Robot ShapIO

Livret de présentation pédagogique Applications en CPGE

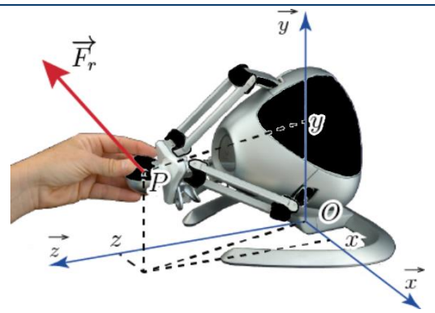
Système réel et Technologie connectée

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).

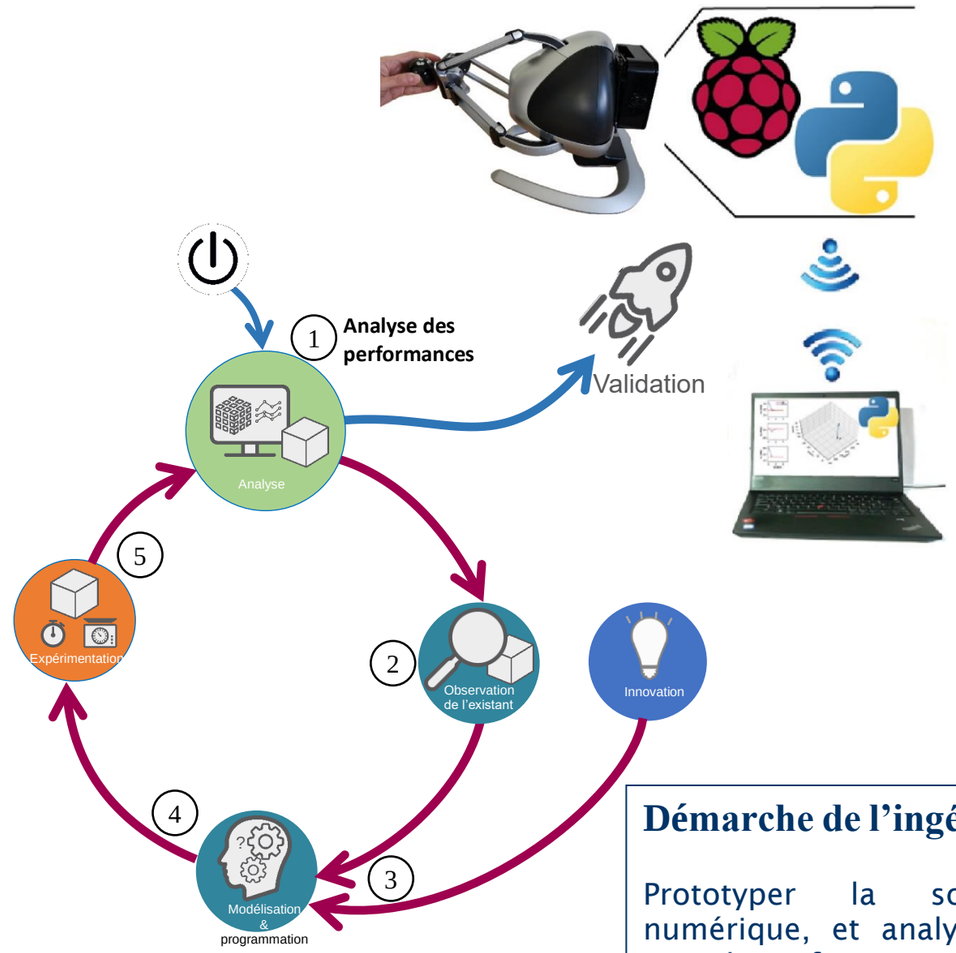
Sa complexité et son interface homme-machine physique en font un support idéal pour la découverte, la mise en œuvre et l'assimilation des méthodes de l'Ingénierie Numérique et de l'Intelligence Artificielle.

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

Système Etude Technique

Ce livret présente des extraits des activités pratiques ou dirigées incluses dans les dossiers d'accompagnement du système didactique "ShapIO", afin de donner un aperçu de ses potentialités pédagogiques.
Ces activités sont des propositions que le professeur peut exploiter en tout ou partie dans la construction de sa progression pédagogique, ou pour une réutilisation dans le cadre d'un projet.

