الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

UNIVERSITE BADJI MOKHTAR - ANNABA
BADJI MOKHTAR – ANNABA UNIVERSITY

جامعة باجي مختار – عنابـــة

Faculté : Sciences de l'Ingéniorat

Département : Electronique

Domaine : Sciences et Technologies

Filière : Automatique

Spécialité : Automatique et Système

Mémoire

Présenté en vue de l'obtention du Diplôme de Master

Thème :

Programmation sous python des algorithme de diagnostique avec implémentation sur raspberrypi 4

Présenté par : *Salhi Aymen*

Encadrant : *Mohamed F.HARKAT* *Prof* *UBM Annaba*

Jury de Soutenance :

Ait Izem Tarek	MCB	UBM Annaba	Président
Mohamed F.HARKAT	Prof	UBM Annaba	Encadrant
Benmoussa Samir	Prof	UBM Annaba	Examinateur

Année Universitaire : 2020/2021

Résumé :

Le travail présenté dans cette thèse comprend le développement d'un système de détection des défauts dans les processus industriels selon des méthodes statistiques pour basé sur le traitement que nous avons mené par la recherche théorique sur l'Analyse en composantes principales a noyau et l'application de techniques sur le système pour détecter les défauts.

Le premier chapitre présente le modèle PCA, KPCA, et modèle KPCA réduit ainsi que les indices de détection.

Le deuxième chapitre, présente la carte " Raspberry Pi " avec sa constitution et ces domaines d'utilisation. Puis une petite présentation sur le python et son fonctionnement.

Le dernier chapitre est consacré à une application sur un procédé réel qui est le processus chimique TENNESSEE EASTMAN, en présentant l'implémentation sous Matlab et Python.

التلخيص

يتضمن العمل المقدم في هذه الأطروحة تطوير نظام للكشف عن العيوب في العمليات الصناعية وفقًا لأساليب إحصائية بناءً على المعالجة التي أجريناها من خلال البحث النظري حول تحليل مكونات النواة وتطبيق التقنيات على نظام لاكتشاف الأعطال.

يعرض الفصل الأول نماذج تحليل المكون الرئيسي وتحليل المكون الرئيسي للنواة وتحليل المكون الرئيسي للنواة المخفضة بالإضافة إلى مؤشرات اكتشاف الأخطاء.

الفصل الثاني، الدراسة النظرية لـ "الرازبيري بي" يقدم مع تكوينه ومجالات استخدامه. ثم عرض تقديمي بسيط عن "البايثون" وكيفية عمله.

الفصل الأخير مخصص لتطبيق على عملية حقيقية وهي العملية الكيميائية لـ "تينيسي ايستمان" ، من خلال تقديم التنفيذ بموجب "الماتالب " و "البايثون".

Abstract :

The work presented in this thesis includes the development of a system for fault detection in industrial processes based on statistical methods and using Kernel Principal Component Analysis.

The first chapter presents the PCA, KPCA, and reduced KPCA models as well as the fault detection indices.

The second chapter, the theoretical study of the "Raspberry Pi" is presented with its components and its fields of use. Then the python is presented.

The last chapter is devoted to an application on a real chemical process, the TENNESSEE EASTMAN process, by presenting the implementation under Matlab and Python.

Dédicaces

Avec amour et respect, ce modeste travail est dédié à

nos familles

A nos chères qui étaient à nos côtés avec amour,

soutien moral et les prières pour qu'on puisse

atteindre nos objectifs,

A tous les collègues et amis dans notre université.

Remerciements :

On tient à exprimer nos gratitudes et présenter nos chaleureux remerciements :

- Tout d'abord à Dieu tout puissant de nous avoir donné le courage, la force et la patience d'achever ce modeste travail.
- Notre reconnaissance s'adresse à notre encadreur Mr. Harkat Mohamed Faouzi que nous apprécions sa qualité professionnelle et intellectuelle, qui n'a pas cessé de nous prodiguer ses conseils et qui a épargné un grand effort pour contribuer à la réussite de notre travail.
- A tous les enseignants de notre département d'électronique – Annaba.
- A toutes les personnes qui nous ont apportés leurs aides et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette année universitaire malgré les circonstances difficiles.

Table des matières

Liste des figures

Liste des tableaux

Notations

Introduction générale

Liste des figures :

Liste des tableaux :

Notation :

X : Matrice de données d'entrainement.

x : Vecteur de mesure.

m : Nombre de variables.

N : Nombre d'échantillons.

Σ : La matrice de covariance de X.

Λ : Matrice de valeurs propres diagonales.

σ : Largeur d'une fonction gaussienne.

ϕ : Fonction de mappage.

K : La matrice Kernel.

PCA : Analyse des composantes principales.

KPCA : Analyse des composantes principales du noyau.

RBF : Radial Basis Function (Fonction de base radiale).

ℓ : Nombre de composantes principales conservés.

RKPCA : Analyse des composantes principales du noyau réduite.

CPV : Cumulative Percentage of Variance (PCV : pourcentage cumulé de la variance).

T^2 : La statique de Hotelling.

Q : Squared Prediction Error (erreur de prédiction au carré).

ϕ : Statistique combinée.

$\tau_\alpha^{T^2}$: Limite de contrôle associée à l'indice T^2.

τ_α^{Q} : Limite de contrôle associée à l'indice ϕ.

Introduction générale :

Le but de ce travail était de développer un système de détection des défauts des processus industriels en se basant sur les techniques statistiques de traitement de données et son implémentation sur la carte Raspberry pi 4.

L'approche adoptée dans ce travail est l'analyse en composants principales à noyaux (KPCA).

Le problème majeur dans le modèle KPCA est sa dimension très élevée, pour une implémentation sur carte il est impératif de réduire le modèle.

Ainsi, un modèle réduit appelé R-KPCA est développé. Ce modèle est implémenté en première phase sous Matlab pour procéder aux tests nécessaire à la validation de l'approche proposée. Ensuite il est implémenté sous Python 3.8 pour une éventuelle implémentation sur carte Raspberry pi 4.

Le mémoire de master est organisé en 3 chapitres.

Dans le premier chapitre explique le modèle PCA, KPCA et le modèle KPCA réduit, ainsi que les indices de détection des défauts.

Le chapitre 2, présente la carte Raspberry pi 4, ses caractéristiques et son architecture.

Le dernier chapitre est consacré à une application sur un procédé chimique « Tennessee Eastman » en présentant l'implémentation sous Matlab et sous Python.

Chapitre 1

1.1 Introduction :

L'analyse en composantes principales (ACP) est une technique descriptive permettant d'étudier les relations qui existent entre les variables, sans tenir compte, a priori, d'un quelconque modèle. Le but de l'ACP est d'identifier la structure de dépendance entre des observations multi variables afin d'obtenir une description ou une représentation compacte de ces dernières. L'analyse en composante principale peut être vue comme une technique de projection orthogonale linéaire qui projette les observations multidimensionnelles représentées dans un espace de dimension m (m est le nombre de variables observées) dans un sous-espace de dimension inférieure $l < m$ en maximisant la variance des projections. Le calcul de distances par rapport à ces axes sert d'outil de détection de valeurs aberrantes.

La majorité des processus industriels sont non linéaire et dynamique, l'application de **l'ACP** n'est pas très adaptée de ce type de systèmes et peut donner donc de mauvaises interprétations sur l'état du système c'est pour cela qu'on utilise une fonction noyau appelée **Kernel** qui as pour but de faire la représentation de donnés non linéaires dans un autre espace linéaire.

Le problème majeur dans le modèle **Karnel PCA** est sa dimension très élevée, pour une implémentation sur carte il est impératif de réduire le dit modèle. Ainsi, un modèle réduit appelé **RKPCA** est développé.

1.2 Préliminaires :

1.2.1 Méthode de l'analyse en composante principale (PCA) :

L'analyse en composantes principales (ACP) est une méthode mathématique d'analyse graphique de données qui consiste à rechercher et mettre en évidence les relations qui existent entre les variables, sans tenir compte, à priori d'une quelconque structure, et élabore un modèle du système à partir de données prélevées sur ce dernier L'ACP élabore un modèle du système à partir de données prélevées sur ce dernier.

L'identification du modèle repose sur deux étapes : la première consiste à estimer ses paramètres alors que la seconde consiste à déterminer sa structure.

Une fois le modèle ACP identifié, des résidus peuvent être générés en comparant le comportement observé celui donné par le modèle ACP de référence, Ces résidus permettent de détecter puis de localiser l'ensemble des variable en défaut.

Le but de l'ACP est donc de trouver un ensemble de facteurs (composantes) qui ait une dimension inférieure à celle de l'ensemble original de données et qui puisse décrire correctement les tendances principales.

.

Soit $(k) \in \mathbb{R}^{m}$ un vecteur de mesure échantillon de m capteurs à temps k, En supposant qu'il y a N échantillons pour chaque capteur, une matrice de données

$X = [x(1)\ x(2)\ \dots\ (N)]$ $T \in \mathbb{R}^{N \times m}$ est normalisé à une moyenne nulle et une variance unité. Alors l'ACP peut être effectué par la décomposition des valeurs propres de la matrice de covariance X, Σ.

$$\Sigma = P\ \Lambda\ P^{T} \qquad (1)$$

Où $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_m)$ est la matrice des valeurs propres de la diagonale ($\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$) et $P = (p_1, p_2, \dots, p_m)$ est la matrice des vecteurs propres. (p_i , $i = 1,2, \dots, m$, Représentent les vecteurs propres normalisés et mutuellement orthogonaux associé aux valeurs propres)

Ensuite, la matrice X est transformée en variables indépendantes T par :

$$T = X\ \mathrm{P} \qquad (2)$$

Où $T = [T_1, T_2, \dots, T_m]$ contient les composantes principales, qui sont orthogonaux les uns aux autres. Le modèle PCA apparaît sous forme de chargements, de scores et d'écarts. Les vecteurs propres (chargements) associés aux valeurs propres de la matrice de covariance des données couvrent les sous-espaces principal et résiduel.

