

TP ADAPI

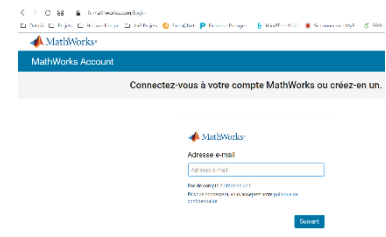
Introduction Partie 1

La première partie a pour but de vous faire découvrir les principes de quelques méthodes utilisées pour l'analyse de données industrielles et d'appliquer ces méthodes sur différents cas d'étude.

Pour cette première partie, le logiciel utilisé est Matlab.

Vous pouvez disposer d'une licence campus sur votre machine personnelle.

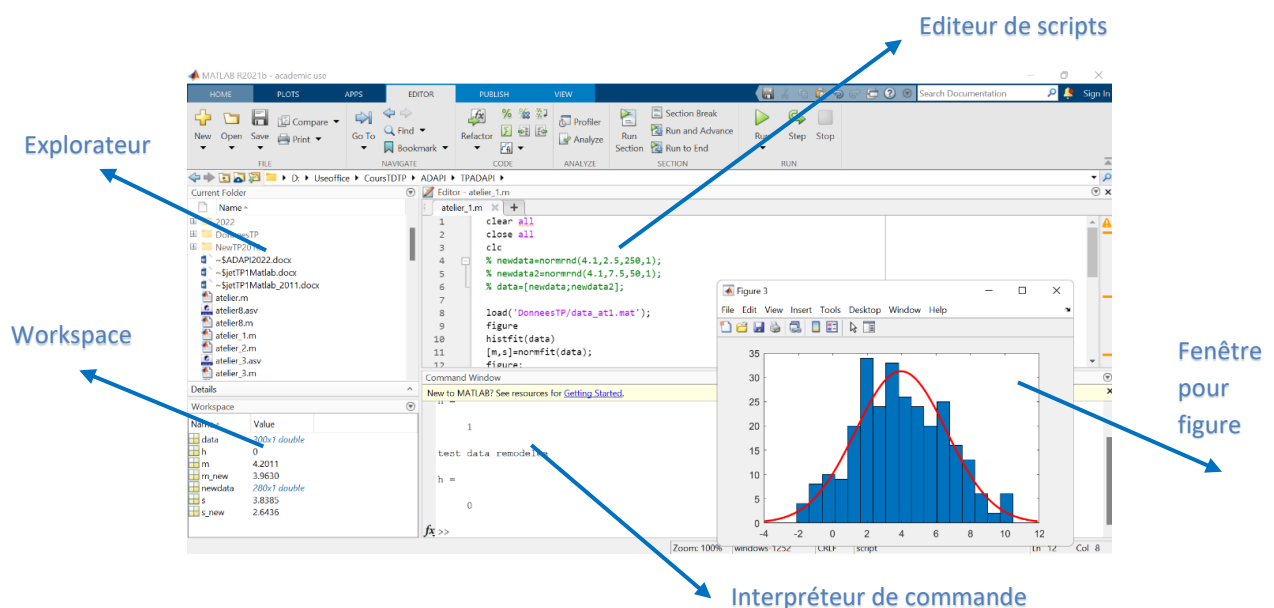
Pour cela, il faut créer un compte sur mathworks (<https://fr.mathworks.com/>) avec votre adresse en @etu.imt-nord-europe.fr.



A noter que des ressources et des formations sont disponibles sur le portail de l'école : <https://fr.mathworks.com/academia/tah-portal/imt-nord-europe-31274703.html>, dont une pour découvrir le logiciel :

https://matlabacademy.mathworks.com/details/matlab-onramp/gettingstarted?s_tid=course_mlor_start1

La licence classroom vous permettra d'accéder à de nombreuses toolboxes, de la représentation symbolique des équations jusqu'à la simulation multiphysique. Les toolboxes nécessaires pour ADAPI sont Signal Processing, Control Systems, Deep Learning, Statistics and Machine Learning.



Nous vous demandons, pour chaque atelier, de travailler avec un fichier « m » et pas avec l'interpréteur de commande directement. De plus, vous sauvegarderez dans votre répertoire un fichier « m » pour chaque atelier. Pour chaque atelier, nous vous proposons d'utiliser l'entête suivant : dessous le début du fichier « m ».

```
% initialisation de affichage
close all
clear all
clc
% chargement des données (X=numéro de
l'atelier de 1 à 8)
load('DonneesTPAdapi\data_atX.mat')
```

Après avoir chargé chaque fichier de données, nous vous conseillons de vous assurer de son contenu en le visualisant à l'aide du workspace de MATLAB.

