UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À

L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN ÉLECTRONIQUE INDUSTRIELLE

PAR

LOUIS LEMIRE

ÉTALONNAGE STATIQUE D'UN SYSTÈME DE MESURE À FIBRE OPTIQUE

DÉCEMBRE 1994

"À MES PARENTS ET À MON FRÈRE"

RÉSUMÉ

Dans un processus de mesure, la reconstruction d'une mesurande consiste en une série de transformations du résultat de conversion du signal de mesure (résultat brut de mesure) en résultat final de mesure (conversion N/N). Dans des cas de mesure plus sophistiqués, les méthodes numériques de la reconstitution sont utilisées par exemple, pour l'amélioration de la qualité des données spectrométriques, la correction dynamique des capteurs, l'interprétation des données séismiques et l'égalisation dans des canaux de télécommunications. L'élément clef dans la reconstitution est le modèle mathématique de la relation entre des signaux x et y. La reconstitution est en général, basé sur le modèle mathématique de la relation entre le résultat brut de mesure et la mesurande. Ce modèle est identifié et estimé pendant le processus d'étalonnage.

Ce projet de recherche est dédié à l'application des fonctions spline à l'étalonnage d'un système de mesure électrique de la pression, utilisant les nouveaux capteurs polarimétriques à fibres optique. Le cas étudié représente un cas statique fortement non-linéaire.

La première partie de ce mémoire est consacrée à la description du système de mesure étudié, utilisant trois capteurs à fibre optique dont les signaux de sortie convertis en tension électrique, constituent la base numérique pour la reconstitution de la pression mesurée.

Par la suite, nous présenterons un résumé de différentes méthodes numériques permettant l'étalonnage d'un système de mesure dans un cas statique et non-linéaire.

Les développements des différents algorithmes basés sur les fonctions spline pour l'étalonnage statique du système et leur étude, constituent la partie principale du travail. La validation à l'aide des données synthétiques, a permis de déterminer l'algorithme le plus performant. Par la suite, cet algorithme fut testé avec des données expérimentales.

Les résultats de ce travail ont confirmé l'efficacité de l'approche à l'étalonnage d'un système de mesure statique non-linéaire basé sur les fonctions spline. Les résultats obtenus ont servi dans le projet de recherche (UQAH/UQTR/CRSNG) concernant le développement des systèmes de mesure utilisant les capteurs en fibres optiques et ils ont permis de compléter un article concernant l'étalonnage de ce système de mesure, publié dans l'IEEE Transactions on Instrumentation and Measurement (avril 1992). Ces résultats peuvent être aussi utiles pour l'étalonnage d'autres systèmes de mesure dans le cas statique et non-linéaire.

REMERCIEMENTS

Je tiens d'abord à remercier mon directeur de thèse, Pr Andrzej Barwicz, ainsi que mon co-directeur, Pr Roman Z. Morawski, pour les conseils qu'ils m'ont donnés tout au long de ce travail.

Je remercie également M. Daniel Massicotte , M. Jean-Caude Desgagné et M. Nicolas Bonnet pour les encouragements et le soutien moral qu'ils ont su m'apporter tout au long de ce travail.

Enfin je désire exprimer ma reconnaissance pour les organismes qui m'ont supporté financièrement lors de mes études sur ce projet soit les fonds F.C.A.R. et les fonds du C. E. U. .

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

L'homme, depuis l'aube des temps, cherche à comprendre son environnement. Cette quête du savoir l'a obligé à développer des systèmes lui permettant de convertir des grandeurs physiques en unités numériques et traitables. Ces systèmes sont appelés ***systèmes de mesure***. La grandeur physique qui caractérise un objet de la mesure est désigné comme le mesurande [ASC87] . L'ensemble des opérations expérimentales qui concurent à la connaissance de la valeur numérique du mesurande constitue son mesurage. Lorsque la mesure utilise des moyens électroniques de traitement du signal, il est nécessaire de produire à partir du mesurande une grandeur électrique qui en soit une représentation aussi exacte que possible, ceci est un signal de mesure. D'où un signal de mesure représente une information accessible et traitable sur un phénomène physique.

Pour avoir la capacité d'interpréter le signal de mesure, il faut pour cela déterminer une relation reliant le signal de mesure au mesurande, bref il faut un moyen d'identifier un modèle mathématique du système. Ceci est l'étalonnage du système. D'où l'étalonnage est un processus permettant d'aboutir à des lois mathématiques reliant le signal de mesure au mesurande. L'étalonnage peut se faire de différentes façons soit par exemple par une corrélation mathématique, par un calcul de paramètres, une convolution de deux signaux , etc. Lorsque le temps n'est pas impliqué dans le modèle mathématique reliant le signal de mesure au

mesurande, l'étalonnage est dit statique et ce modèle a la forme d'une fonction qui doit être approximée.

Les méthodes d'approximation prirent formellement naissance au début du 18ième siècle . Plus particulièrement en 1823 quand Gauss publia un ouvrage intitulé "*Theoria Combinationes* " . Cet ouvrage présenta la méthode des moindres carrés. Cet méthode est à la base de plusieurs techniques d'approximation dites modernes.

Le but de cet ouvrage est d'étalonner un transducteur de pression composé de trois capteurs à fibre optique. Comme la caractéristique statique d'un capteur est hautement non-linéaire, son approximation se fera à l'aide d'une méthode non-paramétrée de type spline. Un principe de mesure utilisant trois capteurs de pressions à fibre optique fût présenté dans [BAR90a]. Ce principe sera repris et ajusté pour qu'il puisse être utilisé avec des méthodes non-paramétrées.

Donc la problématique consiste à élaborer différents algorithmes permettant d'approximer le mesurande à l'aide de signaux de mesure provenant de trois capteurs de pression, d'évaluer les performances de ces algorithmes et vérifier l'algorithme le plus performant en utilisant des données réelles.

Cet ouvrage est divisé en quatre chapitres. Le premier chapitre concerne la description du système de mesure avec les capteurs à fibre optique. Le deuxième chapitre porte sur les différentes méthodes d'étalonnages. Le troisième compare différents algorithmes d'étalonnage basés sur les fonctions splines pour le capteur de pression à fibre optique. Finalement le quatrième chapitre présente

les résultats pratiques obtenues pour l'algorithme présentant le meilleur résultat de l'étude comparative du chapitre 3.

CHAPITRE 1

DESCRIPTION DU SYSTÈME DE MESURE DE PRESSION À FIBRE OPTIQUE

1.1) Description du capteur de pression à fibre optique

Le capteur de pression est une fibre optique hautement biréfringente qui peut être vue comme une fibre interférométrique de type Mach-Zehnder qui possède deux modes de polarisations différents [BOC89].

Les fibres hautements biréfringentes (HB) peuvent maintenir un état de polarisation linéaire seulement si la lumière est émise dans un plan parallèle à un des deux axes principaux de la fibre. Ces axes possèdent un indice de réfraction légèrement différent. Par conséquent les deux plans perpendiculaires de polarisation ont différentes constantes de propagation (figure 1.1). Pour une

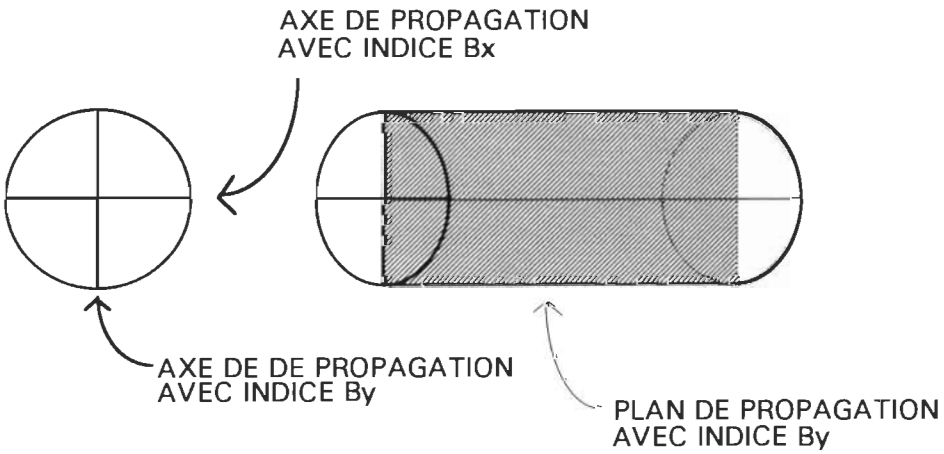

Figure 1.1 Les axes et le plan de propagation de la lumière par une fibre optique

direction de polarisation différente des deux axes de polarisation, la fibre HB est caractérisée par un indice de biréfringence B et une longueur de battement L_b définies de façon suivante:

$$B = \frac{\lambda}{L_b}, \quad L_b = \frac{2\pi}{\beta_x - \beta_y} \tag{1.1}$$

où λ est la longueur d'onde de la lumière , β_x et β_y sont les constantes de propagation pour les deux modes de polarisation linéaire et orthogonale de la fibre. L'état de la polarisation tourne suivant la direction de propagation de la lumière. Ici L_b est un paramètre représentant la longeur de battement et est défini par deux points d'égale phase sur la fibre. La longeur de battement L_b rend la fibre sensible à la pression et lorsqu'une pression est appliquée, cette pression augmente ou diminue la biréfringence de la fibre. D'où, si la lumière suit un plan différent des deux plans de propagation , l'intensité de la lumière devient sensible à la pression [BOC90].

La figure 1.2 montre une configuration possible pour l'utilisation du capteur.