Le sous-espace principal associé aux premières composantes principales (scores) décrit les variations importantes du processus. Cependant, le sous-espace résiduel associé au reste des composants principaux caractérise le bruit dans les données.

En résumé, le modèle PCA est déterminé sur la base d'une décomposition propre de la matrice de covariance Σ et la sélection du nombre ℓ de composantes à retenir.

Les matrices de valeurs propres, vecteurs propres et composantes principales peut être partitionné comme :

$$\Lambda \begin{pmatrix} \Lambda\ell_{\ell} & 0 \\ 0 & \Lambda_{m-\ell} \end{pmatrix} \qquad (3)$$

$$P = [P_{\ell}\ \ P_{m-\ell}],\ T = [T_{\ell}\ \ T_{m-\ell}] \qquad (4)$$

Où ℓ représente le nombre de composants principaux retenus à conserver dans le modèle PCA.

En tenant compte des premières valeurs propres les plus élevées et de leurs vecteurs propres, la matrice X est décomposée en :

$$\mathrm{X} = T_{\ell}\ P_{\ell}^{T} + \mathrm{E} \qquad (5)$$

Où $T_{\ell} = X\ P_{\ell}$ et E est la matrice résiduelle.

Un exemple de vecteur $(k) \in \mathbb{R}^{m}$ peut être projeté sur le sous-espace principal et le résiduel, respectivement,

$$\hat{x}(k) = P_{\ell} t_{\ell}(k)$$
$$= C_{\ell}\, x(k) \qquad (6)$$

Où $\hat{x}(k)$ est le vecteur d'estimation de $x(k)$, $C_\ell = T_\ell\ P_\ell^T$ et

$$t_\ell\ (k) = P_\ell^T\ x(k) \in \mathbb{R}^\ell \qquad (7)$$

Est le vecteur des premier ℓ scores de variables latentes.

Le vecteur de m − ℓ derniers scores de variables latentes, qui représentent les la projection des données de mesure dans le sous-espace résiduel, est donnée par :

$$t_{\ell-\mathrm{m}}\ (k)\ = P_{\ell-\mathrm{m}}^T\ \ x(k) \in \mathbb{R}^{\ell-\mathrm{m}} \qquad (8)$$

1.2.2 Méthode de l'ACP à noyaux (Karnel PCA) :

En fait, Karnel PCA est Une extension non-linéaire de PCA, son idée principale est de mapper les données dans un espace de fonctionnalités via un mappage non linéaire, puis une ACP linéaire est effectué dans l'espace des fonctionnalités.

Étant donné un ensemble de données d'entraînement normalisées $X = [x_1\ x_2\ \dots\ x_N]\ T \in \mathbb{R}^{\mathrm{N}\times\mathrm{m}}$, où m est le nombre de variables du processus et N est le nombre de des mesures.

Un mappage non linéaire dans l'espace des fonctionnalités $\mathcal{H}$, $x_i \in \mathbb{R}^m \longrightarrow \phi i = \phi(x_i)\ \mathbb{R}^h$ mappe le jeu de données d'entraînement en un espace de fonctionnalités dimensionnelle élevée, ou $h \gg m$ est la dimension dans l'espace des fonctions.

Une importante propriété de l'espace des fonctionnalités est que le produit scalaire de deux vecteurs $(x_i)\ et\ (x_j)$, $i,j = 1, \dots, N$, peut être déterminé comme :

$$(x_i)\ T\ (x_j) = k(x_i\ ,x_j) \qquad (9)$$

Où k est la fonction Kernel. L'une des fonctions du noyau les plus utilisées est la fonction de base radiale (RBF) qui est donnée par :

$$(x_i\ ,\ x_j\ ) = \exp\ \left[\frac{-\|x_i\ ,x_j\ \|^2}{2\sigma^2}\right] \qquad (10)$$

Où σ est la largeur d'une fonction gaussienne qui contrôle la flexibilité du noyau

Comme recommandé dans [20], un choix typique pour σ est la moyenne de distance minimale (d) entre deux points dans le jeu de données d'apprentissage, i.e. $\sigma^2 = c\ \frac{1}{N}\ \sum_{i=1}^{N} min_{\mathrm{j}\neq i}\ d^2(x_i\ ,x_j)$ où c est un paramètre défini par l'utilisateur. En supposant que les vecteurs dans l'espace des fonctionnalités sont mis à l'échelle à une moyenne nulle et variance unitaire, les données mappées sont organisées comme : $\mathcal{X} = [\ \phi(x_1)\ \phi(x_2)\ \dots\ \phi(x_N)]^{\ T}$ .La matrice de covariance $\boldsymbol{C}$ du jeu de données dans l'espace d'entités est définie comme suit :

$$(N-1)\ C = \mathcal{X}^T\mathcal{X}$$

$$= \sum_i^N \phi_i\ \phi_i^T \qquad (11)$$

KPCA dans l'espace des fonctionnalités équivalent à la résolution du vecteur propre suivant équation,

$$\mathcal{X}^T X_{\mathcal{v}} = \sum_{i=1}^{N} \phi_i \ \phi_i^T \boldsymbol{v}$$

$$= \lambda \boldsymbol{v} \tag{12}$$

La fonction de mappage ϕi n'est pas explicitement définie, on peut évaluer le matrice de Gram $\mathcal{X}^T\mathcal{X}$ en utilisant la fonction Kernel $k(x_i , x_j) = \phi(x_i)^T \phi(x_j)$.

Avec l'utilisation de l'astuce du noyau, on peut définir la matrice K avec $k(x_i , x_j)$ éléments,

$$k = \mathcal{X}\mathcal{X}^T = \begin{bmatrix} \phi_1^T\phi_1 & \cdots & \phi_1^T\phi_N \\ \vdots & \ddots & \vdots \\ \phi_N^T\phi_1 & \cdots & \phi_N^T\phi_N \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{k}(x_1, x_1) & \cdots & \mathrm{k}(x_1, x_N) \\ \vdots & \ddots & \vdots \\ \mathrm{k}(x_N, x_1) & \cdots & \mathrm{k}(x_N, x_N) \end{bmatrix} \tag{13}$$

KPCA cherche à résoudre l'équation des vecteurs propres dans l'espace des fonctionnalités. Laissant α être un vecteur propre de la matrice K et λ sa valeur propre correspondante,

$$\boldsymbol{v} = \lambda^{-1}\mathcal{X}^T\alpha \tag{14}$$

La matrice du ℓ chargement principal retenu du KPCA dans l'espace de fonctionnalités est notée : $P = [\ \mathcal{v}_1,\ldots,\mathcal{v}_\ell\ ] \in \mathbb{R}^{\mathrm{N}\times\ell}$, et $N - \ell$ dernier chargement principal par : $\tilde{P} = [\mathcal{v}_{\ell+1},\ldots,\mathcal{v}_{\mathrm{N}}] \in \mathbb{R}^{\mathrm{N}\times(\mathrm{N}-\ell)}$,

$$P = [\frac{1}{\lambda_1}\mathcal{X}^T\ \alpha 1, \ldots, \frac{1}{\lambda_\ell}\mathcal{X}^T\ \alpha_\ell] \tag{15}$$

Le choix du nombre ℓ de PCs a fait l'objet de nombreuses études où [25, 26] décrivent certains d'entre eux.

Pour une mesure donnée x et son vecteur mappé $\phi = (x)$ les scores sont calculés comme,

$$t = P^T\ \phi \in \mathbb{R}^{\ell} \tag{16}$$

$$\tilde{t} = P\ \phi \in \mathbb{R}^{\mathrm{N}-\ell} \tag{17}$$

1.3 Méthode proposé :

La dimension du modèle KPCA dépend du nombre d'échantillons du jeu de données d'entraînement. Réduire le modèle KPCA conduit à réduire ce nombre d'échantillons.

Par conséquent, un nouveau schéma basé sur la réduction de dimension PCA technique est proposée. Il vise à extraire uniquement des observations non corrélées à partir de l'ensemble de données de formation à utiliser pour développer un KPCA approprié au modèle.

Comme présenté dans [2.1], la technique PCA peut être utilisée pour réduire la taille d'un jeu de données. Considérons la matrice de données d'entraînement $X = [x_1, x_2\ \ldots\ x_N]^T \in \mathbb{R}^{\mathrm{N}\times\mathrm{m}}$ est normalisé à une moyenne nulle et une variance

unitaire où $x_k \in \mathbb{R}^m$ est un vecteur échantillonné au temps k de m variables et N représente le nombre d'observations. Pour réduire le nombre d'échantillons N de l'original matrice de données X, L'ACP est effectuée par la décomposition en valeur propre de la matrice de covariance de $Y \in \mathbb{R}^{N \times m}$ tel que $Y = X^T$

$$\Sigma_Y = P_Y \Lambda_Y P_Y^T \in \mathbb{R}^{N \times N} \quad (18)$$

Où $\Lambda_Y = \text{diag}(\lambda_{Y,1}, \lambda_{Y,2}, \dots, \lambda_{Y,N})$ est la matrice des valeurs propres de la diagonale ($\lambda_{Y,1} \geq \lambda_{Y,2} \geq \cdots \geq \lambda_{Y,N}$) et $P = (P_{Y,1}, P_{Y,2}, \dots, P_{Y,m})$ est la matrice des vecteurs propres . ($P_{Y,i}$, $i = 1,2, \dots, N$, Représentent les vecteurs propres normalisés et mutuellement orthogonaux associé aux valeurs propres)

Ensuite, la matrice Y est transformée en variables indépendantes T_Y par :

$$T_Y = YP_Y \quad (19)$$

Où $T_Y = [T_{Y,1}, T_{Y,2}, \dots, T_{Y,N}]$ contient les composantes principales, qui sont orthogonaux les uns aux autres.