Les 8 ateliers à réaliser sont les suivants :

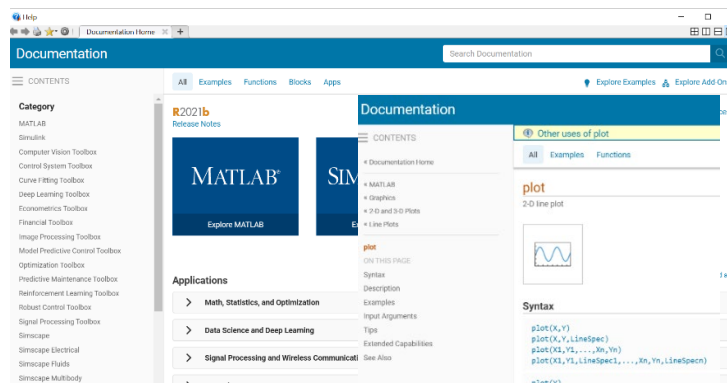
- Atelier 1 : Analyse de distributions normales,
- Atelier 2 : Détection de changements dans une série temporelle,
- Atelier 3 : Analyse en composantes principales,
- Atelier 4 : Modélisation par régression linéaire,
- Atelier 5 : Classification par arbre de décision,
- Atelier 6 : Clustering par kMeans,
- Atelier 7 : Identification de systèmes par réseaux de neurones,
- Atelier 8 : Modélisation de systèmes complexes, étude des influences.

Chaque fichier « m » créé par atelier sera stocké dans votre répertoire de travail, situé à la racine du PC C:\ADAPI_VI avec VI correspondant à vos initiales.

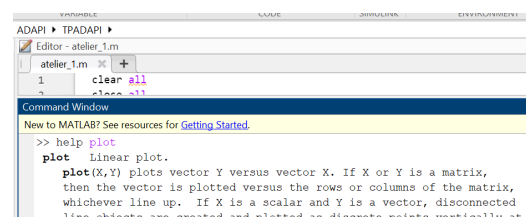
Pour la réalisation de ces ateliers, La plupart des fonctions utilisées sont issues des toolboxes **Statistics** et **Neural Network**. Il vous sera demandé de parcourir la documentation en ligne de Matlab accessible depuis l'interpréteur de commandes ou depuis la barre de menus.

DOC
Doc plot

Cette aide est structurée par arborescence, classant les fonctions par toolbox puis par type



A noter l'autre instruction d'aide `help`, qui permet directement de visualiser dans l'interpréteur de commandes l'entête et les paramètres d'une fonction.



Atelier 1: Analyse de distributions normales

Le but de cet atelier est d'améliorer la qualité de données en cherchant à normaliser la distribution.

Dans un premier temps, il s'agit de visualiser et déterminer les caractéristiques d'une loi normale caractérisant au mieux les données. Pour cela, vous disposez des fonctions Matlab suivantes :

`histfit; qqplot, normplot`

Le fichier à charger s'intitule `data_at1.mat`



Interprétez et comparez les résultats sur les données brutes.

Dans un second temps, pour corriger les données, vous cherchez à éliminer les données aberrantes.



Comment la fonction '`normfit`' (qui permet d'estimer moyenne et covariance d'une fonction normale collant au mieux aux données) pourra être utilisée ? On cherchera à ne conserver qu'environ 90% des données brutes initiales

Astuce : pour éliminer les données aberrantes, il faudra utiliser la fonction `find` telle que : `newdata=data(find(data< seuilhaut &data > seuilbas))`.



En utilisant la fonction '`jbttest`' sur les données initiales puis les données modifiées, montrez le bénéfice de cette opération.

Atelier 2 : Détection de changements dans une série temporelle

Dans cet atelier 2, Il s'agira dans cet atelier de s'initier à la détection de changement dans la distribution des données et d'étudier les performances de deux outils utilisés pour la détection de dérives.

Le fichier à charger s'intitule `data_at2.mat`. Ce fichier regroupe 6 signaux distincts organisés en colonnes.


Le script principal de cet atelier gère la sélection des signaux, l'appel aux fonctions `test_CUSUM` et `test_EWMA` (que vous allez réaliser) ainsi que l'affichage des seuils de décision et des résultats de la détection.

