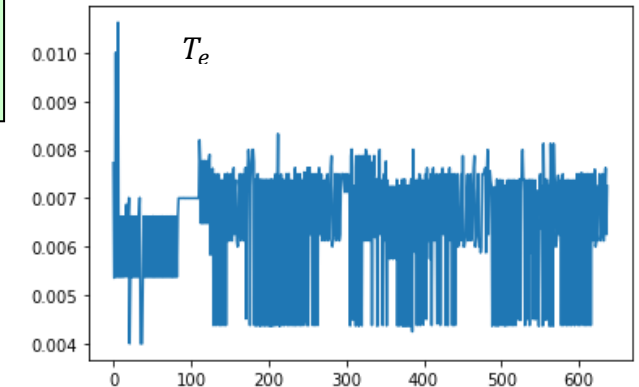
### MESURER LA POSITION ET ESTIMER LA VITESSE

1 – Expérimenter  
Utiliser les fonctions logicielles d'une bibliothèque de pilotage  
Représenter la trajectoire



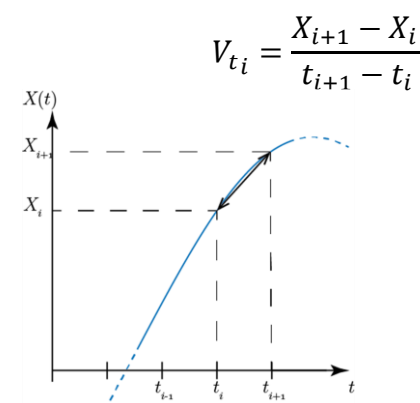
Trajectoire & champ des vitesses

2- Analyser le comportement  
Quantifier la période d'échantillonnage et sa régularité



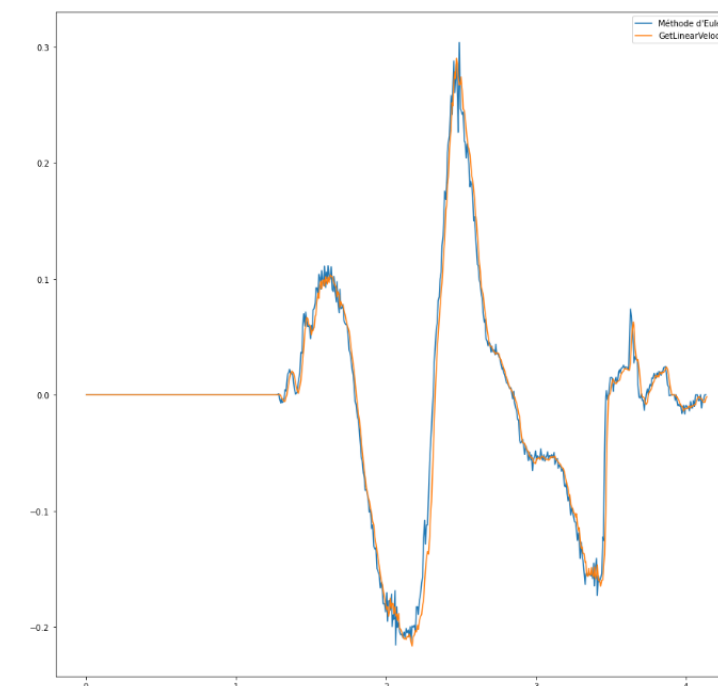
3 – Résoudre  
Mettre en œuvre une dérivation numérique par la méthode d'Euler

Approximation d'Euler avant



```
DT=[]
DX=[]
Vx=[]
for i in range(len(X)-1):
    DT.append(T[i+1]-T[i])
    DX.append(X[i+1]-X[i])
    Vx.append(DX[i]/DT[i])
```

4 – Analyser les performances  
Comparer les estimations par différentes méthodes, pour différents paramètres