En tenant compte des ℓ premières valeurs propres les plus élevées et de leurs vecteurs propres $P_{Y,\ell s}$,les principales composantes de la matrice de données Y, qui représentent les

Échantillons indépendants de la matrice de données originale X sont donné par :

$$T_{Y,\ell s} = Y P_{Y,\ell s} \in \mathbb{R}^{m \times \ell s} \quad (20)$$

Où m est le nombre de variables et ℓs est le nombre de composantes principales de Y ou nombre d'échantillons conservés de la matrice original de données X.

Une fois que le nombre d'échantillons retenus est déterminé et que le la matrice T_Y est calculée, La matrice de données d'entraînement réduite X_r est définie comme suit :

$$X_r = T_{Y,\ell s}^T = [x_1' \; x_2' \; \dots \; x_{\ell s}']^T \in \mathbb{R}^{\ell s \times m} \quad (21)$$

Ces données d'entraînement réduites sont utilisées pour calculer le modèle KPCA tel qu'il est présenté dans la section [2.2], appelé KPCA réduite (RKPCA).

Les tableaux de détection des défauts basés sur la méthode RKPCA sont présentés dans la section [3].

1.4 Détection de défauts par (RKPCA) :

L'utilisation de KPCA pour la détection des erreurs impose un coût de calcul élevé lorsque l'ensemble de données d'entraînement est volumineux car les mesures collectées sont utilisées pour à la fois la modélisation et la détection des défauts. Ainsi, il est important d'utiliser la proposition Modèle RKPCA pour réduire la complexité de calcul et les coûts de stockage le modèle KPCA. Ce modèle est

déterminé sur la base d'une décomposition propre et la sélection du nombre ℓ de composants à conserver.

Dans cette étude, le pourcentage cumulé de variance (PCV) est particulièrement approprié car il peut être facilement redéfini. Le PCV

Mesure la quantité de variation capturée par les premières ℓ variables latentes comme :

$$PCV(\ell) = \frac{\sum_{j=1}^{\ell} \lambda_j}{\sum_{j=1}^{m} \lambda_j} \quad (22)$$

Et ℓ est choisi de telle sorte que le CPV soit supérieur à un seuil donné. Après cela, la détection des erreurs basée sur RKPCA est effectuée à l'aide du système Hotelling T^2, Squared Predictive Error (SPE) ou 𝕼 et φ statistiques combinées. L'indice Hotelling T^2

1.4.1 Statistique T² :

$$t = P_f^T \phi(x) \quad (22)$$

$$t = \Lambda_\ell^{-\frac{1}{2}} \mathrm{P}^{\mathrm{T}} (x) \quad (23)$$

$$T^2 = \hat{\mathrm{t}}^{\mathrm{T}} (k) \Lambda_\ell^{-1} \hat{\mathrm{t}} (k) \quad (24)$$

t : sont les composantes principales de la matrice de Gram.

$$k(x) = [\, k(x_1, x), \ldots, k(x_N, x)]^{\,T} \quad (25)$$

$\Lambda_\ell = \{\lambda_1 , \lambda_2, \ldots , \lambda_\ell \}$: Une matrice diagonale contenant les plus grandes valeurs propres de la matrice de covariance.

Le processus est en fonctionnement anormal (présence d'un défaut), à un instant donné, si :

$$T^2 (k) > \tau^2 = \chi^2 \alpha (\ell) \quad (26)$$

Avec un intervalle de confiance : $(1 - \alpha) \times 100\%$

Où :

$$\chi^2_{\ell,\alpha} \frac{= \ell(N+1)(N-1)}{N(N-\ell)} F_{\ell,(\mathrm{N}-\ell),\alpha} \quad (27)$$

$F_{\ell,(\mathrm{N}-\ell),\alpha}$: est la distribution de Fisher

$\ell, (\mathrm{N} - \ell), \alpha$: Degrés de liberté, et N nombre d'observations.

1.4.2 Statistique SPE :

La technique de surveillance de la KPCA est identique à la procédure utilisée dans l'ACP mais calculer dans l'espace caractéristique, pour la détection de défauts, la statistique SPE est donnée par :

$$Q = \|\|\phi(X) - \phi p\,(X)\| 2 = \|\|\phi N(X) - \phi p\,(X)\| 2$$

$$= \phi_N(X)^T\, \phi_N\,(X) - 2\phi_N(X)^T\, \phi_P\,(X) + \phi_N(X)^T\, \phi_N\,(X)$$

$$=\sum_{j=1}^{N} t_j V_j^T \sum_{k=1}^{N} t_k V_k - 2\sum_{j=1}^{N} t_j V_j^T \sum_{k=1}^{p} t_k V_k + \sum_{j=1}^{p} t_j V_j^T \sum_{k=1}^{p} t_k V_k$$

$$=\sum_{j=1}^{N} t_J^2 - \sum_{j=1}^{P} t_J^2 + \sum_{j=1}^{P} t_J^2$$

$$=\sum_{j=1}^{N} t_J^2 - \sum_{j=1}^{P} t_J^2 \qquad (28)$$

Avec : $t_k = \langle V_k, \tilde{\phi}(X_t)\rangle = \sum_{i=1}^{N} \alpha_i^k \langle \tilde{\phi}(X_i), \tilde{\phi}(X_t)\rangle = \sum_{i=1}^{N} \alpha_i^k \tilde{k}\, t(X_i, X_t)$

Le processus est considéré en fonctionnement anormal (présence du défaut) si :

$SPE(k) > \delta_\alpha^2$

δ_α^2 : est le seuil de détection du SPE avec un seuil de confiance α donné où :

$$\delta_\alpha^2 = \theta_1 \left(\frac{C_\alpha \sqrt{2\theta_2 h_0^2}}{\theta_1} + 1 + \frac{\theta_2 h_2 (h_0 - 1)}{\theta_2^2} \right)^{\frac{1}{h_0}} \qquad (29)$$

$\theta_i = \sum_{j=\ell+1}^{m} \lambda_j^i$ Pour j= 1,2,3.

λ_i : représente la j^{eme} valeur propre de la matrice de covariance Σ

La stratégie de surveillance proposée basée sur le RKPCA qui est réalisée en deux étapes. L'étape on -line où la réduction Le modèle KPCA est calculé et l'étape en ligne où le jeu de données de test est utilisé pour générer des indices de détection de défaut pour la surveillance du processus. Algorithme 1 résume les différentes étapes de la méthode proposée.

• Algorithme 1 :

Training Data

Soit $X \in \mathbb{R}^{N \times m}$ la matrice de données d'entrainement

1- Effectuer l'ACP linéaire sur la matrice Y = X^T et calculer les principales Composantes T_Y=Y P_Y ;

- Calculer la matrice de covariance ;

- Déterminer la décomposition des valeurs propres déterminées de la matrice de covariance.

- Le nombre d'échantillons retenus est déterminé à l'aide du critère CPV; Le nombre ℓs est sélectionné si le CPV est supérieur à 99%.

- Calculer la matrice de données réduites $X_r = T_{Y,\ell s}^T T \in \mathbb{R}^{\ell s \times m}$, - Calculer la moyenne et la variance de l'ensemble de données d'entraînement réduit ; échelle réduite des données ;

2- Appliquer KPCA sur la matrice de données réduiteX_r . - Mappez la matrice de données réduite à l'espace de fonctionnalités ;

- Construire la matrice de noyau réduite et la mettre à l'échelle,

- Résoudre le problème de la décomposition des valeurs propres ;

- Déterminez le nombre de composants principales du noyau conservés à l'aide de Critère CPV;

- Calculer les indices de surveillance T^2, $\mathbb{Q}$ et 'et leurs limites de contrôle à un niveau de confiance donné ;

Testing Data

1- Pour chaque nouvel échantillon de mesure à la fois k, $(x) \in \mathbb{R}^m$, mise à l'échelle avec moyenne et variance des données d'entraînement ;

2- Calcule du vecteur Kernel $k_k = (x_k, x_i)$, $i = 1, \ldots, \ell s \in \mathbb{R}^{\ell s \times 1}$, et le mettre à l'échelle; 3- Calcule des indices de détection de défaut à chaque temps$k$, T_k^2, $\mathbb{Q}_k$ et φ_k. Si un indice dépasse sa limite de contrôle, un défaut est déclaré.

2.5 Conclusion :

Dans ce chapitre, nous avons expliqué que l'ACP est une méthode très intéressante pour la mise en évidence des corrélations linéaires existantes entre les variables du processus.

Il permet de privilégier les directions de l'espace porteur d'un maximum d'informations dans le sens de maximiser la variance de projection, cependant, la plupart des processus industriels non linéaires et dynamiques et donc PCA non réglé pour ce type de système justifie le choix du noyau fonctionnel noyau qui génère des données non linéaires dans un autre espace linéaire.

Le problème que nous avons rencontré était que pour le modèle KPCA, la dimension élevée, cela nous a obligé à développer un modèle KPCA réduit pour après déploiement sur notre carte Raspberry pi .