Les deux fonctions à réaliser sont les suivantes :

```
function [H,G]=test_CUSUM(X,K,ha)
```

```
function [H_E, LA, HA]=test_EWMA(X,WL,k)
```

X représente le signal à tester, l'une des colonnes du fichier d'entrée.

	<p>Coder les deux fonctions CUSUM et EWMA.</p>
---	--

Pour le test CUSUM :

K représente le paramètre de saut et h_A , le seuil d'alarme qui sera à comparer avec la fonction g.

Les paramètres de sortie sont 2 vecteurs : l'un binaire (H) indique par un 1 les endroits de la détection d'un saut ; G retourne les valeurs de la fonction de « somme cumulée » g à chaque échantillon.

Pour commencer la programmation, il vous sera nécessaire de rappeler la forme de la fonction g.

Un programme simple consiste à calculer la somme cumulée à chaque échantillon, puis de comparer le résultat avec le seuil h_A et de calculer la valeur booléenne de H à cet échantillon (mise à 1 si la somme est supérieure au seuil).

Remarque pratique : Concernant la programmation, la moyenne sera initialisée avec les 50 premiers échantillons du signal.

Pour le test EWMA :

Les paramètres d'entrée sont X, le signal à tester ; WL, la largeur de la fenêtre glissante et k le coefficient à appliquer sur la tolérance.

Les paramètres de sortie sont 3 vecteurs, l'un binaire (H_E) indique par un 1 les endroits d'une détection; LA et HA retournent les valeurs des seuils de décision.

La programmation consistera à comparer chaque échantillon avec un seuil limite bas et un seuil limite haut, déterminés selon une proportion de $k \cdot \sigma$ autour de la moyenne glissante. La sortie H_E sera mise à 1 lorsque la valeur de l'échantillon est en dehors de cette plage.

Remarque pratique : Concernant la programmation, la moyenne et l'écart-type glissants seront calculés à partir de l'échantillon i-1 et des WL échantillons les plus récents.

Après avoir réalisé ces deux fonctions, il vous est demandé de les tester sur les signaux présents.

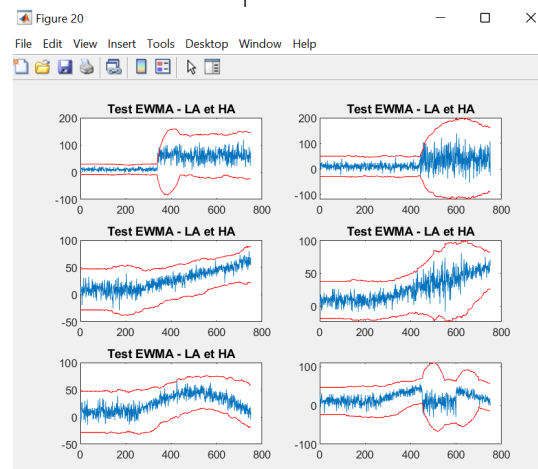


Discuter de l'influence des paramètres, fixés par l'utilisateur, pour les deux tests et pour chacun des signaux présents.

Pour cela, il vous sera nécessaire de justifier leurs performances en fonction de la nature des signaux. Pour commencer, commentez brièvement la nature des signaux puis évoquez pour chacun les critères de choix des paramètres des tests.

Astuce : Pour vous simplifier cette comparaison et cette discussion, regroupez les résultats sur les 6 signaux dans une grille 3x2, à l'aide de la fonction `subplot`.

```
for i=1:6
    [H_E, LA, HA]=test_EWMA(X(:,i),WL,k);
    figure(20)
    title('Test EWMA - LA et HA');
    subplot(3,2,i);
    plot(X(:,i))
    hold on
    plot(LA,'r-');
    plot(HA,'r-');
    figure(21)
    title('Decision');
    subplot(3,2,i);
    plot(H_E,'g-')
end
```



Atelier 3 : Analyse en composantes principales

Cet atelier vise à projeter les données brutes dans un espace à 2 dimensions afin de visualiser les ressemblances entre les véhicules.

Le fichier à charger s'intitule `data_at3.mat`.

La base de données est constituée d'un ensemble de véhicules caractérisés par différents attributs (MPG, Weight, DriveRatio, Horsepower, Displacement, Cylinders).




Dans un premier temps, analysez la dispersion des valeurs avec la fonction `'boxplot'`. Qu'en concluez-vous ? Afin de poursuivre l'analyse des dépendances, vous pouvez également recourir aux deux fonctions suivantes : `'cov'` et `'corr'`


La première modification consiste à centrer les observations en supprimant la valeur moyenne par composantes.