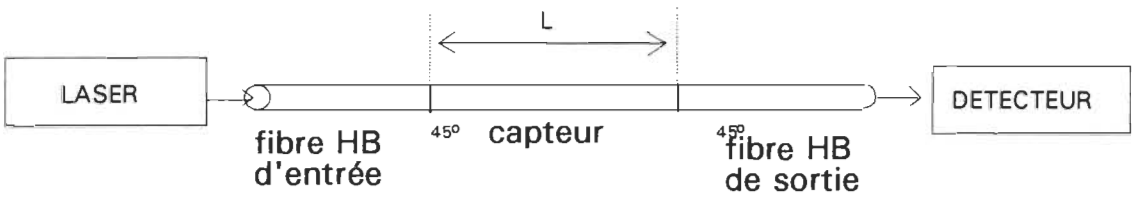

Figure 1.2 Configuration d'un capteur à fibre optique

Le fonctionnement du capteur est comme suit: Le laser envoi une lumière monochromatique qui suit un des plans de polarisation de la fibre. Ceci est représenté sur la figure 1.2 par la partie *fibre HB d'entrée* [BOC89]. Par ce fait la

lumière est faiblement affectée par la pression. Lorsqu'une longueur L de fibre est ajoutée avec un angle de 45° entre les axes de polarisation, ceci fait que la lumière suit un axe de propagation qui est à 45° des axes principaux et en conséquence l'intensité de la lumière devient sensible aux perturbations tel la pression. Pour récupérer le signal lumineux, on ajoute une fibre avec -45° entre les axes de polarisation de la fibre L et de la fibre HB de sortie. Le signal lumineux recueilli par le détecteur est transformé en un signal électrique. La dépendance de ce signal en fonction de la pression est quasi-sinusoïdale et la période de la sinusoïde est fonction de la longueur L de la fibre. Pour les besoins de simulation nous allons considérer cette caractéristique comme étant sinusoïdale. La figure 1.3 montre le signal électrique en fonction de la pression pour deux longueurs L différentes.

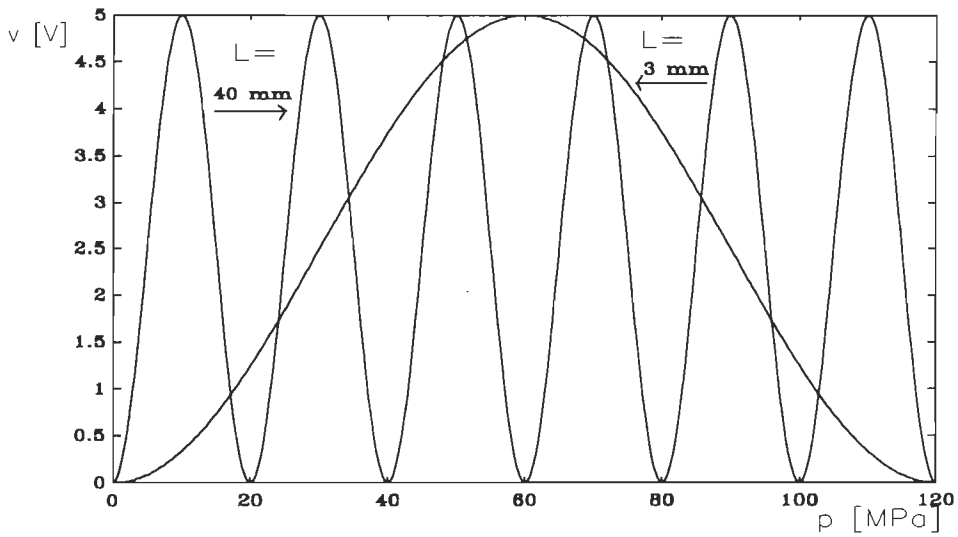

Figure 1.3 Signal électrique en fonction de la pression pour deux longueurs de fibre différentes

Comme notre but est de mesurer la pression, alors il faut considérer la possibilité d'inverser la caractéristique v=f(p). Si nous observons la figure 1.3,

nous observons deux contraintes importantes. La première contrainte est que la caractéristique n'est pas monotone. Ceci fait qu'il est impossible d'inverser f(p). La deuxième contrainte est que la fonction possède des extrémums et que ces extrémums engendrent une grande erreur de calcul. Ceci peut être résonné de la façon suivante: Si la tension recueillie possède une erreur de mesure Δv, alors cette erreur de mesure engendre une erreur de calcul sur la pression de valeur Δp et cette erreur sur la pression est d'autant plus élevée que la pente du signal est faible. Si nous observons la figure 1.3 nous constatons que les endroits où la pente est minimale sont aux extrémums du signal. La figure 1.4 représente bien ce fait.

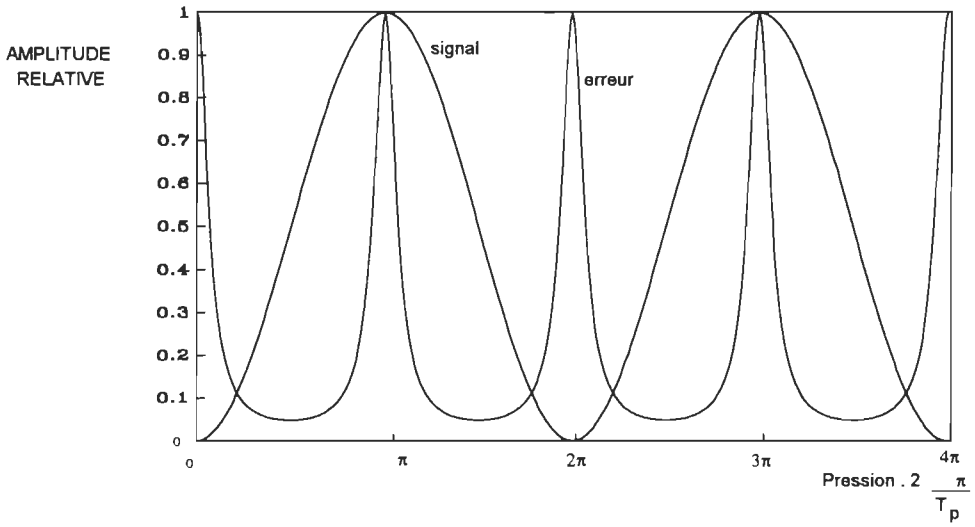

Figure 1.4 Représentation de l'erreur de mesure de la pression

La figure 1.4 montre que l'erreur est maximale aux extrémums du sinus. Donc si un système de mesure est développé avec uniquement un capteur, l'erreur sur la mesure de la pression ne sera pas la même pour une plage de pression considérée. Ceci est inacceptable parce qu'un bon système de mesure doit fournir une valeur du mesurande avec une erreur relativement constante sur la

plage de mesure. La prochaine section présente une façon adéquate de contourner ce problème.

Comme nous venons de le montrer, le capteur à fibre optique est sensible à la pression, ce qui est essentiel lorsqu'on veut développer un système pour la mesure de pressions. Mais en plus d'être sensible à la pression, il est sensible à la température et de façon non négligeable [BOC89]. La figure 1.5 présente la variation de la tension en fonction de la température pour une presion donnée.

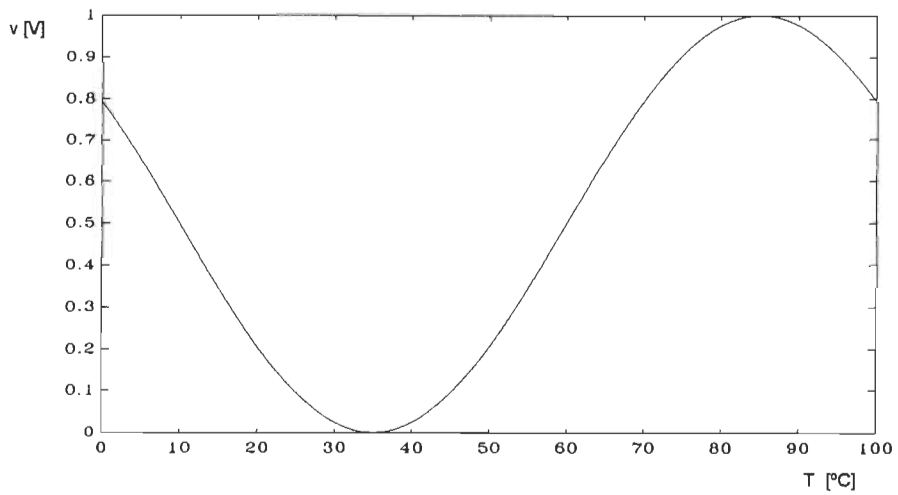

Figure 1.5 Caractéristique du capteur pour différentes températures et une pression constante

Donc pour bien étalonner le système, il nous faudra considérer la température dans la construction des fonctions.

1.2) Stratégie de mesure

Comme spécifié plus haut, l'erreur sur la pression est plus élevée aux extrêmums de la caractéristique statique du capteur à fibre optique. L'idée ici consiste à enlever ces régions. Plus précisément ces extrémums ne doivent pas

être employés lors du calcul de la pression. Pour ce faire nous allons utiliser un deuxième capteur de pression ayant la même caractéristique statique que le premier sauf qu'elle sera déphasé de 90° . D'où lors du calcul de la pression, le capteur choisit sera celui qui possèdera la pente la plus élevée pour une pression à mesurer. Comme le montre la figure 1.6 , les caractéristiques des capteurs C1 et C2 sont périodiques alors non inversibles: plusieurs valeurs de pressions correspondent à une valeur de tension. Donc pour pouvoir calculer adéquatement la pression à partir de ces capteurs il faut faire une première approximation de la pression par un troisième capteur ayant une caractéristique monotone. Ce capteur Cg, que nous nommons guide, permettra de choisir la pression correcte parmi les pressions possibles calculées à partir de la tension du capteur C1 ou C2.

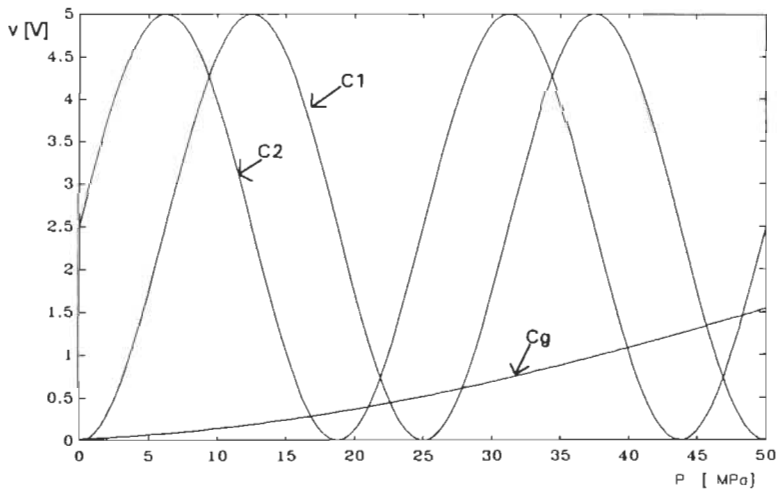

Figure 1.6 Caractéristique des trois capteurs nécessaires pour le calcul de la pression

L'utilisation de deux capteurs et d'un guide pour calculer la pression a pour conséquence d'augmenter l'exactitude, comme décrit précédemment, mais en plus elle augmente le temps du calcul de la pression mesurée. Ceci est dû au fait qu'on

doit calculer les valeurs de deux fonctions, soit celle qui caractérise le guide et celle qui caractérise le capteur choisi. Ceci est un compromis à payer entre précision et vitesse de calcul. La figure 1.8 présente la diminution de l'erreur relative lorsqu'on utilise deux capteurs déphasés. La figure 1.7 présente l'effet de la température sur un des capteurs de référence, C1 ou C2, et sur le guide. Une exigence majeur pour le guide est qu'il doit posséder une caractéristique monotone sur la plage de pression désirée et ce sur toute la plage de température désirée.