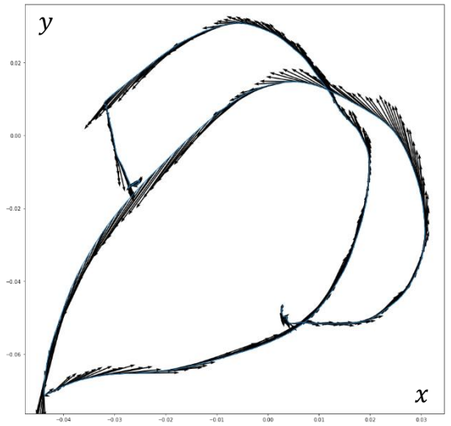
COMPETENCES ET CONNAISSANCES VISEES	Exemples d'activités pédagogiques proposées

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**

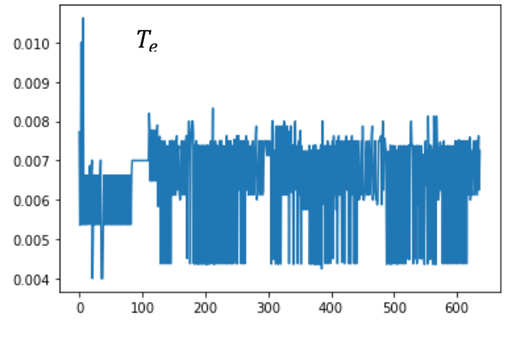
MESURER LA POSITION ET ESTIMER LA VITESSE

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



Trajectoire & champ des vitesses

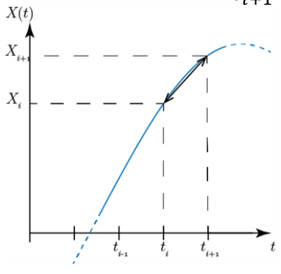
2- Analyser le comportement
Quantifier la période d'échantillonnage et sa régularité