Astuce : Pour centrer les données, il est possible d'utiliser la fonction `repmat(moyenne, N, 1)` afin de créer un tableau dont les n lignes représentent le vecteur moyen

La matrice des données centrées ayant été obtenue, la diagonalisation de la matrice de covariance est effectuée par décomposition en valeurs propres et vecteurs. Pour cela, il vous faudra recourir à la fonction `'eigs'`.


	Estimer les valeurs propres et vecteurs propres de la covariance de la matrice des données centrée.
---	---

Après obtenu la matrice des valeurs propres, tracer l'histogramme suivant : `bar(Val_Prop/trace(Val_Prop))` :


	Qu'en concluez en vous ?
---	--------------------------

Après avoir extrait les deux vecteurs propres (V2D) correspondant aux deux valeurs propres les plus fortes (D), vous projetterez les données initiales dans un espace à 2D

```
newdata=data_norm*V2D*sqrt(inv(D));
```

	Donner la représentation graphique des observations projetées : <pre>plot(newdata(:,1),newdata(:,2),'+') xlabel('1ere composante'); ylabel('2ieme composante');</pre> En déduire les véhicules qui semblent les plus proches.
---	---

Astuce : Pour visualiser le type de voiture associé à chaque, il est possible d'utiliser la fonction `gname(name_car)`. En cliquant sur un point de la figure, le type apparaîtra.

	Afin de définir l'importance relative des caractéristiques, tracer dans l'espace réduit, leurs vecteurs originaux correspondant.
---	--

Atelier 4 : Modélisation par régression linéaire

Le but de cet atelier consiste à définir et à caractériser le modèle linéaire obtenu par la méthode des moindres carrés.

Le fichier à charger s'intitule `data_at4.mat`.



La première colonne des données correspond au numéro de l'observation.

Les colonnes 2 à 5 caractérisent les composantes du vecteur d'entrée X .



La colonne 6 caractérise le vecteur de sortie y .

Avant de commencer, il vous faut justifier l'intérêt de tester une modélisation par régression en estimant la corrélation entre les données.


`corr(data(:,2:end))`

	Ayant discuté de la corrélation des données, appliquez la méthode des moindres carrés pour déterminer θ tel que $y = X\theta + \varepsilon$.
	A partir de $\hat{\theta}$, calculez ensuite \hat{y} puis tracez sur une même figure y et \hat{y} puis, sur une autre figure, l'erreur entre ces deux signaux.


On souhaite améliorer l'estimation en supprimant les données anormales.

	En se basant sur l'atelier 1, proposez une méthode d'analyse des résidus permettant de donner l'indice (le numéro de l'observation) pour lequel l'erreur est dans les 5% des valeurs limites.
	Déterminez les deux nouveaux ensembles <code>newX</code> et <code>newY</code> tels que : <code>newX=X(find(E< seuilhaut&E> seuilbas),:)</code> <code>newY=Y(find(E< seuilhaut &E> seuilbas))</code>

Il s'agit ensuite d'estimer θ_{new} en utilisant ces nouveaux ensembles.

	Tracez l'erreur obtenue avec cette nouvelle estimation θ_{new} et concluez.
---	---

Pour terminer cet atelier, vous pouvez explorer les méthodes Matlab 'regress' et 'rcoplot'.

	Comparer avec vos résultats.
---	------------------------------

Atelier 5 : Classification par arbres de décision

Cet atelier vise à caractériser la qualité de fabrication d'un produit à partir de quatre caractéristiques : température moyenne (°C), ensoleillement (h), jours de chaleur (j), millimètres de pluie (mm).

Le fichier à charger s'intitule `data_at5.mat`

X=data(:,1:4) correspond aux mesures selon cette caractéristique.

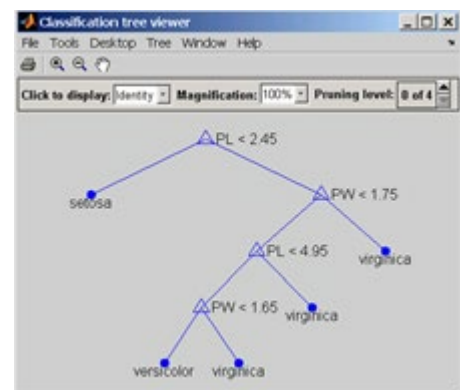
Y=data(:,5) correspond aux labels, variable ordinale.