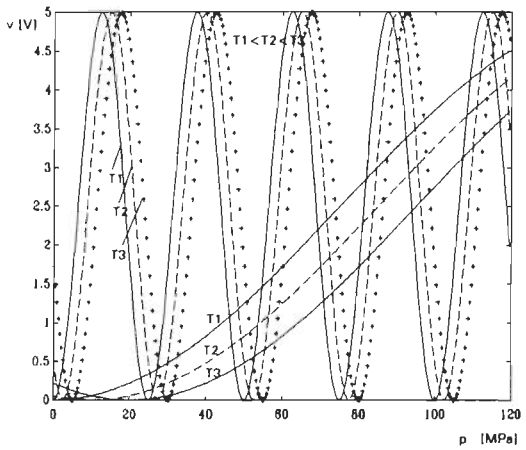

Figure 1.7 Sensibilité du guide et d'un capteur face aux variations de température

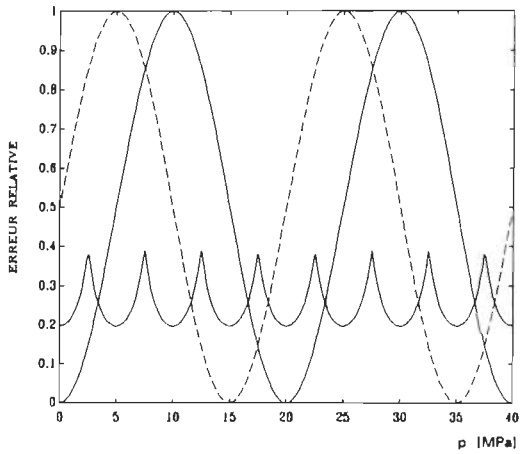

Figure 1.8 Représentation de l'erreur relative avec deux capteurs déphasés de 90°

Donc le système de mesure de pression doit calculer la pression à partir de quatre entrées: un signal de mesure représentant la température et trois signaux provenant des capteurs de pressions (guide + capteurs). Ici le signal de mesure représentant la température est considéré proportionel à cette température. En pratique, il existe des capteurs de température qui ont une bonne linéarité d'où cette présomption peut être utilisée dans les simulations.

1.3) Description du système de mesure

Le système de mesure possède trois parties spécifiques.

1. La conversion des signaux non électrique en signaux électriques et au conditionnement des signaux (conversion A/A).

2. La conversion des signaux électriques en données numériques (conversion A/D).

3. Le calcul du résultat de mesure à partir des données numériques (conversion D/D).

La première partie concerne la conversion de la pression en signaux lumineux par les capteurs et des signaux lumineux en signaux électriques. Ici les signaux électriques sont des tensions DC dont la variation est comprise entre 0 et 5 volts. La deuxième partie concerne la transformation des signaux électriques en données numériques par un convertisseur A/D. Ici nous supposons qu'un convertisseur A/D de 12 bits. Ce choix est fonction de la précision recherchée. La dernière partie concerne le calcul de la pression par un processeur numérique. Ce processeur calcule la pression à partir d'un modèle mathématique implanté dans sa mémoire.

Le processus du traitement de signaux dans un système de mesure peut être décomposé en deux parties: la conversion et la reconstitution. Ici la

conversion est assuré par les parties 1. et 2. décrites plus haut et la reconstitution est assurée par la partie 3. décrite précédemment. La figure 1.9 montre clairement ces différentes parties.

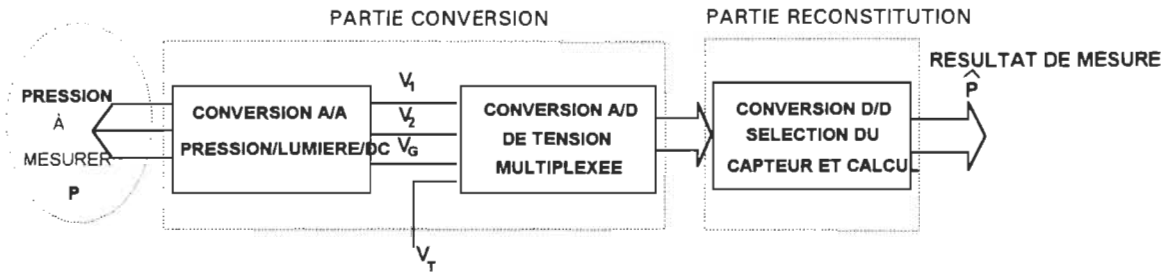

Figure 1.9 Décomposition fonctionnelle du système de mesure de pression

A partir de cette décomposition fonctionnelle du système, il est possible de représenter un algorithme général de mesure . Cet algorithme peut être décrit comme suit:

1) Mesurer successivement des tensions v1, v2, vg correspondant aux trois capteurs, aussi bien que de la quatrième tension qui comporte l'information sur la température;

2) Calculer la première approximation de la pression en utilisant vg. Ceci nous donne le résultats $\hat{p}_g$;

3) Faire le choix du capteur pour le calcul de la pression dont la caractéristique pour la pression $\hat{p}_g$ possède la pente la plus élevée.

4) Calculer la pression à partir de la fonction $p=f(T,V_C)$ où T représente la température et V_C la tension du capteur choisit résultant de l'étalonnage du système.

Nous venons de montrer qu'il nous faut quatre signaux de mesure pour pouvoir reconstruire le mesurande. Maitenant nous allons traiter des sources d'erreurs possibles qui peuvent influencer la précision du calcul.

1.4) Sources d'erreurs

Dans tous systèmes de mesure, il exite des sources d'erreurs. Ces sources peuvent provenir de bruits ambiants, de l'imprécision des éléments employés pour traiter les signaux , de l'instabilité des caractéristiques des capteurs et autres.

Pour le système ici traité, nous pouvons trouver cinq sources possibles d'erreurs.

1) Le modèle utilisé pour les calculs.

2) Les capteurs de pression.

3) Le capteur de température.

4) Le convertisseur A/D.

5) Le processeur numérique utilisé.

Ici on considère que l'erreur que fournit le capteur de pression est négligeable. Elle est négligeable car on considère que le capteur possède une exellente répétabilité et que les capteurs sont interchangeables.

Maintenant si on considère l'erreur provenant du choix du processeur. Comme le calcul se fait en point flottant et que la représentation du point flottant est gérée par des normes IEEE, les résultats obtenues par deux processeurs différents seront identiques si le nombre de chiffres significatifs est le même pour les deux processeurs. Ici l'erreur provient du nombre de bits servant à représenter une valeur. Pour ce qui est de l'influence du nombre de chiffres significatifs employés pour le calcul comme source importante d'erreur, nous considérons que le processeur employé dans le système de mesure utilise un même nombre de chiffre significatifs que le processeur utilisée pour les simulations.

Si nous utilisons un convertisseur A/D de 12 bits avec une plage de tension variant de 0 à 5 volts l'erreur de conversion équivalent à $\pm$ 1 LSB est de $\pm$0.002 volt.

Pour ce qui est du capteur de température, nous considérons que l'erreur sur la température est de $\pm$0.05 °C.

Si nous supposons que la source d'erreur provenant du modèle mathématique est négligeable alors pour l'étalonnage du système on devra considérer les erreurs sur les tensions, sur la température et sur les données servant à l'étalonnage.

Le prochain chapitre traite des différentes méthodes d'étalonnage. Elle se divisera en deux grandes parties. Soit une partie sur les méthodes paramétrées et une partie sur les méthodes non-paramétrées.

CHAPITRE 2

MÉTHODES D'ÉTALONNAGES STATIQUES

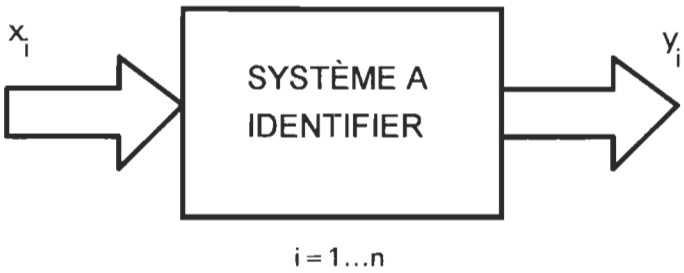

Figure 2.1 Système à identifier (n est le nombre d'observation)

L'étalonnage est une action servant à déterminer une relation empirique qui relie les résultats que fournit un capteur au mesurande. Ici le terme statique implique que le temps n'est pas un terme dominant du système. Donc l'étalonnage statique est un processus par lequel on désire approximer une fonction reliant les y_i aux x_i (voir figure 2.1). Pour commencer notre discussion nous allons considérer un modèle simple. À partir de ce modèle nous allons distinguer les différences entre les méthodes d'étalonnages paramétrées et les méthodes d'étalonnages non paramétrées.

Supposons le système de la figure 2.1. où nous observons comment réagit la sortie lorsqu'on lui applique à l 'entrée le vecteur $x=(x_1, x_2, ..., x_n)^T$. D'où nous avons pour chaque entrée x_i une sortie y_i correspondante. Alors disons que (x_i,y_i), $i=1,...,n$ sont les valeurs de x et y fait pour n observations. Maintenant supposons que les x_i et y_i sont reliés par le modèle d'étalonnage suivant:

$$y_i = \mu(x_i) + \varepsilon_i, i = 1,\ldots,n \qquad (2.1)$$

où ε_i sont des erreurs d'étalonnage de moyenne zéro et de variance σ^2 et $\mu(t_j)$ sont des valeurs d'une fonction μ inconnue. La fonction μ est habituellement référée comme étant une fonction de regression [EUB88].