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

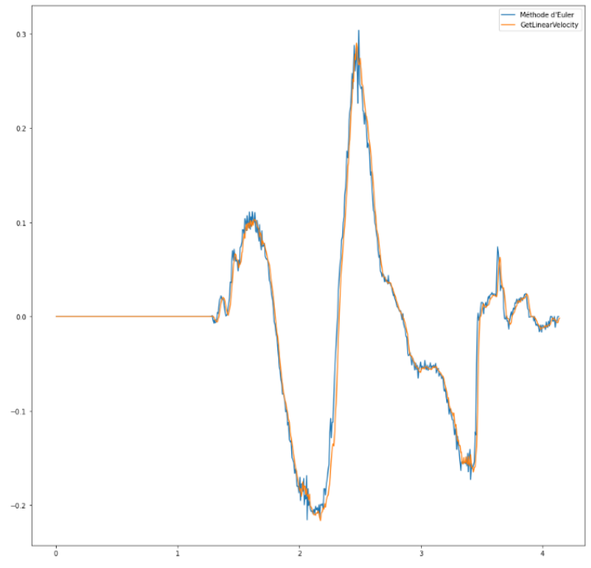
Approximation d'Euler avant

$$V_{t_i} = \frac{X_{i+1} - X_i}{t_{i+1} - t_i}$$




```
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

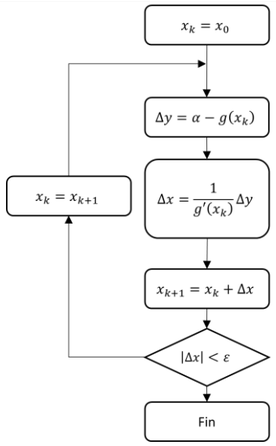
Exemples d'activités pédagogiques proposées

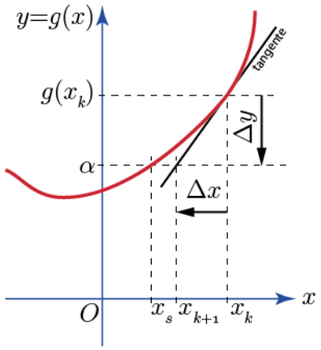
THEME :

- **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**

ESTIMER LA POSITION DE LA POIGNEE
Méthode de Newton & usage en robotique
(résolution d'équation)

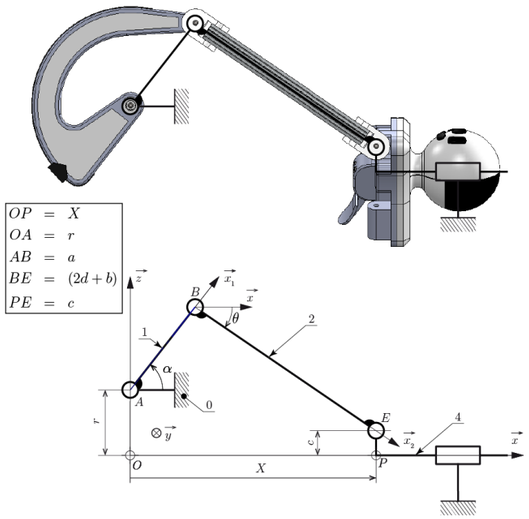
1- Rappel méthodologique

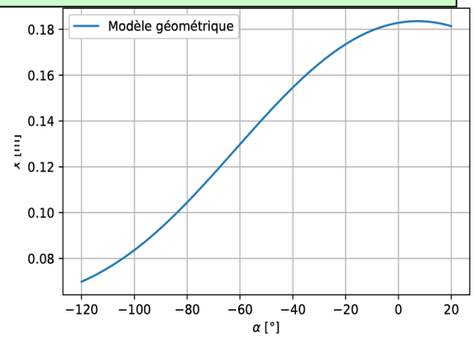


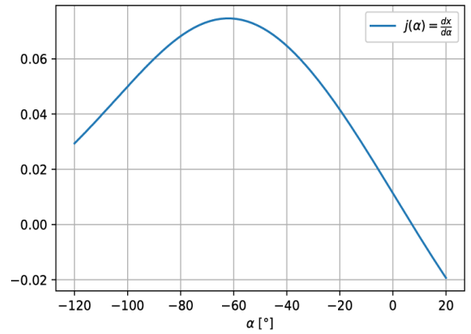


2- Cas du problème plan :

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



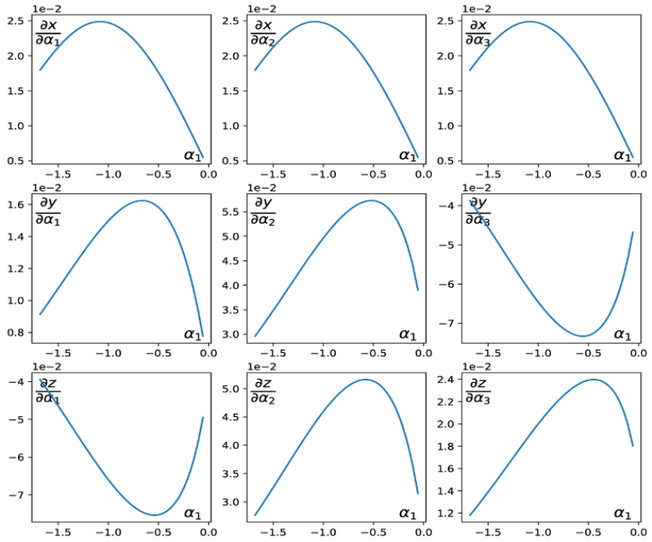


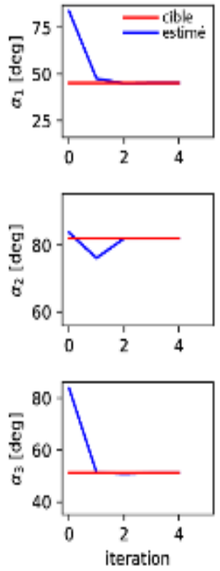


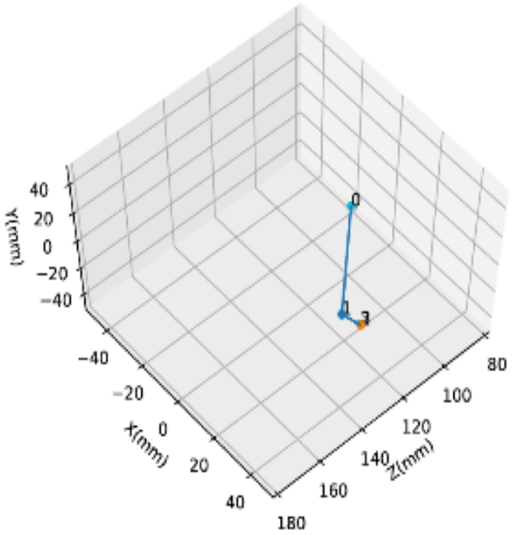
3- Cas du problème dans l'espace : recherche multivariable

- pas de solution mathématique
- analyse de la convergence

$$d\boldsymbol{X} = [J]d\boldsymbol{\alpha}$$







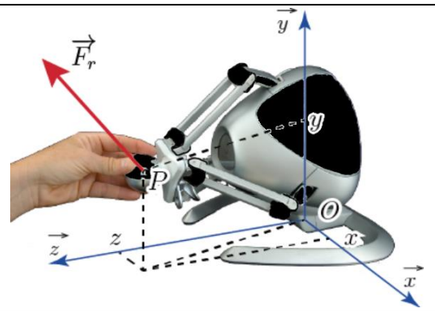
COMPETENCES ET CONNAISSANCES VISEES	Exemples d'activités pédagogiques proposées

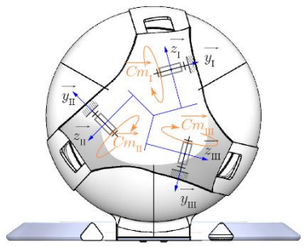
THEME :

- **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**

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

1- Mise en place du problème





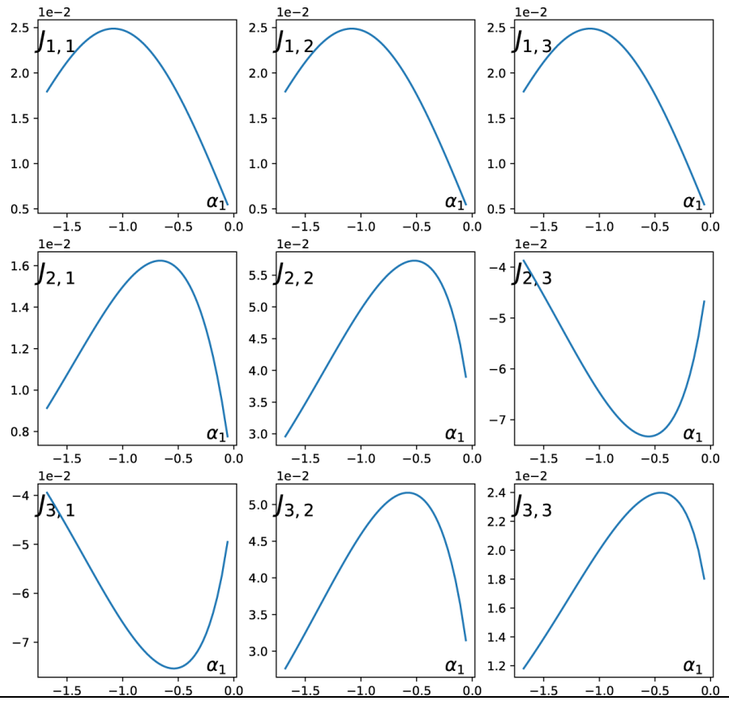
Egalité des travaux élémentaires

$$\delta W_s = \delta W_e$$

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

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

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



2- Résolution

Inverser une matrice revient à trouver la solution de :

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

⇔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