Z correspond aux labels, variable catégorielle correspondant à la qualité.


La construction d'un arbre consiste à trouver, à chaque niveau (cf. exemple de figure Matlab ci-après : triangle = nœud ; . = classe), un critère de décision pour **un attribut** du vecteur X qui permette de séparer l'ensemble des données en deux groupes avec une meilleure homogénéité des classes représentées.

On se propose d'analyser visuellement les données selon les 3 classes.

```
Data=[X,Y];
Test=sortrows(data,5);
figure
subplot(2,2,1);
plot(Test(:,1),Test(:,5),'+')
subplot(2,2,2);
plot(Test(:,2),Test(:,5),'o')
subplot(2,2,3);
plot(Test(:,3),Test(:,5),'gd')
subplot(2,2,4);
plot(Test(:,4),Test(:,5),'k.')
```





	Sur la base de cette visualisation, proposez, d'une façon empirique, un premier critère de scission de la population.
	Quels critères contribueront à la détermination du nombre de nœuds de l'arbre final ?
	<p>Complétez votre analyse en vous aidant de la fonction 'fitctree'</p> <pre>X=data(:,1:4); T=fitctree (X,Z) view(T,'Mode','graph')</pre> <p>Expliquez le rôle du paramètre 'MinParentSize' ?</p>

	Quelle est l'utilité de la fonction 'resubLoss(T, 'Subtrees', 'all')' ?
---	---

On se propose maintenant d'étudier la classification par arbre de décision sur un nouvel angle. Ecrire et tester le programme proposé ci-après :

```
D2_X=pdist(X)
B=squareform(D2_X)
C2_D = linkage(D2_X, 'average');
[H,T] = dendrogram(C2_D, 'colorthreshold', 'default');
```

	Que retourne la matrice B ?
	En déduire la manière utilisée pour construire l'arbre ascendant ?

Atelier 6 : Clustering par kMeans


Cet atelier concerne l'analyse des différents modes de fonctionnement d'une presse hydraulique. Le fichier de données consiste en 17 mesures : 11 pressions, 3 débits, 2 températures et 1 position du vérin. Il s'agit d'extraire les modes de fonctionnement du procédé par la méthode des centres mobiles.



Le fichier à charger s'intitule data_at5.mat.

Pour commencer, nous définissons k, la variable caractérisant le nombre de groupes à extraire. L'initialisation consiste à extraire k vecteurs aléatoirement (les centres initiaux) dans les N observations.

```
c(i,:) = hydrau8(floor(N*rand(1,1)), :);
```

	Créez les K-groupes initiaux en agglomérant les observations les plus proches des centres initiaux.
---	---





L'algorithme k-means étant itératif, il vous **est demandé de créer** une fonction fitcenter telle que :

```
Function newC = fitcenter(hydrau8,C,k);
```

Cette fonction exploitera les principales instructions suivantes :

- $y(:,i)=\text{dist}(X,C(i,:))'$; pour le calcul des distances entre chaque observation et le centre i
- $Z(i,1)=\text{find}(y(i,:)'\leq\min(y(i,:)'))$; pour l'affectation au groupe i selon la distance la plus faible
- $\text{GrpI}=X(\text{find}(Z==i),:)$; pour la création du groupe i composé des données X affectées à i
- $\text{newC}(i,:)=\text{mean}(\text{GrpI})$; pour le calcul du nouveau centre

Cette fonction créée, dans le script principal, détailler la boucle « `while` » permettant d'exploiter cette fonction selon le principe de l'algorithme des centres mobiles.

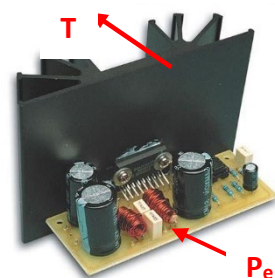
	Appliquez itérativement cette fonction sur les données <code>hydrau8</code> ; à chaque boucle (k) affichée l'écart entre $C(k)$ et $C(k-1)$ tel que : $(C-\text{newC})*(C-\text{newC})'$
	En déduire un critère d'arrêt de votre algorithme itératif.
	Montrez que vous obtenez un résultat cohérent avec celui de la méthode Matlab ' <code>kmeans</code> '. $[\text{IDX},\text{Centre}]=\text{kmeans}(\text{hydrau8},3)$
	Poursuivez votre analyse de la méthode en exploitant la fonction <code>silhouette</code> et en justifiant du choix optimal de k a posteriori. $\text{silhouette}(\text{hydrau8},\text{IDX})$

Atelier 7 : Identification de systèmes par réseaux de neurones

Le procédé à modéliser est un dissipateur thermique. Il s'agira de déterminer l'évolution de la sortie température selon un coefficient de puissance électrique appliqué au composant électronique. Le dissipateur a pour but d'évacuer la chaleur emmagasinée par le composant.