Pour un modèle d'étalonnage paramétrée la forme de la fonction μ est connue, alors seulement un nombre fini de paramètres doit être déterminé. Les modèles paramétrés peuvent dépendre des paramètres d'une façon linéaire et/ou non-linéaire. Voici quelques exemples de modèles non-linéaires:

$$\mu(t) = \beta_1 t^{\beta_2}$$

$$\mu(t) = \beta_1 + \beta_2 e^{-\beta_3 t^{\beta_4}}$$

$$\mu(t) = \beta_1 \cos(\beta_2 t) + \beta_3 \sin(\beta_4 t)$$

où β_1, β_2,... sont les paramètres inconnus. À l'opposé un modèle paramétré linéaire a la forme suivante:

$$\mu(t) = \sum_{i=1}^{p} \beta_i x_i(t)$$

où les $x_i(t)$ sont des fonctions connues. Les modèles paramétrés sont des fonctions qui peuvent conduire à la déduction de μ. Le modèle résultant est une fonction qui estime μ [EUB88].

Un modèle de regression non paramétré, comme son nom peut le laisser supposer, ne fait aucune supposition qu'il existe une fonction unique pour estimer μ. Ceci fait que ces modèles sont très flexibles et s'ajustent très bien aux données. La seule supposition faite pour un modèle non-paramétré est que μ appartient à une collection de fonctions de dimension infinie. Par exemple on peut supposer que μ est différentiable ou différentiable avec une seconde dérivé intégrable au carré. Ces suppositions concernent seulement les propriétés qualitatives de μ. Si nous comparons les contraintes qu'un modèle paramétré demande face à un modèle non paramétré, les modèles paramétrés demandent un plus grand niveau de spécifications .

Une différence importante entre les modèles paramétrés et non-paramétrés est leurs degrés de dépendance sur les informations de μ obtenues de l'expérimentateur et des données. Pour construire un modèle de régression non-paramétré, l'expérimentateur choisit un espace de fonction auquel μ est supposé appartenir. Le choix est généralement motivé par les propriétés de lissage que la fonction de régression est supposée posséder. Les données sont alors utilisées pour déterminer un élément de cet espace de fonction qui est représentatif de la courbe de régression inconnue. À l'opposé, pour un modèle paramétré, l'expérimentateur choisit une famille de courbes possibles, d'une collection de plusieurs courbes, et entre ce choix dans un procédé de déduction. Les informations que les données peuvent fournir concernant le développement du modèle est restreint à ce qui peut être extrait des données sous cette présumée forme paramétrée [EUB88]. Conséquemment, les techniques de régression non-

paramétrées dépendent davantage sur les données pour obtenir des informations sur μ que leurs homonymes.

2.1) Méthodes d'étalonnage dites paramétrées

Nous allons traiter premièrement des modèles d'étalonnage paramétrés. L'étude se veut ici non exhaustive, mais présente les méthodes paramétrées les plus fréquemment utilisées.

2.1.1) Méthode des moindres carrés linéaires

Cette méthode consiste à trouver les paramètres β_i de la fonction (2.2)

$$\mu(t) = \sum_{j=1}^{p} \beta_j x_j(t) \tag{2.2}$$

où $\mu(t)$ est un estimateur satisfaisant l'équation (2.3)

$$y_i = \mu(t_i) + \varepsilon_i \quad , i=1,...,n \tag{2.3}$$

et les ε_i sont des erreurs de moyenne zéro et de variance σ^2. La méthode classique pour estimer les paramètres de l'équation (2.2) consiste à minimiser l'erreur moyenne suivante:

$$\mathrm{MSE}(\underline{\beta}) = \frac{\sum_{j=1}^{n}\left(y_j - \sum_{l=1}^{p}\beta_l x_l(t_j)\right)^2}{n} \tag{2.4}$$

pour les β_i . Cette approche peut être justifiée par le fait géométrique que la minimisation de (2.4) donne un estimé de μ qui est le plus près des y_j pour j=1, ...,n au point de vue d'une distance Euclédienne.

Il est facile de montrer [BJÖ90] que l'estimé $\hat{\beta}$ de β satisfait l'équation normale

$$X' X\underline{\hat{\beta}} = X' \underline{y} \tag{2.5}$$

où

$$\underline{\hat{\beta}} = (\hat{\beta}_1, \ldots, \hat{\beta}_p)'$$

$$\underline{y} = (y_1, \ldots, y_n)'$$

et

$$\mathbf{X} = \begin{vmatrix} x_1(t_1) & \cdots & x_p(t_1) \\ \vdots & \ddots & \vdots \\ x_1(t_n) & \cdots & x_p(t_n) \end{vmatrix}$$

alors la solution pour trouver $\hat{\beta}$ est donnée par (2.6).

$$\underline{\hat{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\underline{y} \tag{2.6}$$

La solution de cette équation correspond à la méthode des moindres carrés. Ici la difficulté dans l'équation (2.6) consiste en l'inversion de la matrice. Plusieurs méthodes fûrent développées pour résoudre ce problème tel que la méthode de décomposition QR [BJÖ90] ou la décomposition de type Cholesky [BJÖ90].

2.1.2) Méthode des moindres carrés non-linéaires

Considérons le modèle (2.1) avec une fonction μ non-linéaire. Plus spécifiquement , disons qu'il existe une fonction connue $\mu(t,\underline{\beta})$ qui est non-linéaire pour au moins un de ses paramètres. De plus, nous supposons que l'estimateur $\mu(t)$ est égale à la fonction $\mu(t,\underline{\beta})$. Des exemples de fonctions non-linéaires ont été donnés précédemment. Nous allons discuter dans ce qui suit d'une méthode de déduction à partir d'une estimation par moindre carré.

La fonction est maintenant considérée dépendre non-linéairement des paramètres $\underline{\beta}$. Par contre, pour estimer $\underline{\beta}$ et, du même coup, $\mu(t)$, nous pouvons encore procéder comme dans le cas d'un modèle linéaire et utiliser un estimateur par moindre carré. Un estimé de $\underline{\beta}$ (définie $\hat{\underline{\beta}}$) doit être trouvé en minimisant

$$\mathrm{MSE}(\underline{\beta}) = \frac{\sum_{j=1}^{n}\left(y_j - \mu(t_j;\underline{\beta})\right)^2}{n} \qquad (2.7)$$

Comme la fonction $\mu(t)$ est non-linéaire, il peut exister plusieurs solutions qui minimisent l'équation (2.7).

Considérant que $\mu(t,\underline{\beta})$ est différentiable par rapport aux β_i , nous pouvons différentier l'équation (2.7) pour voir que $\hat{\underline{\beta}}$ doit satisfaire l'équation suivante

$$\sum_{j=1}^{n} \hat{x}_{ij}\, \mu(t_j\,;\hat{\underline{\beta}}) \;=\; \sum_{j=1}^{n} \hat{x}_{ij}\, y_j \;, \qquad \text{i=1,...,p} \qquad (2.8a)$$

avec

$$\hat{x}_{ij} = \left.\frac{\partial \mu(t_j ; \underline{\beta})}{\partial \beta_i}\right|_{\underline{\beta}=\hat{\underline{\beta}}} \tag{2.8b}$$

Les équations (2.8a) et (2.8b) sont des équations non-linéaires analogues à l'équation normale (2.5).

Il existe une variété d'algorithmes qui ont été conçus pour résoudre les équations (2.8a) et (2.8b). Un des plus connus est la méthode de Gauss-Newton [JAM85] [MEL82] que nous allons maintenant présenter.

Supposons que $\underline{\beta}_0$ est une première approximation prise au hasard de $\hat{\underline{\beta}}$. Alors il y a un vecteur $\partial\underline{\beta}_0 = (\partial\beta_{10},\ldots,\partial\beta_{p0})'$ tel que $\hat{\underline{\beta}}=\underline{\beta}_0+\partial\underline{\beta}_0$. Une expansion du premier ordre en série de Taylor de $\mu(t,\underline{\beta})$ pour $\underline{\beta}_0$ nous permets d'écrire

$$\mu(t_j;\hat{\underline{\beta}}) \dot{=} \mu_{j0} + \sum_{l=1}^{p} x_{ij0}\partial\beta_{i0} \tag{2.9a}$$

avec $\mu_{j0} = \mu(t_j;\underline{\beta}_0)$ et

$$x_{ij0} = \left.\frac{\partial \mu(t_j;\underline{\beta})}{\partial \beta_l}\right|_{\underline{\beta}=\underline{\beta}_0} \tag{2.9b}$$

Substituant (2.9) dans (2.7), l'équation à minimiser devient

$$\mathrm{MSE}(\hat{\underline{\beta}}) \dot{=} \tilde{\mathrm{MSE}}(\partial\underline{\beta}_0) = \frac{\sum_{j=1}^{n}\left(y_j - \mu_{j0} - \sum_{i=1}^{p} x_{ij0}\partial\beta_{i0}\right)^2}{n} \tag{2.10}$$

Le vecteur $\partial\underline{\beta}_0 = \hat{\underline{\beta}} - \underline{\beta}_0$ est toujours inconnu. Par contre, puisque $\mathrm{MSE}(\hat{\underline{\beta}})$ est l'erreur par moindre carré minimale, elle doit correspondre à la valeur minimale de $\tilde{\mathrm{MSE}}(\partial\underline{\beta}_0)$. Comme montré pour le cas linéaire, ce minimum se produit pour un vecteur $\partial\hat{\underline{\beta}}_0$ satisfaisant

$$\mathbf{X}(\underline{\beta}_0)'\mathbf{X}(\underline{\beta}_0)\partial\hat{\underline{\beta}}_0 = \mathbf{X}(\underline{\beta}_0)'\underline{e}(\underline{\beta}_0) \tag{2.11a}$$

où

$$\mathbf{X}(\underline{\beta}) = \begin{vmatrix} \dfrac{\partial\mu(t_1;\underline{\beta})}{\partial\beta_1} & \cdots & \dfrac{\partial\mu(t_1;\underline{\beta})}{\partial\beta_p} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial\mu(t_n;\underline{\beta})}{\partial\beta_1} & \cdots & \dfrac{\partial\mu(t_n;\underline{\beta})}{\partial\beta_p} \end{vmatrix} \tag{2.11b}$$

et

$$\underline{e}(\underline{\beta}) = \left(y_1 - \mu(t_1;\underline{\beta}),\ldots,y_n - \mu(t_n;\underline{\beta})\right)' \tag{2.11c}$$