## COMPETENCES ET CONNAISSANCES VISEES

### THEME :

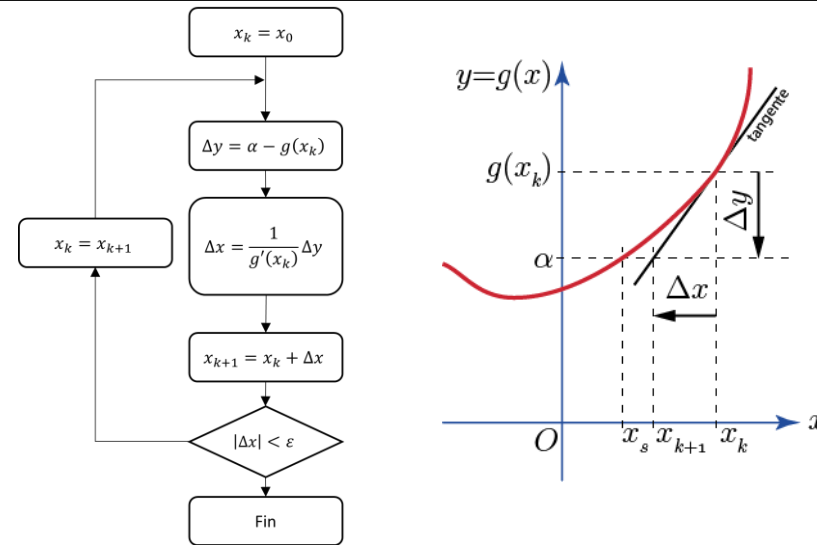
- Modéliser la cinématique
- Analyser un algorithme
- Mettre en œuvre un système en suivant un protocole
- Choisir une démarche de résolution d'un problème d'ingénierie numérique
- Résoudre numériquement une équation
- Implémenter et exécuter un programme (méthode de Newton)
- Résoudre numériquement une équation
- Implémenter et exécuter un programme sur une cible
- Valider le fonctionnement du prototype

## Exemples d'activités pédagogiques proposées

### ESTIMER LA POSITION DE LA POIGNEE

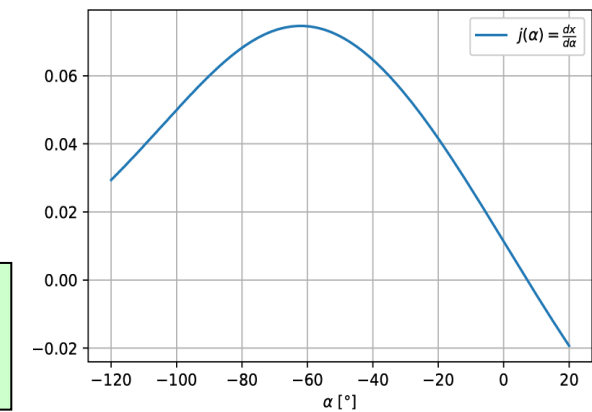
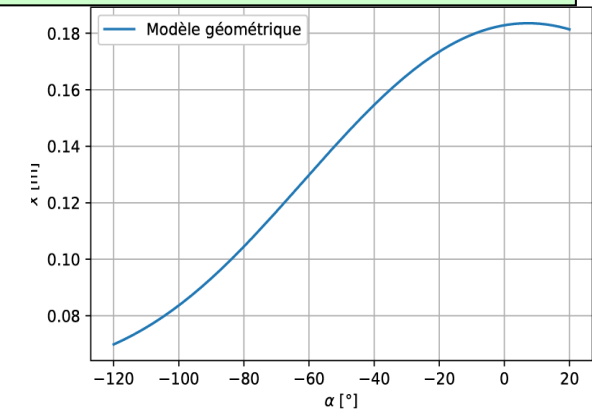
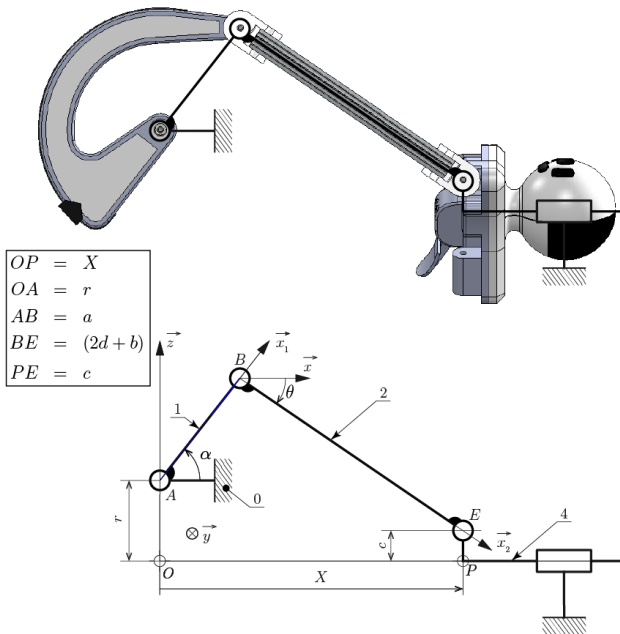
Méthode de Newton & usage en robotique (résolution d'équation)