Le fichier à charger s'intitule `data_at6.mat`. La base de données est constitué de 3 variables `coef_puis` (puissance électrique en entrée), T (température de sortie), `temps` (date des échantillons – facultative) de 50 000 mesures.

Dans cet atelier, il s'agira de proposer la structure du modèle à identifier puis d'utiliser un réseau de neurones de type



backpropagation pour estimer un modèle de comportement entrée-sortie reproduisant les données.

Une base d'apprentissage sera constituée à partir des 10 à 20% des données présentes dans la base.



Sur la base de vos connaissances en électronique et en thermique, proposez une première structure pour le régresseur et le vecteur de sortie. En déduire les matrices y et X à utiliser pour l'apprentissage.

L'apprentissage est effectué grâce à la fonction `train` qui utilise comme variables les données de la base et la structure d'un réseau initialisé par la fonction `newff` (ou `feedforwardnet`).

```
HL=[nbN1 nbN2 nbN3]
net = newff(X,y,HL,{'t','l','m'});
net.trainParam.lr = 0.05;
net.trainParam.mc = 0.9;
net = train(net,X,y);
y_hat = sim(net,X)
```



Quelles sont les influences des variables `HL`, `lr`, `mc` et de la méthode `trainml` sur l'apprentissage ?.

Il s'agira ensuite d'estimer les performances de plusieurs structures en proposant un critère de performance (simple).



Après avoir (brièvement) justifié le choix du modèle jugé optimal, il est demandé de valider ses performances sur l'ensemble des données brutes disponibles. Concluez.


Atelier 8 : Modélisation de systèmes complexes, étude des influences

Dans ce dernier atelier de la partie 1, le procédé à analyser est multivarié, non-linéaire et soumis à diverses influences électriques, thermiques, chimiques difficilement modélisables. Ainsi, seule une modélisation non-linéaire du type boîte-noire est envisageable.

Le fichier à charger s'intitule `data_at7.mat`.

Les 7180 observations du procédé sont caractérisées par 13 variables d'entrée (les colonnes 2 à 14 de la matrice `data`) et 1 variable de sortie (la 15^{ème} colonne).


En utilisant un réseau de type backpropagation (feedforward) tel qu'utilisé dans l'atelier précédent, proposez un modèle reproduisant une sortie estimée proche de la sortie mesurée.


	Validez le choix du réseau en caractérisant l'erreur de sortie.
---	---

A partir du modèle choisi, ce qui suit consiste en l'étude de l'influence des entrées sur la sortie. Un premier principe sommaire est de tester le modèle obtenu avec différentes bases de synthèse. Chaque base de synthèse est constituée d'observations correspondant au vecteur moyen de la base d'apprentissage X (au centre du modèle) où seule l'une des caractéristiques d'entrée évoluera dans cette base. Cette évolution se fait selon deux fois l'écart-type de l'entrée concernée.

Le ratio entre l'écart-type de la sortie simulée et l'écart-type de la variable d'entrée donnera une idée de l'influence de l'entrée sur la sortie.

```
Mat_moyenne=repmat(mean(X),2L,1);
MTM=Mat_moyenne;
T=1:2L ;
Delta=std(X(:,k));
DS=Delta*(T-L)/L;
MTM(:,k)=MTM(:,k)+ DS ;
```

	Construisez ces 13 matrices MTM (une pour chaque variable d'entrée soumise à une variation) puis simuler votre modèle avec chacune des matrices utilisées comme entrée.
---	---

	Estimez l'écart-type de la sortie pour chaque base puis calculez le ratio entre les écart- types : $\text{var}(k) = \text{std}(\hat{y}_{\{k\}}) / \text{std}(X(:,k));$
---	---

	Terminez par l'affichage de l'histogramme des différentes influences. <pre>bar(var) set(gca,'XTicklabel',{'Bottom_freeze_layer','C O','CO2','CaO_flux','CrO_Cr2O3','Feed_power_pro duct','H2','MW','MgO','Ore_feed','Power_Feed', 'Reductant','slag_depth','metal_depth'})</pre>
---	---