Puisque (2.10) est seulement une approximation il s'en suit que $\underline{\beta}_0 + \partial\hat{\underline{\beta}}_0$ est seulement une approximation de $\hat{\underline{\beta}}$. et que $\mathrm{MSE}(\underline{\beta}_0 + \partial\hat{\underline{\beta}}_0)$ ne sera pas l'erreur quadratique minimale. Pour remédier au problème nous pouvons répéter le

procédé en posant $\underline{\beta}_1=\underline{\beta}_0+\delta\hat{\underline{\beta}}_0$, écrire que $\hat{\underline{\beta}}=\underline{\beta}_1+\partial\underline{\beta}_1$ et obtenir $\underline{\beta}_2=\underline{\beta}_1+\delta\hat{\underline{\beta}}_1$ avec $\partial\hat{\underline{\beta}}_1=\left(X(\underline{\beta}_1)'X(\underline{\beta}_1)\right)^{-1}X(\underline{\beta}_1)'\underline{e}(\underline{\beta}_1)$. Ce procédé peut être repété encore et encore jusqu'à ce qu'il y ait convergence. Par exemple, si $\underline{\beta}_j$ et $\underline{\beta}_{j+1}$ sont les approximations de $\hat{\underline{\beta}}$ à la j ième et (j+1)ième itération, alors il peut être décidé d'itérer jusqu'à

$$\frac{\left(\underline{\beta}_{j+1}-\underline{\beta}_j\right)'\left(\underline{\beta}_{j+1}-\underline{\beta}_j\right)}{\underline{\beta}_j'\underline{\beta}_j}<\alpha_1$$

et

$$\frac{\left|\mathrm{MSE}(\underline{\beta}_{J+1})-\mathrm{MSE}(\underline{\beta}_J)\right|}{\mathrm{MSE}(\underline{\beta}_J)}<\alpha_2$$

pour des valeurs de α_1 et α_2 assez faibles.

La méthode Gauss-Newton fonctionne bien pour la plupart des cas, mais il existe des cas où cette méthode ne converge pas. Dans ces cas, il faut utiliser d'autres méthodes telles que la méthode de Gauss-Newton modifiée ou la méthode de Newton [BJÖ90], [GIL78].

2.2) Méthodes d'étalonnage dites non-paramétrées

Il existe plusieurs méthodes non-paramétées. Le but de cette section est de présenter les méthodes les plus courantes de même que celles que nous allons utiliser pour étalonner le système de mesure traité précédemment.

Nous allons présenter deux classes de méthodes non-paramétrées. La première classe considère les données comme non-bruitées, tandis que la seconde considère les données comme bruitées.

2.2.1) Méthodes non-paramétrées avec données non-bruitées

Comme la construction de ces méthodes utilise différentes fonctions dont une des exigeances est que la courbe définie par ces fonctions passe par les points servant à la construction des fonctions, alors nous pouvons dire que ces méthodes utilisent des fonctions d'interpolations. Dans cette partie nous allons présenter deux méthodes générales d'interpolations, soit les interpolations polynômiales et les interpolations splines .

2.2.1.1) Interpolations polynômiales

Supposons que nous ayons un ensemble de n+1 points

$$P_0(x_0, y_0), P_1(x_1, y_1), \ldots, P_n(x_n, y_n) \tag{2.12}$$

et nous supposons que les x_k sont distincts et ordonnés, soit

$$x_0 < x_1 < \ldots < x_k < x_{k+1} < \ldots < x_n \tag{2.13}$$

Notre objectif est de trouver un polynôme qui interpole plusieurs de ces points, c'est-à-dire que la courbe passe par quelques uns ou l'ensemble des $P_0, \ldots, P_n$ tel qu'illustré à la figure 2.2

Sur cette figure p désigne un polynôme et P un point que le polynôme interpole. En particulier $p_{k,k+m}(x)$ dénote un polynôme qui interpole m+1 points consécutifs, soit P_k, P_{k+1}, . . . ,P_{k+m}.

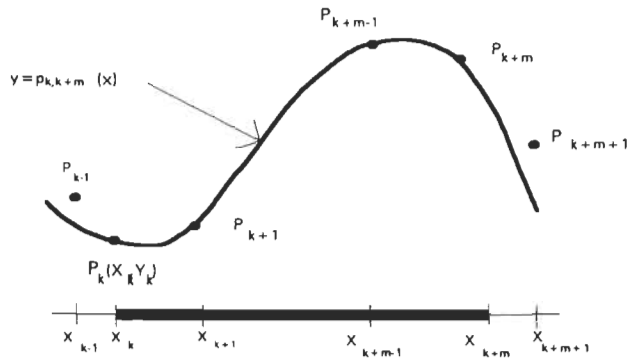

Figure 2.2 Interpolation polynômiale

Exigences pour l'interpolation polynômiale

La première exigence est que la fonction $p_{k,k+m}(x)$ passe par les m+1 points d'où ceci impose les m+1 contraintes suivantes:

$$p_{k,k+m}(x_i)=y_i \qquad \text{pour } i=k,k+1, \ldots ,k+m \tag{2.14a}$$

La deuxième exigence est que $p_{k,k+m}(x)$ a au plus m+1 coefficients, d'où

$$p_{k,k+m}(x) \text{ est un polynôme de degrée} \leq m. \tag{2.14b}$$

Donc donnant un ensemble de points $P_0, \ldots, P_n$ et les indices k et k+m, le problème de trouver un polynôme $p_{k,k+m}(x)$ satisfaisant les deux exigences (2.14a) et (2.14b) est appelé interpolation polynômiale [MEL82], [JAM85].

Il existe plusieurs méthodes pour construire $p_{k,k+m}(x)$. Parmi ces méthodes nous allons en décrire deux, soit la méthode de Lagrange et la méthode de Newton.

Interpolation de Lagrange

Il est possible de montrer que $p_{k,k+m}(x)$ peut être écrit sous la forme suivante [MAR82]:

$$p_{k,k+m}(x) = y_k L_k(x) + y_{k+1} L_{k+1}(x) + \cdots + y_{k+m} L_{k+m}(x) \qquad (2.15)$$

$p_{k,k+m}$ est un polynôme de degré inférieur ou égal à m

$$\text{où } L_j = \frac{\prod_{i \neq j} (x - x_i)}{\prod_{i \neq j} (x_j - x_i)} = \frac{(x - x_k)\ldots(x - x_{j-1})(x - x_{j+1})\ldots(x - x_{k+m})}{(x_j - x_k)\ldots(x_j - x_{j-1})(x_j - x_{j+1})\ldots(x_j - x_{k+m})} \qquad (2.16)$$

alors L_j est le $j^{ième}$ polynôme de Lagrange. C'est un polynôme de degré m qui est égal à zéro pour x_i avec i=k, k+1, . . ., j-1, j+1, . . . ,k+m sauf pour i=j où il prend la valeur un.

$$L_j(x_i) = \begin{Bmatrix} 0, & i \neq j \\ 1, & i = j \end{Bmatrix} \qquad (2.17)$$

Interpolation de Newton

L'interpolation de Newton est basée sur les formules de différences. Supposons que nous avons n points, alors les valeurs

$$\Delta y_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}, \qquad k = 0,1,\ldots,n-1$$

sont appelées les premières différences aux points k (P_k). Géométriquement, Δy_k correspondont à la pente un pas en avant de P_k. La seconde différence aux P_k est:

$$\Delta^2 y_k = \frac{\Delta y_{k+1} - \Delta y_k}{x_{k+2} - x_k}, \qquad k = 0,1,\ldots,n-2$$

Par induction pour m≥1, nous définissons la m^{ième} différence aux P_k comme étant égale à :

$$\Delta^m y_k = \frac{\Delta^{m-1} y_{k+1} - \Delta^{m-1} y_k}{x_{k+m} - x_k}, \qquad k = 0,1,\ldots,n-m \qquad (2.18)$$

D'où à partir de l'équation (2.18) il est possible de construire des polynômes d'interpolation $p_{k,k+m}(x)$. La forme de $p_{k,k+m}(x)$ est la suivante:

$$p_{k,k+m} = p_{k,k+m-1} + \Delta^m y_k (x - x_k)(x - x_{k+1})\ldots(x - x_{k+m-1}) \qquad (2.19)$$

Nous voyons que l'équation (2.19) a une forme récursive. Sachant que $P_{k,k}=y_k$, alors (2.19) peut être développée et aboutir à l'équation (2.20).

$$p_{k,k+m} = y_k + \Delta y_k(x - x_k) + \Delta^2(x - x_k)(x - x_{k+1}) \\ \ldots + \Delta^m y_k(x - x_k)(x - x_{k+1})\ldots(x - x_{k+m-1}) \tag{2.20}$$

Un des problèmes majeurs de l'interpolation polynômiale est sa tendance à osciller lorsque l'ordre du polynôme augmente, d'où l'exactitude de l'interpolation diminue lorsque le degrée da la fonction augmente.

2.2.1.2) Interpolation spline

Les fonctions splines sont des fonctions composées de polynômes qui sont créées pour chaque intervale d'un ensemble de points. Elles permettent d'approximer une courbe sans discontinuité et sans oscillation.

Supposons que nous possédons n+1 points

$$P_0(x_0, y_0),\ P_1(x_1, y_1), \ldots, P_n(x_n, y_n) \tag{2.21}$$

avec $x_0 < x_1 < \ldots < x_n$, alors l'idée consiste à trouver un polynôme de degré m

$$f_k(x) = c[m]_k(x - x_k)^m + c[m-1]_k(x - x_k)^{m-1} + \cdots + c[1]_k(x - x_k) + c[0] \tag{2.22a}$$

pour chaque intervalle $[x_k, x_{k+1}]$ qui satisfait les conditions d'interpolations

$$f_k(x_k) = y_k \\ f_k(x_{k+1}) = y_{k+1} \tag{2.22b}$$

La difficulté de cette méthode réside dans la détermination des coefficients $c[m]_k$ pour $0 \leq k \leq n$.

L'idée des fonctions splines consiste à égaler les dérivées des différents polynômes pour chaque points de collage $P_k(x_k,y_k)$. Si nous exprimons ceci mathématiquement, il faut que

$$\begin{aligned} f_k^1(x_{k+1}) &= f_{k+1}^1(x_{k+1}) \\ &\vdots \\ f_k^{m-1}(x_{k+1}) &= f_{k+1}^{m-1}(x_{k+1}) \end{aligned} \tag{2.23}$$

où $f^m(x)$ est la $n^{ième}$ dérivé d'une fonction f(x).