#### 1- Rappel méthodologique



#### 2- Cas du problème plan :

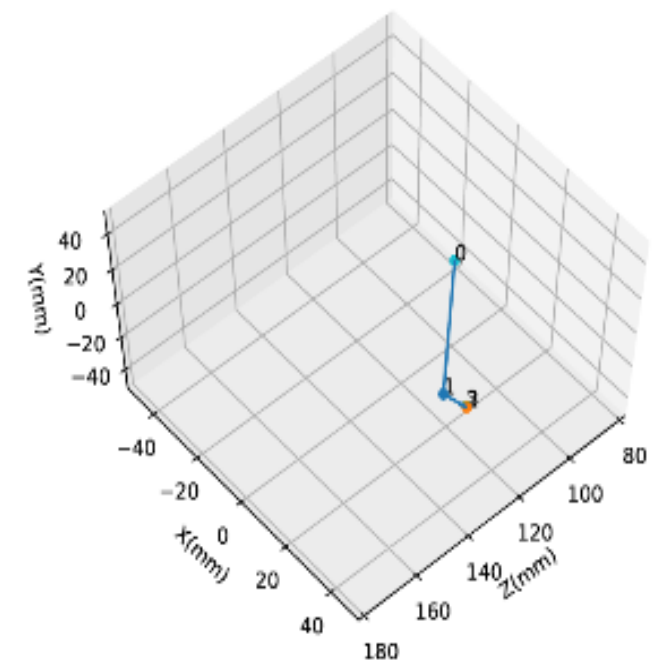
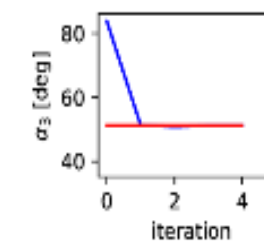
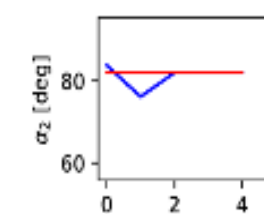
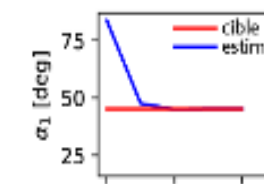
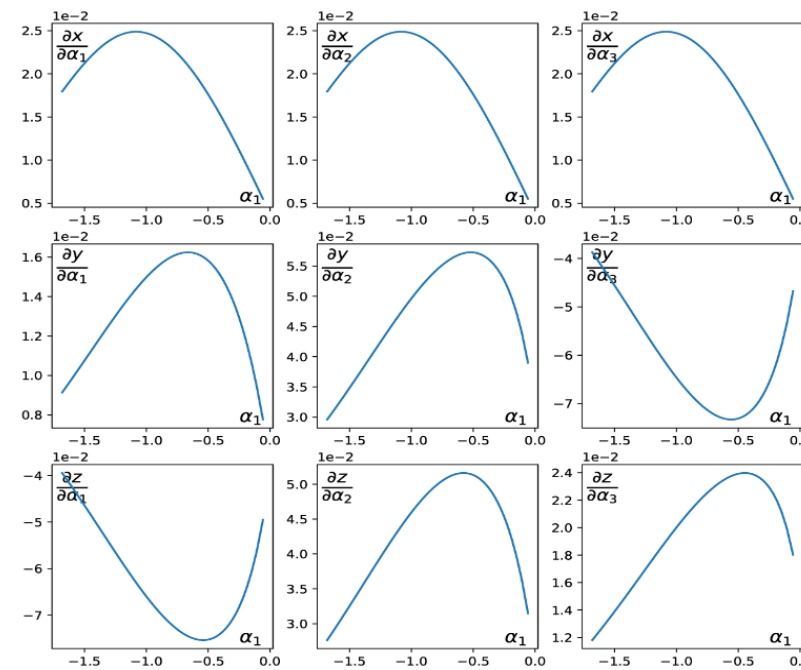
- mise en place de la méthode : recherche de solution (s'appuie sur le gain cinématique)
- performances selon la configuration