Chapitre 2

2.1 Introduction :

Apres l'invention de premier transistor dans l'électronique ont a été capable à réduire et reproduire des machines en petit taille avec des performances très élevés comme l'ordinateur jusqu'au temps ou on a vue des micros et des nano ordinateur comme la raspberry pi.

Dans ce chapitre on va discuter un peu sur la raspberry pi avec ces composants et les technologies et le langage de programmation utilisé dans ce projet.

Fig 2.1 – Symbole de la fondation Raspberry pi.

2.2 Raspberry Pi

Est un nano ordinateur mono-carte qui contient un processeur ARM crée par Pr Eben Upton est son équipe dans les laboratoires de recherche de l'université de Cambridge, elle comporte un processeur ARM, GPU, RAM, est d'autres ports comme HDMI, RG45(pas dans tous les modèles) est même des pins ou on accroche des autre composant comme une caméra, des détecteurs …etc.

Raspberry Pi utilise le model SOC ou System on chirp c'est-à-dire toute les composant précédant se trouve sur une seule carte ; ce nano ordinateur support Les system d'exploitation basé sur linux Debian est-elle confortable même avec Windows OS.

2.3 Comparaison entre Raspberry Pi est Les autre circuit :

Le tableau ci dessue représente une comparaison entre les différentes cartes programmables dans le marché :

Nom	Raspberry Pi	BeagleBoard	Arduino
Model	B	X 15	R3
Prix	35$	90-120$	24$
Carte Soc	ARM-cortex A58	ARM-cortex A8	ATMega 328
Vitesse	1.4GHz	1GHz	16-20MHz
RAM	1Gb	512Mb	12Kb
Mémoire	Micro SD	2Gbit + Micro SD	32Kb
Tension	5 v	5 v	5-17V
HDMI	Oui	non	Non
Langage de programmation supporté	Tous les langages qui peuvent compiler en Linux	Tous les langages qui peuvent compiler en Linux	Objective-C Embedded Scratch

Tableau 2-1 : Comparaison entre les différentes cartes programmables dans le marché.

2.4 Les diffèrent modèle de raspberry Pi

Modèle	Date de réalisation
Raspberry Pi 4	**24/06/2019**
Raspberry Pi 3 Model A+	**15/11/2018**
Raspberry Pi 3 B+	**14/03/2018**
Raspberry Pi Zero WH	**12/01/2018**
Raspberry Pi Zero W	**28/02/2017**
Raspberry Pi 3	**26/02/2016**
Raspberry Pi Zero	**30/11/2015**
Raspberry Pi 2	**1/02/2015**
Raspberry Pi A+	**10/11/2014**
Raspberry Pi B	**15/02/2012**

Tableau 2-2 : différent modèle et la date de réalisation de raspberry pi.

2.5 Architecture du Raspberry pi 4 : (Notre model)

- La carte Raspberry pi 4 (Model B) se constitue de :

• Broadcom BCM2711, Cortex-A72 Quad core (ARM v8) SoC 64 bits 1.5GHz

• *1 go, 2 go ou 4 go LPDDR4-2400 SDRAM (selon modèle)

• 2.4 GHz et 5.0 GHz IEEE 802.11ac sans fil, Bluetooth 5.0, BLE

• Gigabit Ethernet

• 2 ports USB 3.0; 2 ports USB 2.0.

• Tête GPIO standard 40 broches framboise Pi (entièrement rétrocompatible avec les cartes précédentes)

• 2 ports micro-hdmi (jusqu'à 4kp60 pris en charge)

• Port d'affichage MIPI DSI à 2 voies

• Port de caméra MIPI CSI à 2 voies

• Port audio stéréo et vidéo composite 4 pôles

• H.265 (4kp60 décoder), H264 (1080p60 décoder, 1080p30 coder)

• OpenGL ES 3.0 graphique

• Fente pour carte micro-sd pour charger le système d'exploitation et le stockage de données • 5V cc via connecteur USB-C (minimum 3A *)

• 5V cc via l'en-tête GPIO (minimum 3A *)

• Alimentation sur Ethernet (PoE) activée (nécessite un chapeau PoE séparé)

• Température de fonctionnement : 0 - 50 degrés C ambiant

2.6 Domaine d'utilisation :

La Raspberry Pi a d'abord été créé à vocation éducative et de ce point de vue, il remplit parfaitement son rôle. On peut y installer plusieurs distributions de Linux et de là, il est possible d'installer quantité de logiciels, notamment des environnements de programmation. On peut tout à fait imaginer qu'un professeur d'école décide d'équiper sa salle en Raspberry Pi afin d'éviter l'achat de postes informatiques (qui demande un budget plus conséquent).

En collège et notamment en technologie, le Raspberry permet de faire acquérir des compétences en programmation sans dédier forcément un poste informatique avec des logiciels coûteux. Gain considérable donc au niveau du budget.

Toutefois, le Raspberry-Pi permet de réaliser de nombreux projets assez simples à mettre en œuvre. Citons par exemple quelques idées assez courantes comme créer une console de retro-gaming, un media center, un serveur web, VPN, email, ou NAS, créer un système de vidéosurveillance, une station météo, un système de gestion d'aquariums, ou encore gérer la domotique chez nous ! On peut également se servir de notre Raspberry comme d'un ordinateur de bureau afin de naviguer sur le web, écouter de la musique, lire vos mails, etc.

En plus, il peut capter de l'internet ceux qui veut dire qu'on peut bien le contrôler à distance dans le cas de son utilisation pour le contrôle d'une machine :

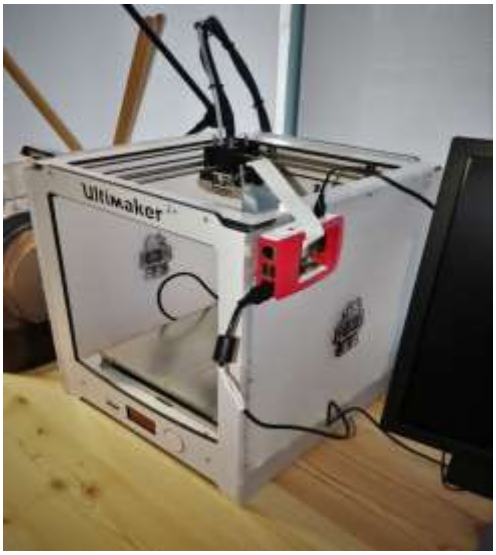


Fig 2.2 Monitorage de l'imprimante 3D avec Raspberry-PI.

2.6.1 System d'opération :

La carte Raspberry-PI nécessite un system d'exploitation pour sa gestion et qui permet d'accéder est gérer le côté matériel et les entrées sorties. Pour cela ce système est à base de linux qui s'appelle Raspbian

Ce qui caractérise ce système c'est la capacité de gérer le matériel raccordé à la carte d'une manière facile et il est open source.

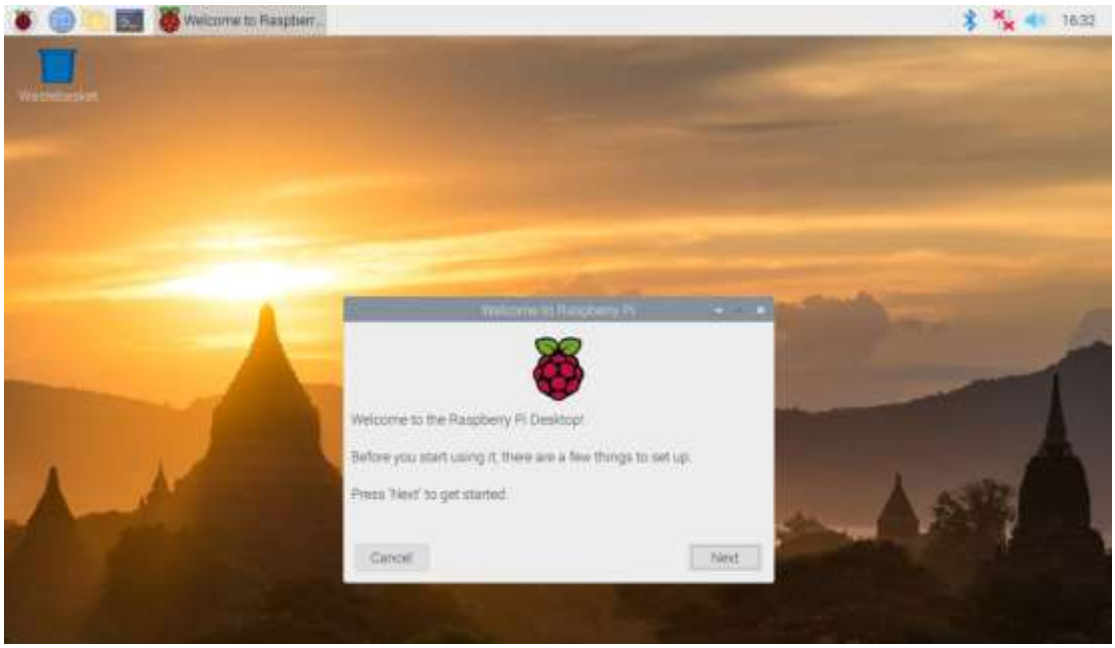


Figure 2-3: Bureau de Raspbian OS.

2.6.2 Préparation de Raspbian :

Après la préparation de tout le matériel présenté précédemment on peut passer à l'installation du system d'exploitation. Pour se faire, il faut suivre les étapes :

2.6.3 Téléchargement de system :

On accède à ce lien d'où nous téléchargeons la dernière version de Raspbian la version NOOB est la plus complète.