```python
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

COMPETENCES ET CONNAISSANCES VISEES	Exemples d'activités pédagogiques proposées

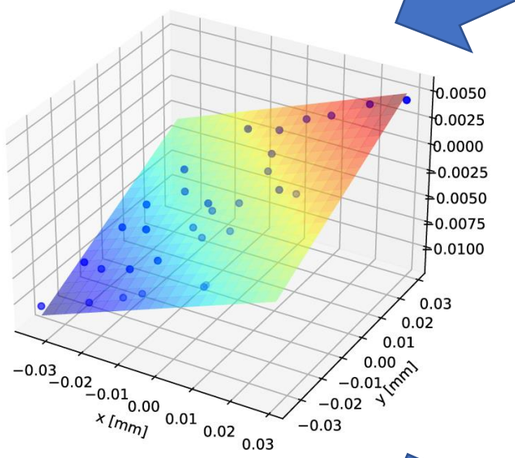
THEME :

- **Apprentissage supervisé.**
- **Phases d'apprentissage et d'inférence.**
- **Choix des données d'apprentissage.**
- **Modèle linéaire monovariable ou multivariable.**
- **Réseaux de neurones (couches d'entrée, cachées et de sortie, neurones, biais, poids et fonction d'activation).**
- **Choix et mise en œuvre des algorithmes**

VIRTUALISER UN OBJET
L'Intelligence Artificielle est utilisée pour reconnaitre l'objet et ses propriétés, et restituer un retour haptique de son jumeau numérique.