Si le degré du polynôme est de m, il faut que m-1 dérivés soient égales. Il a été démontré que l'application de (2.23) à la construction de (2.22) donne n-m+1 équations, donc il faut déterminier m coefficients arbitrairement. Il existe différentes méthodes pour déterminer ces coefficients mais plus généralement on pose m coefficients égaux à zéro. Ce type d'interpolation a pour nom spline naturelle [DEB78], [SCH81]. Nous allons donner un exemple de développement. Comme la forme est générale, le développement pour d'autres types de polynômes se fait similairement à la méthode qui sera présentée ici. Cet exemple est pour un polynôme de degré trois. Ce type de spline a pour nom spline cubique.

Spline cubique

Pour un sous intervalle $[x_i,x_{i+1}]$ où $0 \le i \le n$, on doit trouver un polynôme de la forme suivante [PRE86]:

$$f_i(x) = A_i + B_i(x - x_i) + C_i(x - x_i)^2 + D_i(x - x_i)^3 \tag{2.24}$$

où A_i, B_i, C_i et D_i sont des paramètres à déterminer pour chaque intervalle $[x_i,x_{i+1}]$. De la condition $f_k(x_k)=y_k$ nous trouvons

$$y_i = A_i \tag{2.25}$$

à partir de l'équation (2.23) nous trouvons

$$y_{i+1} = A_i + B_i h_i + C_i h_i^2 + D_i h_i^3 \tag{2.26}$$

$$B_{i+1} = B_i + 2C_i h_i + 3D_i h_i^2 \tag{2.27}$$

$$C_{i+1} = 3D_i h_i + C_i \tag{2.28}$$

avec $h_i = x_{i+1} - x_i$, Remplacant (2.25), (2.26), (2.27) et (2.28) dans (2.24) nous trouvons

$$C_{i-1}h_{i-1} + 2(h_i + h_{i-1})C_i + C_{i+1}h_i = 3\left[\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right] \tag{2.29}$$

si nous posons

$$Q^T \underline{y} = \begin{pmatrix} \frac{1}{h_0} & -\left(\frac{1}{h_0}+\frac{1}{h_1}\right) & \frac{1}{h_1} & & & \\ & \frac{1}{h_1} & -\left(\frac{1}{h_1}+\frac{1}{h_2}\right) & \frac{1}{h_2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{1}{h_{n-1}} & -\left(\frac{1}{h_{n-1}}+\frac{1}{h_n}\right) & \frac{1}{h_n} \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{2.30}$$

alors nous pouvons écrire (2.29) sous forme matricielle, ce qui donne

$$\begin{pmatrix} h_0 & 2(h_0+h_1) & h_1 & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \end{pmatrix}_{(n-1)(n+1)} \begin{pmatrix} C_0 \\ C_1 \\ \vdots \\ C_n \end{pmatrix}_{(n+1)*1} = Q^T \underline{y} \tag{2.31}$$

Alors nous avons n-1 équations pour n+1 inconnues. Il faut trouver deux paramètres, par exemple C_0 et C_n, d'une autre façon. Comme décrit précédemment on peut les poser égaux à zéro (dans ce cas on appelle les splines: spline naturelle) ou les trouver par une méthode numérique tel que la méthode de Lagrange (voir [DEB78]). Si nous observons (2.31), on voit que la matrice est tridiagonale, d'où l'équation (2.31) peut être résolue rapidement en utilisant la décomposition LU [JAM85]. Une fois les paramètres C_i trouvés, il est facile de calculer à partir de (2.27) et de (2.28) les paramètres D_i et B_i.

Pour les splines naturelles, vue que C_0 et C_n sont égaux à zéro, on peut poser la forme suivante:

$$T\underline{C} = \begin{pmatrix} \frac{2(h_0+h_1)}{3} & \frac{h_1}{3} & & & \\ \frac{h_1}{3} & \frac{2(h_1+h_2)}{3} & \frac{h_2}{3} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{h_{n-2}}{3} & \frac{2(h_{n-2}+h_{n-1})}{3} & \end{pmatrix}_{(n-1)(n-1)} \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_{n-1} \end{pmatrix}_{(n-1)*1} \tag{2.32}$$

ce qui donne la forme matricielle suivante:

$$T\underline{C} = Q^T \underline{y} \tag{2.33}$$

Lorsque les données servant à la construction des splines ne sont pas bruitées, cette méthode est très efficace. Par contre lorsque les données sont buitées, il faut procéder différemment. On peut utiliser des estimateurs non-paramétrés. La prochaine section traite de deux types d'estimateurs non-paramétrés, soit les splines de lissage et les splines moindre carré.

2.2.2) Méthodes non-paramétrés avec données bruitées

Pour cette section, nous considérons que les données utilisées à la construction du modèle mathématique représentant différentes courbes sont bruitées. D'où les modèles mathématiques construits à partir des données devront considérer deux critères, la précision de la courbe et le degrée de lissage de cette dernière. Nous allons présenter deux méthodes qui tiennent compte de ces critères. La première concerne les splines de lissage et la seconde les splines moindre carré.

2.2.2.1) Splines de lissage

Nous considérons que nous avons le modèle de régression suivant:

$$y_j = \mu(t_j) + \varepsilon_j \tag{2.34}$$

où $a \leq t_1 < \ldots < t_n \leq b$ et les ε_j sont des erreurs non corrélées, de moyenne zéro et de variance σ^2 . Considérons que μ est une fonction possédant un certain lissage. Le but est de retouver μ à partir des données bruitées (t_j, y_j). L'estimé $\hat{\mu}(t)$ de $\mu(t)$ est faite à partir de fonctions f qui, pour un certain paramètre $p \in [0,1]$, minimisent le critère suivant:

$$J = p\sum_{j=1}^{n}\left(\frac{y_j - f(t_j)}{\sigma_j}\right)^2 + \int_a^b \left(f^{(m)}(t)\right)^2 dt \tag{2.35}$$

pour toutes fonctions f possédant m dérivés continues. La minisation de (2.35) constitue un compromis entre le désir de rester près des données et le désir

d'avoir une fonction avec un certain degré de lissage. Le choix du paramètre p est fonction de l'importance qu'on accorde à l'un ou à l'autre de ces objectifs [REI67], [REI71], [DEB78], [FES91], [HUT85].

La solution $\hat{\mu}(t)$ est une fonction spline polynômiale de degré k=2m et satisfaisant la condition pour les splines naturelles, soit que

$$f^{(J)}(t_1) = f^{(J)}(t_n) = 0 \quad \text{pour j = m,...,k-2}$$

Ceci correspond aux conditions générales pour construire des splines de lissage de degré k=2m. Ici nous allons donner un exemple de développement pour m=2 . Ce qui correspond au spline cubique de l'équation (2.24)

Nous avons vu précédemment que pour les splines d'interpolation nous avons la condition $f_k(t_k)=y_k$. Pour les splines de lissage cette condition ne peut s'appliquer: en général $f_k(t_k)=A_k \neq y_k$ modifie l'équation (2.33) pour donner

$$T\underline{C} = Q^T \underline{A} \tag{2.36}$$

Maintenant développons l'intégrale de (2.35) en terme de $\underline{A}$ et de $\underline{C}$. Nous savons que pour une ligne droite L on peut approximer l'intégrale suivant (2.37).

$$\int_0^h (L(x))^2 dx \approx \frac{h}{3}\left[(L(0))^2 + L(0)L(h) + (L(h))^2\right] \tag{2.37}$$

Pour une spline cubique

$$\left(f_j^{(2)}(t)\right)^2 = 4C_j^2 + 24C_jD_j(x - x_j) + 36D_j^2(x - x_j)^2 \qquad (2.38)$$

d'où en utilisant (2.28) on obtient

$$\int_a^b \left(f^{(2)}(t)\right)^2 dt = \sum_{i=1}^{n-1} \int_a^b \left(f_j^{(2)}(t)\right)^2 dt = \frac{4}{3}\sum_{i=1}^{n-1} h_i\left(C_i^2 + C_iC_{i+1} + C_{i+1}^2\right) \qquad (2.39)$$

avec $h_i=x_{i+1}-x_i$. Remplaçant (2.39) dans (2.35) alors Il est possible d'écrire l'équation (2.35) sous la forme matricielle suivante

$$J[\underline{A},\underline{C}] = p\left(\underline{y} - \underline{A}\right)^T D^{-2}\left(\underline{y} - \underline{A}\right) + 2\underline{C}^T T\underline{C} \qquad (2.40)$$

où D est une matrice diagonale $D=\text{diag}(\sigma_1,\ldots,\sigma_n)$ et T la matrice définie à (2.32). Ici les paramètres σ_i sont des valeurs connues. Si disponible on doit utiliser pour les σ_i l'écart type des données. Le but est de minimiser (2.40). Comme (2.40) est exprimée en fonction de deux vecteurs de paramètres, alors il nous faut la réécrire pour un vecteur. Nous voyons par l'équation (2.36) que $\underline{C} = T^{-1}Q^T\underline{A}$ d'où nous pouvons écrire (2.40) en terme de $\underline{A}$ uniquement. Ce qui donne (2.41)

$$J[\underline{A}] = p\left(\underline{y} - \underline{A}\right)^T D^{-2}\left(\underline{y} - \underline{A}\right) + 2\left(T^{-1}Q^T\underline{A}\right)^T T\left(T^{-1}Q^T\underline{A}\right) \qquad (2.41)$$

Comme il faut minimiser (2.41), alors on doit la dériver par rapport au vecteur $\underline{A}$ et l'égaler à zéro, d'où nous obtenons (2.42)

$$-2pD^{-2}(\underline{y} - \underline{A}) + 4\left(T^{-1}Q^T\right)^T T\left(T^{-1}Q^T\right)\underline{A} = 0 \qquad (2.42)$$

Profitant du fait que $T^{-1}=(T^{-1})^T$ [JAM85], alors on peut simplifier (2.42) pour donner l'équation (2.43).

$$Q\underline{C} = pD^{-2}(\underline{y} - \underline{A}) \tag{2.43}$$

donc à partir de (2.43) et de (2.36) il est possible de trouver les paramètres nécessaires à la construction des splines. Rappelons que les équations (2.36) et (2.43) sont:

$$T\underline{C} = Q^T \underline{A}$$
$$Q\underline{C} = pD^{-2}(\underline{y} - \underline{A})$$

Nous allons essayer d'écrire ces équations sous une autre forme. Pour ce faire on multiplie les deux parties de chaque côté de l'égalité de l'équation (2.43) par $Q^T D^2$, alors nous pouvons séparer la variable $\underline{C}$ des deux équations (2.43) et (2.36), ce qui donne les équations (2.44) et (2.45).