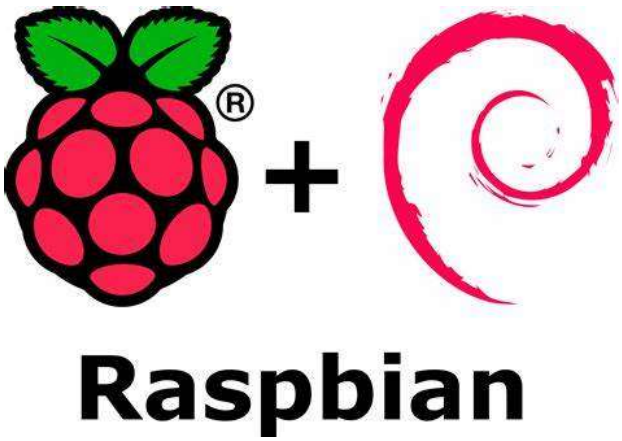


Figure 2-4 : capture d'écran prie de site lien précédent.

2.6.4 Graver l'OS sous la carte SD :

Le cas où nous voulons Installer une version différente de NOOB il faut graver l'OS sur la carte SD d'une manière spéciale. Pour cela nous insérons la carte SD dans le lecteur de carte du pc et nous utilisons un programme spécial comme Win32Image ou Etcher. On clique sur le bouton choisir une image est on choisit la version que nous avons téléchargé sur le pc.

Figure 2-5 : Flasher la carte SD avec Raspbian.

Quand cette opération est terminée nous insérons la carte SD sur la carte Raspberry Pi et nous connectons les autres périphériques (souris et clavier) et le câble HDMI enfin nous terminons par l'alimentation.

2.6.5 Installation du system :

Apres l'alimentation de la carte raspberry le system termine automatiquement la procédure de l'installation puis il affiche l'image du Bureau avec une petite plaque d'où on configure la langue d'affichage et le nom d'utilisateur.

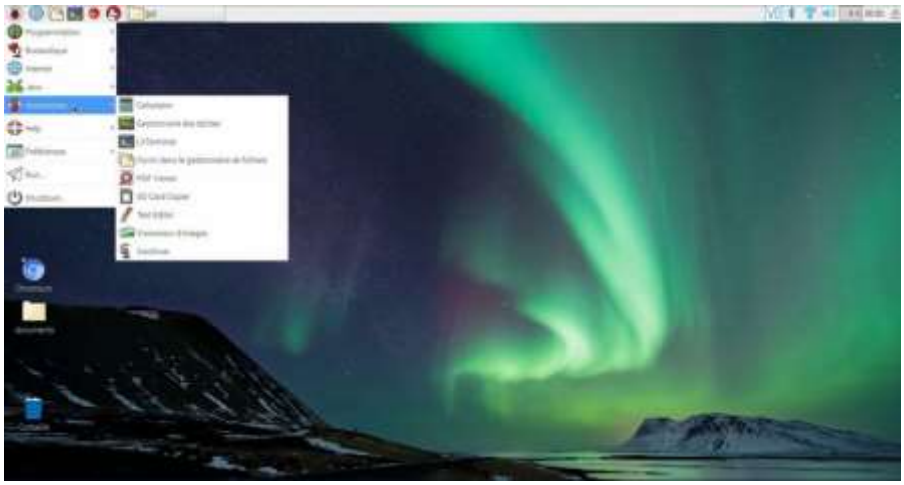


Figure 2-6: Bureau de Raspbian OS après l'installation.

Raspbian desktop contient :

➢ Bar des taches comme dans Windows

➢ Un buton rassemble « démarrer » dans Windows qui contient une liste des listes des applications qui sont :

o List de programmation : qui contient des IDE pour aide à programmer dans plusieurs lagunages comme PYTHON, C/C++, JAVA.

o List d'office : qui contient des logiciels dédiés au traitement des donné de type texte.

o Liste des accessoires : d'où on Install les logiciels d'une manière graphique On trouve plusieurs listes comme des liste des jeux, vidéo et son et même une liste pour les navigateurs en internet

➢ Elle a qu'un seul desktop (pas comme les autres versions de linux qui ont plusieurs).

2.6.6 Configuration :

C'est-à-dire l'activation ou la permission de gérer les périphériques est le port GPIO connecter avec la carte avec des programmes ou avec des codes implémenter à travers le terminal de Raspbian.

2.7 Python :

Le Langage Python est utilisé dans notre projet par sa richesse en Library et sa souplesse de programmation. Il a été développé par Guido van Rossum au centre de recherche informatique et mathématique d'Amsterdam, Pays-Bas.

La première version de ce langage est apparue le 19 février 1991 qui peut être utilisé sur les différents OS comme Windows, Linux …etc.et il est possible de traduire le code de python dans plusieurs langages de programmation comme JAVA et C#.

La communité de ce langage et énorme due aux facilitées et la mobilité de ce langage.il est considéré comme le langage de programmation plus proche au langage humain. En 2018 Python a été considérer comme le langage de science due à ces Library qui permettent d'implémenter Python dans tous les domaines de science. Nous pouvons trouver python dans tous les domaines comme les logiciels les applications Android, développement web …etc.

Figure 2-7 : Symbole de Python.

2.7.1 Philosophie de python :

Après la création de python, Tim Peters qui un des développeurs de python à poster sur son compte Gmail ce que on appelle le « ZEN de python » ou la philosophie de python qui et un ensemble de lois utilisées pendant la création de python l'une de ces règles sont :

- Beau vaut mieux que moche.
- Explicite est meilleur qu'implicite.
- Simple c'est mieux que complexe.
- Les erreurs ne doivent jamais passer silencieusement.
- Si la mise en œuvre est difficile à expliquer, c'est une mauvaise idée.

Et autre règle qui détermine pour quoi python et un langage de programmation simple.

2.7.2 Pour Quoi Python :

Dans ce projet nous avons implémenté python car il est facile, ces codes ne sont pas très longs ou compliquer, il supporte les nouveaux domaines de recherche comme l'intelligence artificiel. La richesse en Library, et la culpabilité d'exécuter les codes python sur les processeurs ARM.

2.7.3 Points forts du python :

- langage de très haut niveau ;
- sa lisibilité ;

• « Langage algorithmique exécutable ».

2.7.4 Conclusion :

Le Raspberry Pi avec sa taille et son coût réduits ouvre des perspectives immenses pour la réalisation d'applications. Les quelques exemples présentés dans ce chapitre sont autant de points de départ pour vos propres réalisations.

le choix de Python et du a sa simplicité d'implémentation sur une carte raspberry pi et que nous pouvons l'utiliser dans presque tous les domaines, due à sa richesse en librairies.

Chapitre 3

3.1 Introduction du Processus TE :

Le simulateur du processus chimique Tennesse Estman Process (TE), est considéré comme une installation pilote de l'industrie chimique conçue par Eastman company. Il est largement utilisé par la communauté scientifique pour évaluer les performances des algorithmes de commande et de diagnostic. Le TE est un réacteur chimique de grande dimension.

Ce processus fournit les produits chimiques finis **G** et **H** à partir de quatre réactifs **A, C, D** et **E**. L'installation possède plusieurs modes de fonctionnement opératoires, 41 variables mesurées (capteurs) et 12 variables manipulées (commandes). De plus, il existe 21 perturbations IDV1 à IDV21qui peuvent être simulées pour perturber le fonctionnement du système. Ce processus représente un défi pour la communauté scientifique, sur l'identification, la commande et la surveillance des processus industriels. Un diagramme simplifié du processus est montré sur la Figure 4.1.

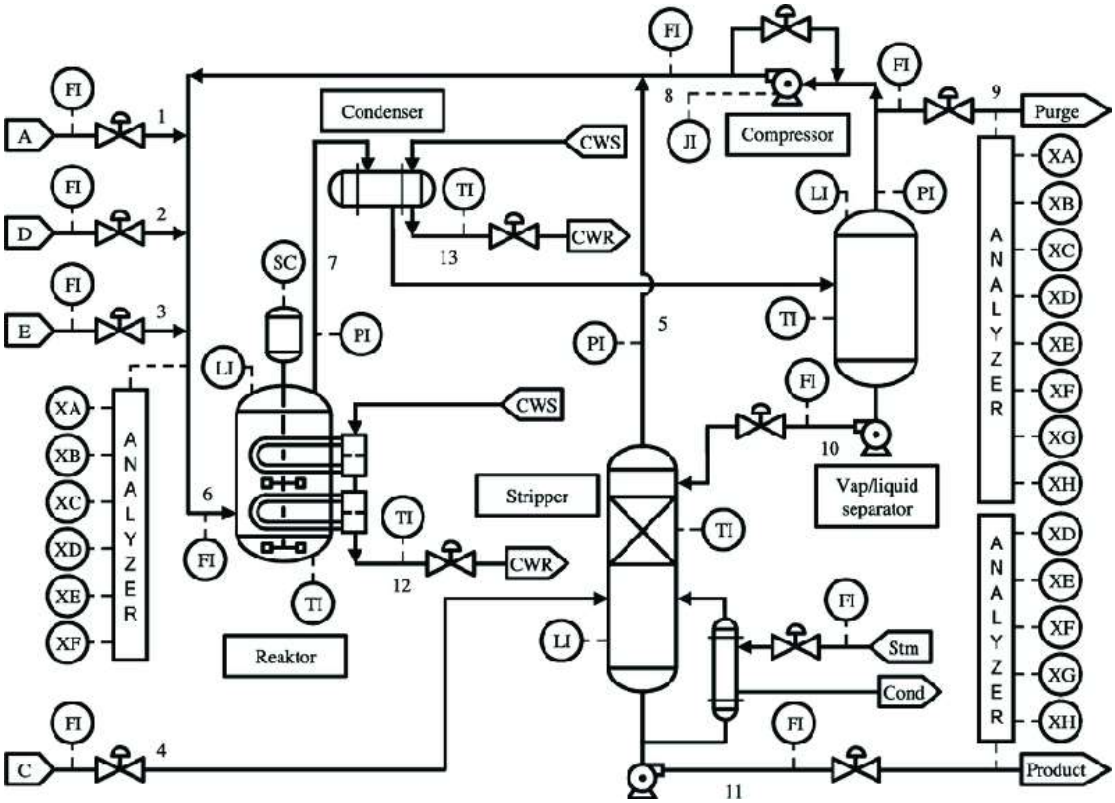


Fig 3.1 - Diagramme simplifié du processus de Tennessee Eastman.