#### 3- Cas du problème dans l'espace : recherche multivariable

- pas de solution mathématique
- analyse de la convergence

$$dX = [J]d\alpha$$





## COMPETENCES ET CONNAISSANCES VISEES

### THEME :

- Modéliser la cinématique
- Analyser un algorithme
- Mettre en œuvre un système en suivant un protocole
- Choisir une démarche de résolution d'un problème d'ingénierie numérique
- Résoudre numériquement un système d'équations linéaires
- Implémenter et exécuter un programme (Résolution d'un système linéaire)
- Modifier la commande pour faire évoluer le comportement du système

## Exemples d'activités pédagogiques proposées

**RESTITUER UN EFFORT**  
Détermination de la loi de transmission en effort (inversion de matrice)

### 1- Mise en place du problème

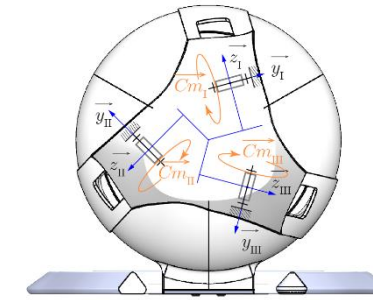
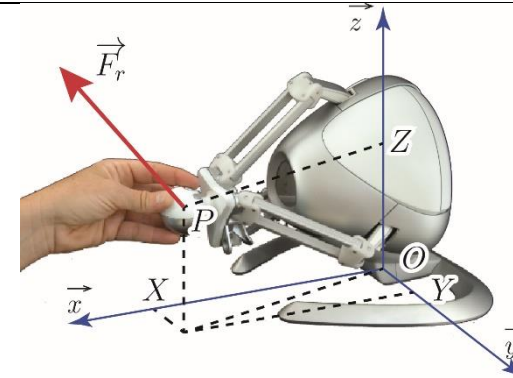
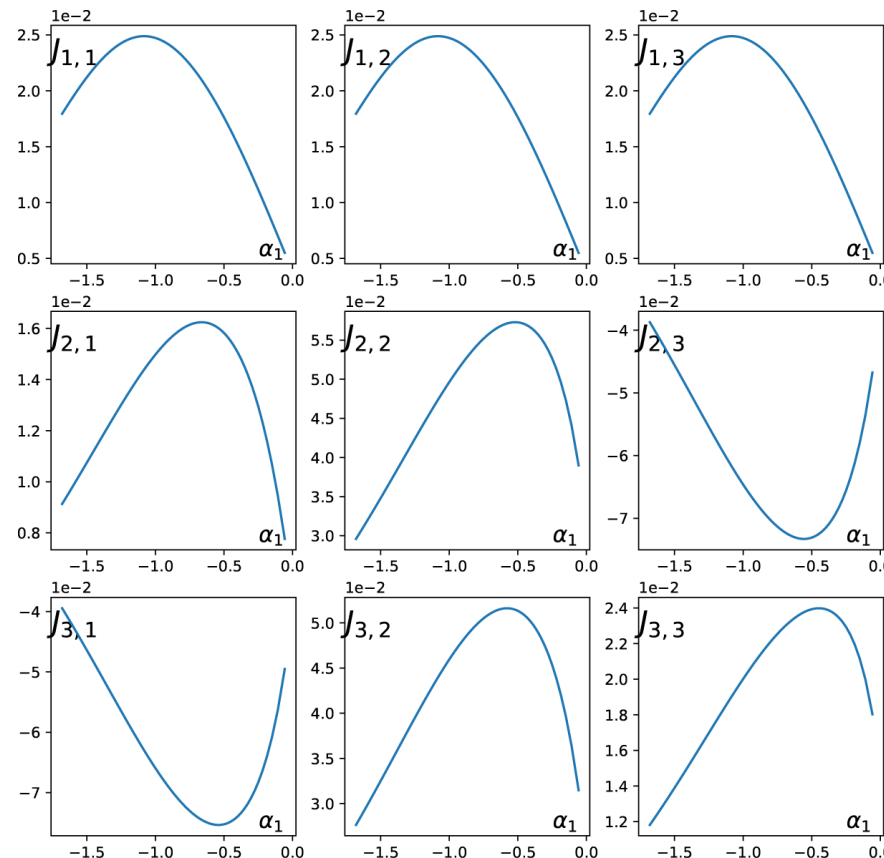
Egalité des travaux élémentaires

$$\delta W_s = \delta W_e$$

$$[J]^T \cdot F_r = C_i$$

$$F_r = [J]^{T^{-1}} \cdot C_i$$

$$F_r = \begin{bmatrix} J_{1,1} & J_{1,2} & J_{1,3} \\ J_{2,1} & J_{2,2} & J_{2,3} \\ J_{3,1} & J_{3,2} & J_{3,3} \end{bmatrix}^{T^{-1}} \cdot C_i$$