$$\left(Q^T D^2 Q + pT\right)\underline{C} = pQ^T \underline{y} \tag{2.44}$$

$$\underline{A} = \underline{y} - p^{-1} D^2 Q \underline{C} \tag{2.45}$$

Ici précisons que $\underline{A}$ est un estimé. D'où pour un p choisi l'équation (2.45) fournit les paramètres A_i, l'équation (2.44) fournit les C_i et les équations (2.27) et (2.28) fournissent les B_i et les D_i servant à construire les splines de forme

$$f_i(x) = A_i + B_i(x - x_i) + C_i(x - x_i)^2 + D_i(x - x_i)^3$$

D'où le nom de spline de lissage. Ici il nous faut présenter l'estimateur sous une autre forme. Pour ce faire il faut remplacer $\underline{C}$ dans (2.45) par (2.44) et isoler $\underline{y}$. Ce qui nous donne (2.46)

$$\underline{A} = \left[\mathrm{I} - D^2 Q \left(Q^T D^2 Q + pT \right)^{-1} Q^T \right] \underline{y} \tag{2.46}$$

qui peut être écrit sous la forme

$$\underline{A} = H(p)\underline{y} \tag{2.47a}$$

avec

$$H(p) = \left[\mathrm{I} - D^2 Q \left(Q^T D^2 Q + pT \right)^{-1} Q^T \right] \tag{2.47b}$$

Ici H(p) porte le nom de matrice chapeau [EUB88] . Cette matrice est utilisée dans la détermination de p.

Calcul du paramètre p

La détermination du paramètre p est guidé par les données. La première méthode pour déterminer p fût présentée par [REI67].

Cette méthode consiste à introduire un paramètre S dans le calcul des splines de lissage . Ce paramètre a une valeur comprise dans l'intervalle

$$n - \sqrt{2n} \le S \le n + \sqrt{2n} \tag{2.48}$$

et sert à déterminer p. Ceci se fait à partir de l'équation suivante

$$F(p) = \sqrt{S} \tag{2.49}$$

avec

$$F(p) = \left\| DQ(Q^T D^2 Q + pT)^{-1} Q^T \underline{y} \right\|_2 \qquad (2.50)$$

Comme montré dans [REI67] le calcul de p se fait à partir d'une méthode pour trouver des racines, soit la méthode de Newton. De plus l'article fournit un algorithme permettant ce calcul. Cet algorithme est comme suit:

1. Commencer avec p=0;
2. Faire une décomposition de type Cholesky R^TR de Q^TD^2Q+pT;
3. Calculer $\underline{u}$ à partir de $R^TR\underline{u} = Q^T\underline{y}$ et $\underline{v} = DQ\underline{u}$, accumuler $e=\underline{v}^T\underline{v}$;
4. Si e>S
   - calculer $f=\underline{u}^T\underline{u}$ et $g=\underline{w}^T\underline{w}$ où $R^T\underline{w}=T\underline{u}$
   - remplacer p par $p + \dfrac{e - \sqrt{S \times e}}{f - p \times g}$
   - repartir à 2.

   sinon
   - calculer $\underline{A} = \underline{y} - D\underline{v}$, $\underline{C} = p\underline{u}$.

Cette méthode a l'inconvénient de donner un lissage trop profond. Par contre même si le lissage est profond, cette méthode peut être utilisée avec peu de points et donner de bons résultats ($n \leq 100$). Si le nombre de points est assez élevés, cette méthode doit être remplacée par la méthode G.C.V. (ang. "General Cross Validation") [WAH79].

Cette méthode est reconnue aujourd'hui comme étant la méthode à utiliser pour déterminer p. Ceci est dû aux différentes propriétés optimales que possède cette méthode [EUB88], [WAH80], [ROB89].

Cette méthode consiste à minimiser un critère défini à partir de la matrice chapeau H(p)

$$GCV(p) = \frac{\sum_{j=1}^{n} \left(y_j - A_j\right)^2}{n\left(1 - \frac{\sum_{j=1}^{n} h_{jj}(p)}{n}\right)^2} \qquad (2.51)$$

où $h_{jj}(p)$ sont les éléments diagonale de la matrice H(p) définie par l'équation (2.47b). La façon d'utiliser ce critère est la suivante. Premièrement on subdivise p en k parties. D'où $p_i=i/k$ pour i=0,...,k. Pour chaque p_i on calcule $GCV_i(p_i)$. Donc la meilleur valeur de p correspond au p_i donnant la valeur minimum des $GCV_i(p)$.

Nous venons de montrer deux méthodes pour le calcul du paramètre p. Considérant que le nombre de points servant à construire notre modèle est faible, la première méthode sera celle que nous allons considérer lors de notre étude.

2.2.2.2) Splines moindre carré

Dans cette section, nous allons considérer une méthode de régression non-paramétrée qui a pour nom splines moindre carré [EUB88].

Pour commencer, rappelons le modèle de base où nous avons les couples de données (t_i, y_i) qui satisfont la relation

$$y_i = \mu(t_i) + \varepsilon_i \qquad , \ i = 1,\ldots,n \tag{2.52}$$

avec les ε_i représentant les erreurs non corrélées de moyenne zéro et de variance σ^2. De plus nous considérons que $\mu(t)$ est compris dans l'espace Sobolev $W_2^m[a,b]$. Pour ce cas, à partir du théorème de Taylor, l'équation (2.52) peut être écrite sous la forme suivante

$$y_i = \sum_{j=1}^{m} \theta_j t_i^{j-1} + \mathrm{Res}(t_i) + \varepsilon_i \tag{2.53a}$$

où

$$\mathrm{Res}(t) = \left[(m-1)!\right]^{-1} \int_a^b \mu^{(m)}(t)(t-\xi)_+^{m-1} d\xi \tag{2.53b}$$

Si les $\mathrm{Res}(t_1), \ldots, \mathrm{Res}(t_n)$ sont faibles alors une régression polynômiale semble être une méthode raisonnable pour l'analyse des données. Cependant si ce n'est pas le cas, les régressions polynômiales ne seront pas satisfaisantes et ,alors, un type quelconque de protection contre cette possibilité doit être construit pour pouvoir créer un estimateur à partir de l'équation (2.53).

Une méthode d'estimation qui permet d'obtenir ce type de protection pour les modèles polynômiaux sont les splines de lissages étudiées précédemment. Quand le paramètre de lissage est choisi ceci fait qu'une frontière est estimée sur

l'amplitude des Res(t) dans (2.53b) et cet estimé est considéré dans la création de l'estimateur.

La méthode des splines moindres carrées est une autre façon de pouvoir considérer l'amplitude de Res(t) dans la création d'un estimateur. L'hypothèse de base pour cet estimateur est la suivante, en utilisant une règle de quadrature nous pouvons approximer l'intégrale de l'équation (2.53b) par une somme de la forme

$$\sum_{j=1}^{k} \delta_j (t - \xi_j)_+^{m-1}$$

pour un ensemble de coefficients $\delta_1, \ldots, \delta_n$ et de points $a<\xi_1< \ldots <\xi_k<b$. Donc combinant ceci avec l'approximation polynômiale originale, nous trouvons alors une forme d'estimateur. La forme de l'estimateur est la suivante

$$s(t) = \sum_{j=1}^{m} \theta_j t^{j-1} + \sum_{j=1}^{k} \delta_j (t - \xi_j)_+^{m-1} \qquad (2.54)$$

Ici s est définie comme étant une spline d'ordre m avec les noeuds $\xi_1 \ldots \xi_k$. L'ensemble de telles fonctions , $S^m(\xi_1 \ldots \xi_k)$, consiste en un espace vectoriel de dimension m+k . De plus la fonction $(t - \xi_j)_+$ est définie comme étant

$$\max(0, (t - \xi_j))$$

Ayant des valeurs pour $\lambda=\{\xi_1 \ldots \xi_k\}$, nous pouvons estimer μ en estimant les coefficients de l'équation (2.54). Une méthode accomplissant ceci est d'utiliser les moindres carrés. Disons que nous avons

$$
\begin{aligned}
x_1(t) &= 1 \\
x_2(t) &= t \\
&\vdots \\
x_m(t) &= t^{m-1} \\
x_{m+1}(t) &= (t-\xi_1)_+^{m-1} \\
&\vdots \\
x_{m+k}(t) &= (t-\xi_k)_+^{m-1}
\end{aligned}
\tag{2.55}
$$

et l'ensemble de coefficients

$$\underline{\beta} = (\theta_1, \ldots, \theta_m, \delta_1, \ldots, \delta_k)' \tag{2.56}$$

alors l'estimateur splines moindres carrées de μ est

$$\mu_\lambda = \sum_{j=1}^{m+k} \beta_{\lambda j} x_j(t) \tag{2.57}$$

où $\underline{\beta}_\lambda = (\beta_{\lambda,1}, \ldots, \beta_{\lambda,m+k})$ est un minimum de

$$\frac{\sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m+k} \beta_j x_j(t_i) \right)^2}{n} \tag{2.58}$$

Plus particulièrement , si nous définissons

$$X(\lambda) = \begin{vmatrix} x_1(t_1) & \cdots & x_{m+k}(t_1) \\ \vdots & \ddots & \vdots \\ x_1(t_n) & \cdots & x_{m+k}(t_n) \end{vmatrix} \tag{2.59}$$

alors $\underline{\beta}_\lambda$ est trouvé comme étant une solution aux équations normales

$$X(\lambda)'X(\lambda)\underline{\beta} = X(\lambda)'\underline{y} \tag{2.60}$$

Si nous isolons $\underline{\beta}_\lambda$ dans (2.60) nous trouvons (2.61)

$$\underline{\beta}_\lambda = \left[X(\lambda)'X(\lambda)\right]^{-1}X(\lambda)'\underline{y} \tag{2.61}$$

Il s'en suit qu'à partir de (2.57) et de (2.61), donnant $\lambda=\{\xi_1 \ldots \xi_k\}$, μ_λ est un estimateur linéaire de μ. Ici soulignons que l'ensembe λ est un paramètre de lissage. Le choix de λ se fait à partir du nombre et de la position des noeuds pour les splines.

La sélection de λ se fait selon le besoin de courbure que possède les données. Pour plus de détails concernant les splines moindres carrées voir [EUB88].