Nous allons présenter l'application de l'ACP et KACP à la détection de défauts de capteurs du processus TE. Les deux tableaux suivants décrivent les variables manipulés et les variables mesurées du processus (manipulated variables, measured variables) :

Variables	Description
XMV(1)	D Feed flow (Stream 2)
XMV(2)	E Feed flow (Stream 3)
XMV(3)	A Feed Flow (Stream 1)
XMV(4)	Total feed flow (Stream 4)
XMV(5)	Compressor recycle valve
XMV(6)	Purge Valve (Stream 9)
XMV(7)	Separator pot liquid flow (Stream
XMV(8)	10) Stripper liquid product flow
XMV(9)	(Stream 11) Stripper stream Valve
XMV(10)	Reactor cooling water flow
XMV(11)	Condenser cooling water flow
XMV(12)	Agitator speed

Tableau 3.1 - Description des variables manipulées du processus.

Variables	Description	Variables	Description
Xmeas(1)	A Feed (stream 1)	Xmeas (22)	Separator cooling water outlet temp
Xmeas(2)	D Feed (stream 2)	Xmeas (23)	Composition of A in reactor Feed
Xmeas(3)	E Feed (stream 3)	Xmeas (24)	Composition of B in reactor feed
Xmeas(4)	Total feed flow (stream 4)	Xmeas (25)	Composition of C in reactor feed
Xmeas(5)	Recycle flow (stream 8)	Xmeas (26)	Composition of D in reactor feed
Xmeas(6)	Reactorfeed rate (stream 6)	Xmeas (27)	Composition of E in reactor feed
Xmeas(7)	Reactor pressure	Xmeas(28)	Composition of F in reactor feed
Xmeas(8)	Reactorlevel	Xmeas(29)	Composition of A in purge gas flow
Xmeas (9)	Reactortemperature	Xmeas (30)	Composition of B in purge gas flow
Xmeas (10)	Purge rate (stream 9)	Xmeas (31)	Composition of C in purge gas flow
Xmeas (11)	Product sep. temp.	Xmeas (32)	Composition of D in purge gas flow
Xmeas (12)	Product sep. level	Xmeas (33)	Composition of E in purge gas flow
Xmeas (13)	Product sep. Pressure	Xmeas (34)	Composition of F in purge gas flow
Xmeas (14)	Prod. Sep. Underflow (stream 10)	Xmeas (35)	Composition of G in purge gas flow
Xmeas (15)	Stripper level	Xmeas (36)	Composition of H in purge gas flow
Xmeas (16)	Stripper pressure	Xmeas (37)	Composition of D in productflow
Xmeas (17)	Stripper underflow (stream 11)	Xmeas (38)	Composition of E in productflow
XMeas (18)	Stripper Temperature	Xmeas (39)	Composition of F in productflow
Xmeas (19)	Stripper stream flow	Xmeas (40)	Composition of G in productflow
Xmeas (20)	Compressorwork	Xmeas (41)	Composition of H in productflow
Xmeas (21)	Reactor cooling water outlet temp		

Tableau 3.2 - Description des variables mesurées du processus.

Le tableau suivant donne la description des 21 Défauts du processus TE :

Fault	Description	Type
1	A/C feed ration, B composition (stream 4)	Step
2	B composition, A/C ration constant (stream 4)	Step
3	D feedtemperature (stream 2)	Step
4	Reactorcooling water inlet temperature	Step
5	Condenser cooling water inlet temperature	Step
6	Feedloss (stream 1)	Step
7	C header pressure loss, reduced availability (stream 4)	Step
8	A,B,C feed composition (stream 4)	Random variation
9	D feed temperature (stream 2)	Random variation
10	C feedtemperature (stream 4)	Random variation
11	Reactor cooling water inlet temperature	Random variation
12	Condenser cooling water inlet temperature	Random variation
13	Reactionkinetics	Slow drift
14	Reactioncooling water valve	Sticking
15	Condenser cooling water valve	sticking
16	Unknown	Unknown
17	Unknown	Unknown
18	Unknown	Unknown
19	Unknown	Unknown
20	Unknown	Unknown
21	Valve position constant (stream 4)	Constant position

Tableau 3.3 Description des défauts du TE processus.

3.2 Résultat de la simulation :

3.2.1 Sous Matlab :

D'abord sur Matlab on a commencé à préciser notre choix de perturbation, ensuite on a lu les deux matrices (xtr : matrice des données du système en bon fonctionnement) et (xtst : matrice défectueuse).

Puis avec la fonction : « Datapreprocess » on a pu faire la moyennes et l'écart type des données pour le centrage.

Après cela on a calculé la matrice de covariance des données d'entrainement pour avoir nos valeurs et des vecteurs propres.

Les valeurs propres vont être classés d'une façon décroissante puis avec la fonction « MyCPV » .Nous allons calculer le nombre optimal de composantes principales retenues qui vont constituer la nouvelle matrice seine réduite.

```
1
2 -    fault = 'IDV_1';
3 -    [xtr,xtst] = Data_Gen(fault);
4 -    [xtr,xtst] = datapreprocess(xtr,xtst);
5 -    [N1,m1]=size(xtr);
6 -    [N2,m2]=size(xtst);
7 -    X1=xtr;
8 -    [N,m]=size(X1);
9 -    Cov=X1*X1';
10 -   [vp0,d0]=eig(Cov/(m-1));
11 -   d0=diag(d0);
12 -   [a0,t0]=sort(d0);
13 -   t0=t0(N:-1:1);
14 -   vp0=vp0(:,t0);
15 -   d1=d0(t0);
16 -   L=MyCPV_col(d1,.99);
17     %L=41;
18 -   newdata=X1'*vp0(:,1:L);
19 -   X1=newdata';
20 -   xxtrain=X1;
21 -   xxtest=xtst;
22     %function prédéfinie utilisé pour
23 -   a=0.95; % le niveau de confiance
24 -   [npc,T2,SPE,clT2,clSPE,phi,clphi,T20,SPE0,phi0,P,d]=KPCA_model_red(xxtrain,xxtest,a);
```

Fig 3.2 – Programme Matlab.

- Appliquer KPCA sur la matrice de données réduite.

- Mapper la matrice de données réduite à l'espace de fonctionnalités ;

- Construire la matrice de noyau réduite et la mettre à l'échelle,

- Résoudre le problème de la décomposition des valeurs propres ;

- Déterminer le nombre de composants principaux du noyau conservés à l'aide de Critère CPV ; - Calculer les indices de surveillance T^2, 𝕼 et 'et leurs limites de contrôle à un niveau de confiance donné ;

• Et ensuite :

- Pour chaque nouvel échantillon de mesure à la fois k, $(x) \in \mathbb{R}^m$, mise à l'échelle avec moyenne et variance des données d'entraînement ;

- Calcule du vecteur Kernel $k_k = (x_k, x_i)$, $i = 1, \ldots, \ell s \in \mathbb{R}^{\ell s \times 1}$, et le mettre à l'échelle;

- Calculer les indices de détection de défaut à chaque temps , $T^2{}_k$, $\mathbb{Q}_k$ et φ_k. Si un indice dépasse sa limite de contrôle, un défaut est déclaré.

• **Résultat :**

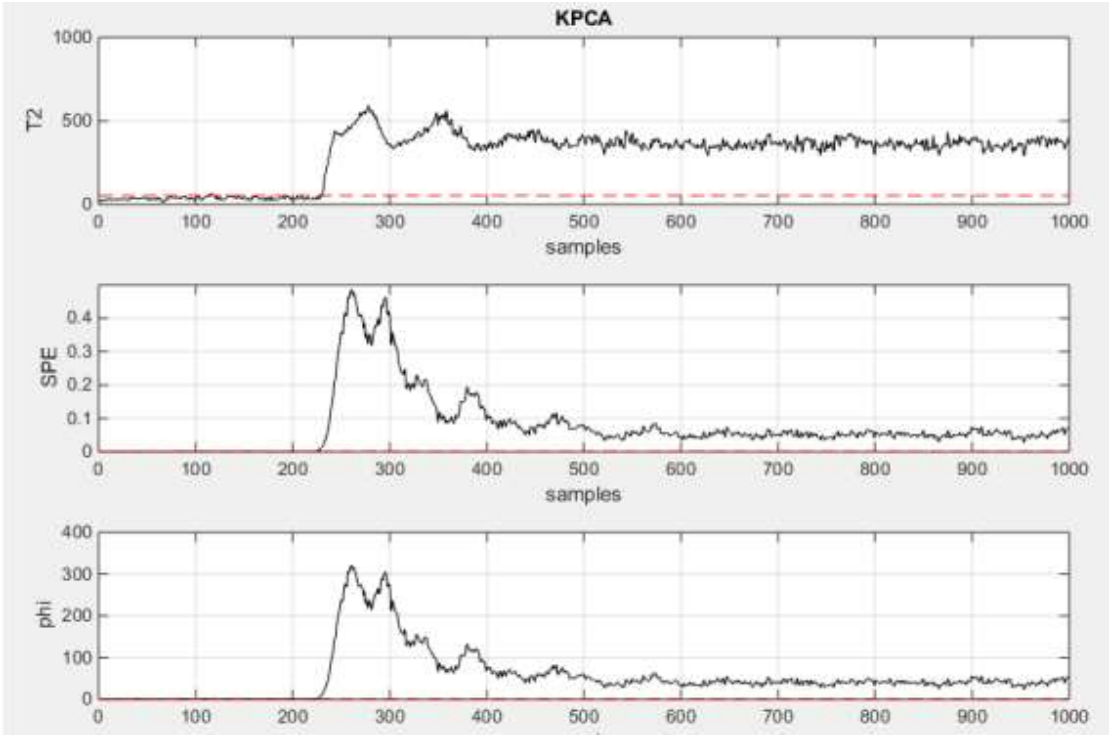


Fig 3.3 – modèle KPCA.