Apprentissage

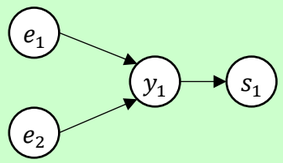
Apprentissage

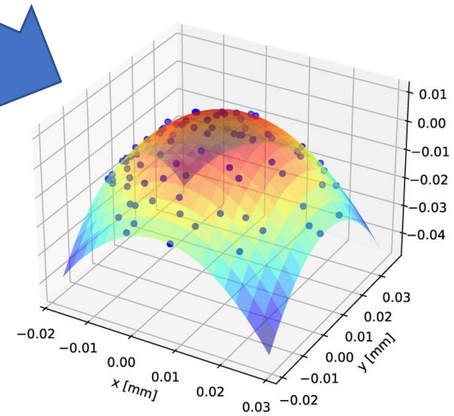


APPRENTISSAGE
Méthodes utilisées :

- Moindres carrés et régression linéaire multivariable
- Usage des bibliothèques dédiées
- Réseau de neurones (perceptron)

Paramétrage du modèle
et mesure de performance

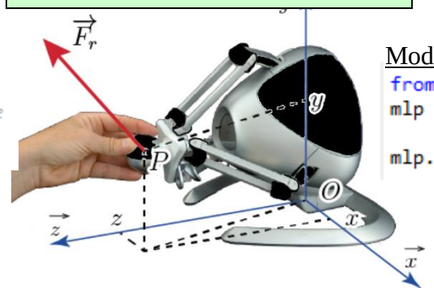




Inférence

Inférence

INFERENCE
Le modèle identifié est utilisé pour calculer et restituer l'effort afin de faire ressentir l'objet virtualisé



Modèle pour la régression linéaire

```
from sklearn import linear_model
clf = linear_model.LinearRegression()    #définition du type de modèle
clf.fit(X, Y)                            #apprentissage
```

Modèle pour le réseau de neurones

```
from sklearn.neural_network import MLPRegressor
mlp = MLPRegressor([500,200],activation='relu', solver='adam',
                   max_iter= 100000,  tol=1e-16,verbose=True)
mlp.fit(X, Y)
```

Shap IO : IA : Approche pédagogique

Démarche inductive :		découverte					Approfondissement
		TP1	TP2	TP3	TP4	TP5	TP6
		Découverte de l'« I.A. »	L'« I.A. » par u n module dédié		L'« I.A. » par réseau de neurones		
		Régression linéaire	Régression linéaire	Régression polynomiale	Régression linéaire	Fonction d'activation	Problème complexe
Durée		2h	1h	1h	1h	1h	1h
Nature de l'objet étudié		plan	plan	complexe	plan	prisme	complexe
Objectif pédagogique		Régression linéaire multivariable : identification et usage	Découverte d'un module d'Intelligence Artificielle	Prétraitement pour adapter les données à l'outil	Découverte du perceptron	Fonction d'activation pour un modèle linéaire par parties	Choix de stratégie et taille du réseau de neurones
Prérequis		Structures algorithmiques, listes & arrays (voir TP Spé SI), équation d'un plan	TP1	TP2	TP1	TP4	TP3 & TP5
Modules utilisés		numpy (array, solve), matplotlib (plot3D, scatter, surface),	sklearn (linreg, fit & predict)	sklearn (polyfit)	sklearn (perceptor)	sklearn (activation function)	sklearn (multi layer perceptor)
Programmes fournis							
Préparation		decouverte.py	decouverte.py	decouverte.py	decouverte.py	decouverte.py	decouverte.py
		palpage.py	palpage.py	palpage.py	palpage.py	palpage.py	palpage.py
		affichage_points.py	affichage_points.py	affichage_points.py	affichage_points	affichage_points	affichage_points
Apprentissage		Moindre_carre.py	sklearn_plan	sklearn_citron	ann_sklearn_plan	ann_sklearn_biplan	ann_sklearn_citron
inférence	affichage de l'objet appris	inference1.py	inference1.py	inference1.py	inference1.py	inference1.py	inference1.py
	restitution de l'objet appris	inference2.py	inference2.py	inference2.py	inference2.py	Analyse de performances	Analyse de performances
	utilisation d'une I.A. fournie		inférence3.py	inférence3.py	inférence3.py		

python powered

scikit learn

www.setdidact.com