### 2- Résolution

Inverser une matrice revient à trouver la solution de :

$$[J]^T \cdot [M] = [I_n]$$

$$\Leftrightarrow \text{Résoudre 3 systèmes de 3 équations}$$

$$\begin{cases} [J]^T \cdot \{M_1\} = \{E_1\} \\ [J]^T \cdot \{M_2\} = \{E_2\} \\ [J]^T \cdot \{M_3\} = \{E_3\} \end{cases}$$

Implémentation

```
J=Fc.GetDeltaJacobian() [0]
I=np.identity(len(J))

def transvection(A,b,i,j,mu):
    A[i,:]=A[i,:]+mu*A[j,:]
    b[i]=b[i]+mu*b[j]
    return(A,b)

def triangulation(A,b):
    for i in range(n-1):
        pivot=(A[i][i])
        for j in range(i+1,n):
            A,b=transvection(A,b,j,i,-A[j,i]/pivot)
    return(A,b)

def remontee(A,b):
    xi=[0.]*n
    for i in range(n-1,-1,-1):
        temp= b[i]
        for k in range (i+1,n):
            temp= temp - A[i,k] * xi[k]
        xi[i]=temp/A[i,i]
    return(xi)

M=np.zeros(shape(J))
for i in range(n):
    U,b=triangulation(J,I[:,i])
    M[:,i]=remontee(U,b)
```

**3- Analyse et validation**  
Comparaison avec les valeurs annoncés et mesurées sur le robot



[www.setdidact.com](http://www.setdidact.com)