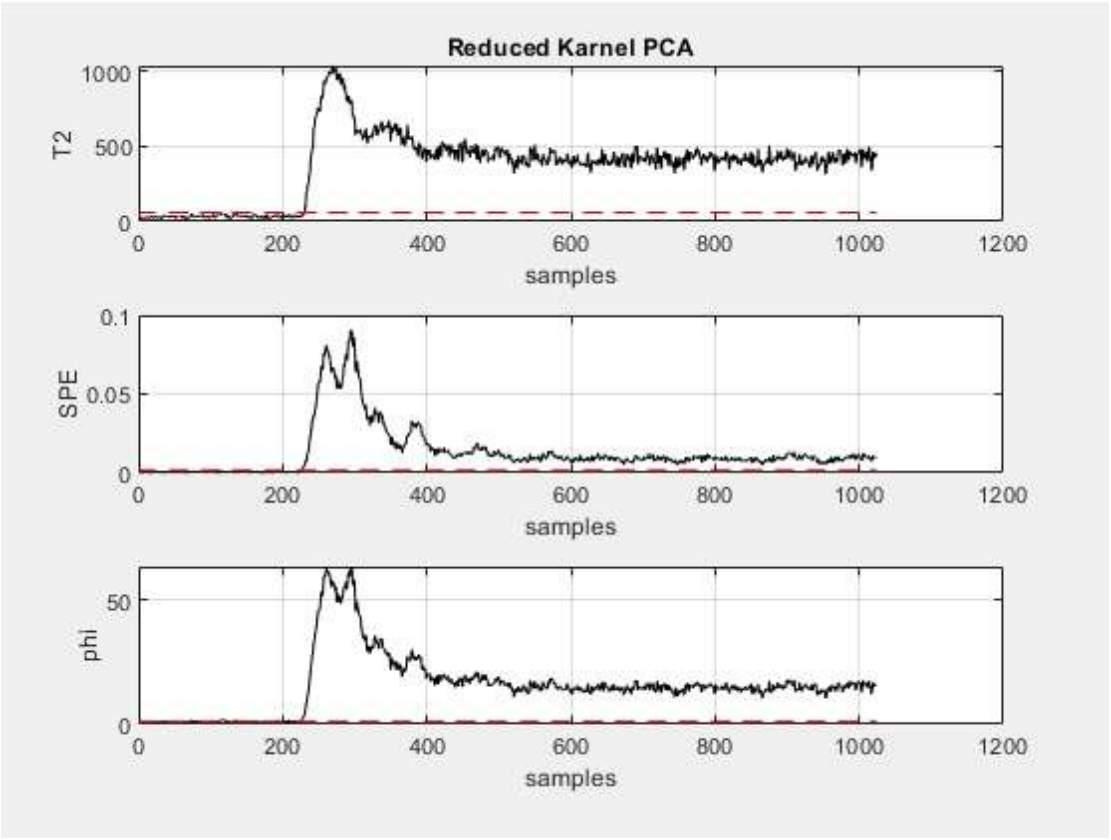
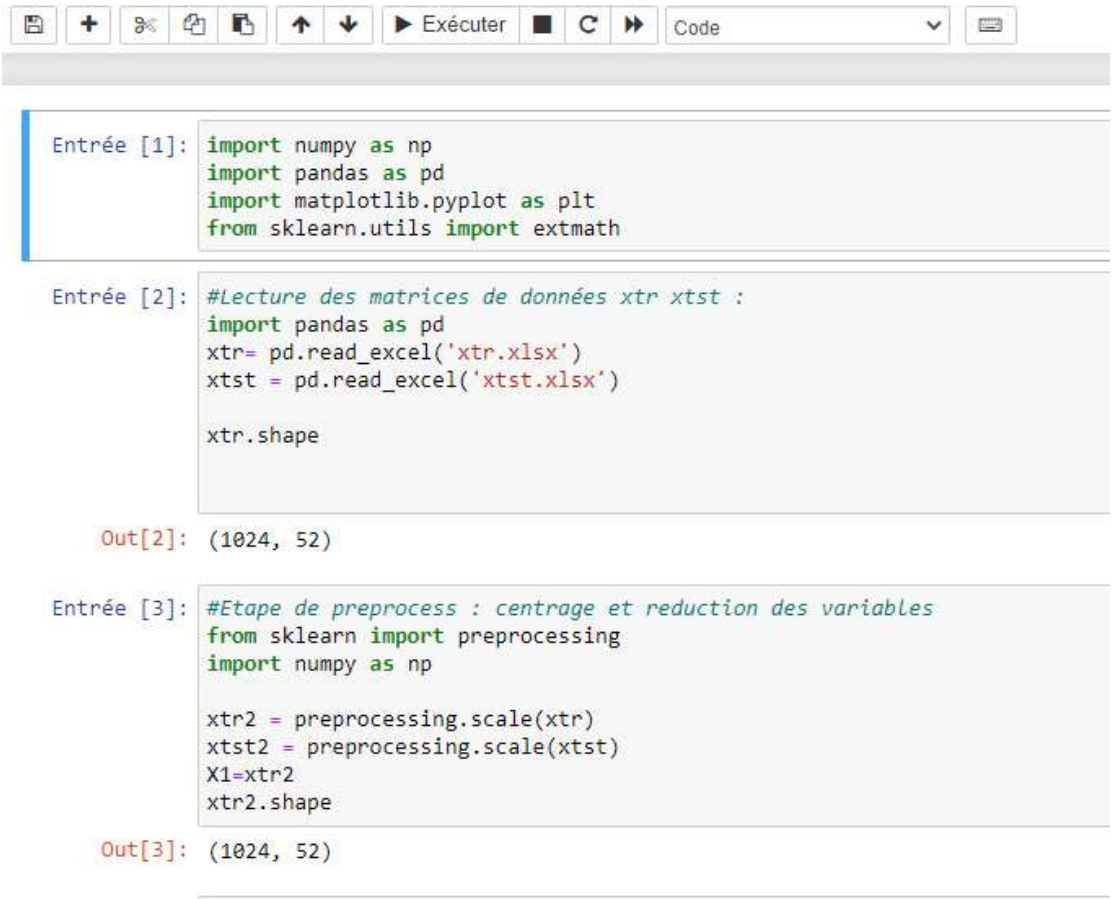


Fig 3.4 – Modèle RKPCA

3.2.2 Sous Python :

- Apres téléchargement du Python 3.8 et (Miniconda pour le fonctionnement de jupyter notebook afin d'avoir la possibilité d'obtenir un code python exécutable)



Entrée [4]:
```
#Calcul de la matrice de Covariance :
xtr2t=xtr2.transpose()

conv = np.dot(xtr2,xtr2t)
conv.shape
```

Out[4]: (1024, 1024)

Entrée [5]:
```
#Calcul des vecteurs propres + Valeurs propres par ordre decroissant
from numpy import linalg as LA

values, vectors = np.linalg.eig(conv/51)
```

Entrée [6]:
```
#Calcul du nombre de composante principales retenus (Fonction PCV)
alpha = [0]*1024
print(len(values))

for i in range(1, len(values)):
    alpha[i]=sum(values[0:i])/sum(values)
    if(alpha[i]>=0.99):
        L=i
        break
L
```

```
1024
```

Out[6]: 41

Entrée [4]:
```
#Calcul de la matrice de Covariance :
xtr2t=xtr2.transpose()

conv = np.dot(xtr2,xtr2t)
conv.shape
```

Out[4]: (1024, 1024)

Entrée [5]:
```
#Calcul des vecteurs propres + Valeurs propres par ordre decroissant
from numpy import linalg as LA

values, vectors = np.linalg.eig(conv/51)
```

Entrée [6]:
```
#Calcul du nombre de composante principales retenus (Fonction PCV)
alpha = [0]*1024
print(len(values))

for i in range(1, len(values)):
    alpha[i]=sum(values[0:i])/sum(values)
    if(alpha[i]>=0.99):
        L=i
        break
L
```

```
1024
```

Out[6]: 41

Entrée [7]:
```
#la matrice de données reduite seine
newdata = np.dot(X1.transpose(),vectors[:,0:L])

X1=newdata.transpose()
xxtrain=X1
xxtest=xtst2
a=0.95
xxtrain.shape
```

Out[7]: (41, 52)

Entrée [8]:
```
#Reduced Kenrnel Matrix
import numpy.matlib
Nx = len(xxtrain)
Mx = len(xxtrain[0])
mean_X=xxtrain.mean(0)
std_X=np.std(xxtrain)
column=np.ones((Nx,Mx))

xxtrain=xxtrain-mean_X*column

xxtrain=xxtrain.real
nzstd= np.nonzero(std_X)
xxtrain =  xxtrain/std_X*column

# distance matrix matLab code
#function D=distanceMatrix(X)

N=len(xxtrain)
XX= np.dot(xxtrain.transpose(),xxtrain)

XX=XX.sum(axis=1)
XXT=XX.transpose()
XX1=np.kron(np.ones((N,1)),XX)
XX2=np.kron(np.ones((N,1)),XXT)

D=XX1+XX2 - 2*(xxtrain* xxtrain)

D[D<0] = 0

DIST=np.sqrt(D)
```

Entrée [9]:
```
from numpy import linalg as LA
import math
from numpy import inf,nan

#%%%%%%%%%%%%%%% Computing parameter c %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
DIaST=DIST.min(0)
c=DIaST.mean()
c=c*c

#%%%%%%%%%%%%%%% Kernel Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

N=len(xxtrain)

K=np.zeros((Nx,Nx), dtype=float)

for i in range(0, Nx-1):
    for j in range(0, Nx-1):
         kint=LA.norm(xxtrain[i,:]-xxtrain[j,:])**2/c
K[i,j]=math.exp(-kint)

K[K == -inf] = 0
K[K == nan] = 0
K[K == inf] = 0


K.shape
```

Out[9]:
```
(41, 41)
```

Entrée [11]:
```
#%%%%%%%%%%%%%%%% Compute Gram matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Summationkx=K.sum(axis=1)
Summationkx1=Summationkx.transpose()
Summationkx2=Summationkx1.sum()

n1=np.ones((Nx,Nx))
N2=1/N
N1=  [element * N2 for element in n1]

Kp=N1*K*N1
Kp=Kp-K*N1-N1*K+K

Kp
```

Out[11]:
```
array([[0.       , 0.       , 0.       , ..., 0.       , 0.       ,
        0.       ],
       [0.       , 0.       , 0.       , ..., 0.       , 0.       ,
        0.       ],
       [0.       , 0.       , 0.       , ..., 0.       , 0.       ,
        0.       ],
       ...,
       [0.       , 0.       , 0.       , ..., 0.       , 0.       ,
        0.       ],
       [0.       , 0.       , 0.       , ..., 0.       , 0.9518144,
        0.       ],
       [0.       , 0.       , 0.       , ..., 0.       , 0.       ,
        0.       ]])
```

Entrée [12]:
```
 import pandas as pd
import numpy as np

 #%%%%%% Calculating number of retained PC%%%%%%%%%%%%%%%%%%%%%%%%

values1, vectors1 = np.linalg.eig(Kp/N)
d=np.diag(vectors1)
ab=np.sort(d)

t11=np.arange(1, 42, 1)
t11=sorted(t11, reverse=True)
d=d[t11[1:]]
lamda=d
s=np.diag(d)
p=vectors1
npc=1
while ((lamda[:npc].sum())/(lamda.sum()) ) < 0.95 : npc=npc+1

npc
```

Out[12]: 38

Entrée [13]:
```
#%%%%%%%%%%%%%% Calcutaing T2 and SPE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

P=vectors1
lamda=np.diag(lamda)
for i in range(1, npc): P[:,i]=P[:,i]/(LA.norm(P[:,i])*np.sqrt(N*s[i,i]))

t1=np.zeros((Nx,npc), dtype=float)
tt1=np.zeros((Nx,npc), dtype=float)
t=np.zeros(npc, dtype=float)

for i in range(1, Nx):
    for j in range (1,npc):
            t[j]=np.dot(P[:,j],Kp[:,i])
            t1[i,:]=t

t1.shape
```

Out[13]: (41, 38)

Entrée [14]:
```
T20=np.zeros((Nx,npc), dtype=float)
SPE0=np.zeros((Nx,npc), dtype=float)
#Calcul des indices d'erreurs T² et SPE :
for i in range(1, Nx):

    T20[i,:]=np.dot(t1[i,:],(1/(lamda[0:npc,0:npc]+0.1)),np.transpose(t1[i,:]))
    SPE0[i,:]=1-t1[i,:]*np.transpose(t1[i,:])-2/(N*Summationkx1[i]+0.01)+1/(N*N)*Summationkx2
npc
```

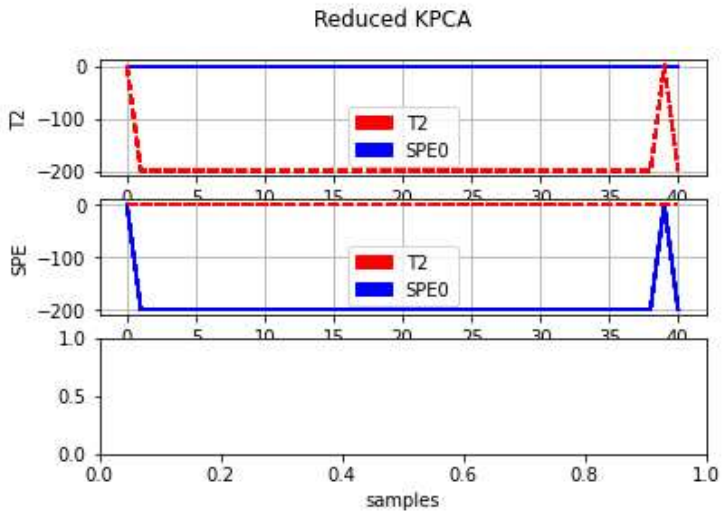
Out[14]: 38

Entrée [15]:
```
clT2=np.percentile(T20,95)
S0=(np.std(SPE0))**2
miu=np.mean(SPE0)
P1=np.round(2*miu**2/S0)
clSPE=np.percentile(SPE0,95)
```

Entrée [16]:
```
fig, axs = plt.subplots(3,1)
fig.suptitle('Reduced KPCA')

axs[0]. plot(T20 , 'b')
# plt.hold(True)
axs[0]. plot(SPE0, 'r--')
# axs[0]. plot(clT2(np.ones(1,Ny)) ,'r--')
axs[0].grid()
axs[0].set( ylabel='T2' )

import matplotlib.patches as mpatches
pop_a = mpatches.Patch(color='red', label='T2')
pop_b = mpatches.Patch(color='blue', label='SPE0')
axs[0].legend(handles=[pop_a,pop_b])
```

```
axs[1].plot(SPE0 ,'b')
axs[1]. plot(T20 ,'r--')
# axs[1]. plot(clSPE(np.ones(1,Ny)) ,'r--')
axs[1].grid()
axs[1].set( ylabel='SPE' )
pop_a = mpatches.Patch(color='red', label='T2')
pop_b = mpatches.Patch(color='blue', label='SPE0')
axs[1].legend(handles=[pop_a,pop_b])

plt.xlabel('samples')
```

Out[16]: Text(0.5, 0, 'samples')



3.3 Conclusion générale :

La complexité grandissante des processus industriels et des pièces fabriquées, les exigences croissantes en termes de sûreté de fonctionnement ainsi que la volonté d'optimisation de la durée de vie des pièces conduisent à mettre en place des contrôles de qualité de plus en plus poussés.

Dans ce travail nous avons présenté une implémentation sur carte Respberry-PI des algorithmes de diagnostic basées sur les méthodes statistiques multivariées à savoir le modèle Kernel PCA réduit.

Dans un premier temps l'algorithme est développé sous Matlab pour les différents tests puis développé sous Python.

L'implémentation sur carte est le développement sous Python a pu être réalisé.

Bibliographie :

1- Harkat M.F., Mourot G., Ragot J. (2000). Sensor failure detection of air quality monitoring network. IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS'2000, Budapest, Hungary, June 14-16.
2- R. J. PATTON et J. CHEN, «Observer-based fault detection and isolation: Robustness and applied», Control Engineering Practice, Vol. 5 (5), pp. 671-682,1997.
3- I. Jolliffe, Principal Component Analysis, 2nd edition éd., SpringerVerlag, 2002.
4- Bergoug Aicha, mémoire (2017), détection et localisation de défaut par la méthode d'ACP.
5- Chakour C., Harkat M-F., and Djeghaba M. (2013). Adaptive kernel principal component analysis for nonlinear dynamic process monitoring. In Control Conference (ASCC), 2013 9th Asian (pp. 1-6).
6- Thèse Doctorat, Nati Slimani Boukhalfa, Synthése d'observateur non linéaires : application au daignostic de défauts, Université Mouloud Mammeri de Tizi-ouzou.
7- L'Analyse en composantes principales (A.C.P.) Pierre-Louis GONZALEZ, math cnam.
8- Thèse, Harkat Mohamed-Faouzi, Détection et Localisation de Défauts par Analyse en Composantes Principales, Centre de Recherche en Automatique de Nancy, 2003.
9- DjoudiChemse-Eddine, mémoire (2015), Modélisation et diagnostic des systèmes non linéaire par ACP à noyaux.
10- Méthodes à noyaux Cours Master 2005/06 Jean-Philippe Vert JeanPhilippe.Vert@mines.org.
11- Mohamed-Faouzi Harkat. Détection et localisation de défauts par analyse en composantes principales. Automatique / Robotique. Institut National Polytechnique de Lorraine - INPL, 2003. Français.
12- DREIBINE Mahieddine, thèse (2014), Diagnostic des systèmes par Analyse des Composants Principales.
13- RACHEDI Temer Abdelatif, thèse (2019), Diagnostic des systèmes non linéaires par les méthodes statistiques à base de fonctions à noyaux.
14- Pascal Pujades - Octobre 2015, 1er pas sur Raspberry
15- Matthieu Legouge, 15 février 2019, Qu'est-ce qu'un Raspberry Pi ?