Ceci complète la partie concernant les méthodes d'étalonnages. Ce chapitre s'est voulu non pas une étude exhaustive des différentes méthodes d'étalonnage mais voulait souligner celles les plus couramment employées. Le prochain chapitre traite des algorithmes d'étalonnage pour la pression qui ont été construits avec certaines de ces méthodes.

CHAPITRE 3

ALGORITHMES D'ÉTALONNAGE BASÉS SUR LES FONCTIONS SPLINES

Au chapitre précédant nous avons montré différentes méthodes mathématiques permettant la construction d'algorithmes d'étalonnage. Dans ce chapitre nous allons présenter trois algorithmes d'étalonnage pour le système de mesure de pression décrit au chapitre 1. Ces algorithmes sont construits à partir de fonctions splines d'interpolation et de splines de lissage. Ce chapitre comporte quatre parties. La première partie décrit les algorithmes d'étalonnage, la deuxième montre un méthode permettant l'évaluation de ces algorithmes, la troisième partie montre les résultats obtenues pour chaque algorithme et la dernière partie porte sur le choix du meilleur algorithme.

3.1) Description des algorithmes d'étalonnage

Une méthode d'étalonnage doit impliquer la solution de deux problèmes de base [BAR92]:

- choix de la structure des modèles mathématiques pour les relations physiques

  $p,T\cdot v_1$ $p,T\cdot v_2$ $p,T\cdot v_g$

- estimation des paramètres de ces modèles

Le premier problème est du domaine heuristique, tandis que le deuxième peut être résolu par l'utilisation d'algorithmes standards destinés à l'estimation de paramètres. D'après [BAR90b], il fût montré que les modèles de type polynômial ou polinômial inverse sont peu adéquats pour des applications métrologiques à cause de leur rigidité. D'un autre côté, les méthodes d'approximation basées sur des fonctions splines offrent une grande flexibilité [DEB78], [EUB88]. Ceci est nécessaire à l'obtention d'un étalonnage précis. La flexibilité des fonctions splines est présente même lorsqu'elle est appliquée à des caractéristiques statiques complexes.

Les fonctions splines à une variable, telles que décrites au chapitre 2, ont été utilisées pour approximer les caractéristiques $f_1(p;T)$, $f_2(p;T)$ et $f_g(p;T)$ qui sont des fonctions à deux variables : p et T.

Le problème à résoudre consiste à approximer des fonctions à partir de données d'étalonnages reliant la pression à la température et à la tension de sortie d'une fibre. Comme la mesure de la pression nécessite trois fibres différentes (voir cahpitre 2), il nous faut alors approximer trois fonctions différentes. En termes plus généraux, il faut approximer les fonctions G_i de l'équation (3.1) à partir de mesures de tension, de pressions et de températures

$$p_i = G_i(v,T) \qquad \text{pour i=1,2,3,...} \tag{3.1}$$

Pour analyser les différents algorithmes d'étalonnage, nous posons premièrement que les données pour l'étalonnage sont ordonnées de la façon suivante:

$$D_1^{ETAL} = \left\{ p_i^{ETAL} , \left\{ T_j^{ETAL} \; v1_{i,j}^{ETAL} \; v2_{i,j}^{ETAL} \; vg_{i,j}^{ETAL} \,\middle|\, j = 1,\ldots, NP \right\} \,\middle|\, \mathrm{i} = 1,\ldots,\mathrm{NT} \right\} \tag{3.2}$$

avec comme conditions

$$P_{\min} \cong p_1^{ETAL} < p_2^{ETAL} < \ldots < p_{NP}^{ETAL} \cong P_{\max} \tag{3.3}$$

et

$$T_{\min} \cong T_1^{ETAL} < T_2^{ETAL} < \ldots < T_{NT}^{ETAL} \cong T_{\max} \tag{3.4a}$$

$$vg_{1,j}^{ETAL} < vg_{2,j}^{ETAL} < \ldots < vg_{NP,j}^{ETAL} \qquad \text{pour j=1, ... ,NT} \tag{3.4b}$$

De plus nous avons la caractéristique généralisée

$$\mathrm{v}_{k,i,j}^{\mathrm{ETAL}} = G^{-1}\left(p_i^{ETAL} ; T_j^{ETAL} \right) \quad \text{pour k=1,2,g i=1,...,NP et j=1,...,NT} \tag{3.5}$$

Ces données sont sujettes aux erreurs de mesure ΔT_j^{ETAL}, Δp_j^{ETAL}, $\Delta v_{i,j}^{ETAL}$ qui satisfont les énigalités suivantes:

$$\left|\Delta t_j^{ETAL}\right| \le \overline{\Delta t}^{ETAL} , \left|\Delta p_i^{ETAL}\right| \le \overline{\Delta p}^{ETAL} , \left|\Delta v_{i,j}^{ETAL}\right| \le \overline{\Delta v}^{ETAL} \tag{3.6}$$

où $\overline{\Delta t}^{ETAL}$, $\overline{\Delta p}^{ETAL}$, $\overline{\Delta v}^{ETAL}$ sont les erreurs limites absolues caractérisant les mesures de T, p, v1, v2 et vg.

Avant de décrire chacune des procédures, il nous faut introduire certaines notations qui sont utilisées dans la description de certaines procédures. Les notations à définir sont les suivantes:

$[v_{min}, v_{max}]$ valeurs des tensions de sortie (des capteurs C1 C2) effectivement utilisées pour la reconstruction de la mesurande

k indice des capteurs (k=1,2,G);

m indice des fragments monotones des caractéristiques statiques des capteurs.

Q^k nombre de fragments monotones de la caractériatique du capteur Ck .

$p=f_{k,m}(v,T)$ inverse du $m^{ième}$ fragment monotone de la caractéristique statique du capteur Ck.

CC capteur de référence choisi pour la reconstruction fine du mesurande (C1 ou C2).

Pour la première procédure nous considérons un système de trois capteurs C1, C2 et Cg dont les caractéristiques statiques sont répétées à la figure 3.1

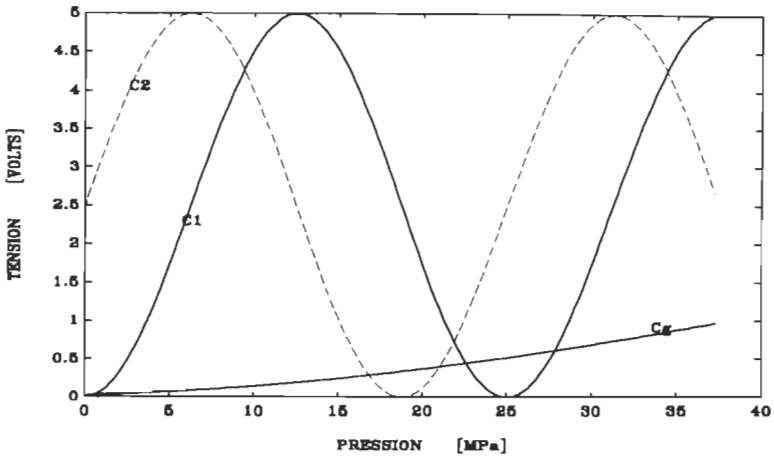

Figure 3.1 caractéristique des trois capteurs de pression pour une certaine température

Pour plus de simplicité nous définissons la fonction spline Spl(x;$\underline{q}$) comme:

$$\mathrm{Spl}(x;\underline{q}) = C1_n + C2_n(x - x_n) + C3_n(x - x_n)^2 + \ldots + CM_n(x - x_n)^{M-1} \qquad (3.7a)$$

avec des paramètres

$$\underline{q} = \left[C1_1, C2_1, C3_1, \ldots, CM_1, \ldots, C1_{N-1}, C2_{N-1}, C3_{N-1}, \ldots, CM_{N-1}\right]^T \qquad (3.7b)$$

Cette spline peut être une spline d'interpolation où une spline de lissage. Les procédures seront décrites à partir de cette fonction générale.

Qualitativement la première procédure PCR1 peut être expliquée comme suit. Premièrement nous choisissons le capteur de référence C1 ou C2 qui minimise l'erreur du résultat final de mesure. Par la suite nous fabriquons un ensemble de fonctions splines approximant la caractéristique statique du capteur

choisi et du guide . Graphiquement ceci peut se représenter par les figures 3.2 et 3.3. Nous voyons que pour une température mesurée de 24 °C, nous avons une droite qui coupe plusieurs courbes, ceci autant pour le guide que pour le capteur choisit.

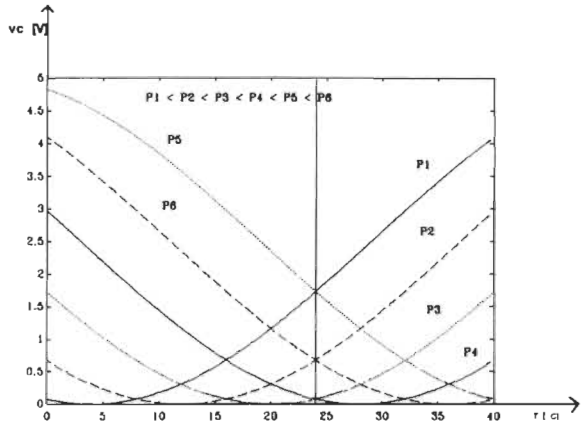

Figure 3.2 Tension en fonction de la température pour le capteur choisit

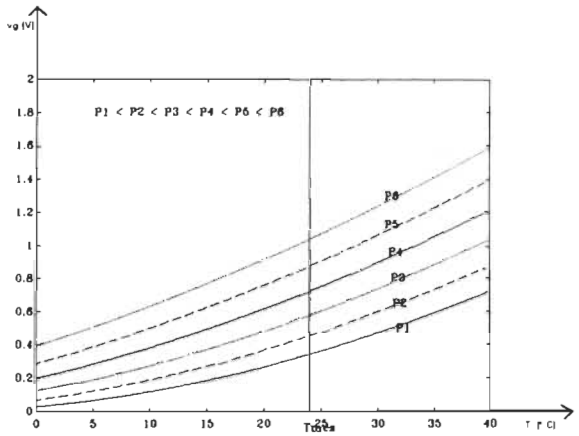

Figure 3.3 Tension en fonction de la température pour le guide

Les points d'intersection de la droite avec les courbes sont interpolés à partir des splines créées auparavant. Après cette étape, nous avons, pour le guide et pour le capteur choisit, des ensembles de tensions et de pressions. Les figures 3.4 et 3.5 montrent la répartition de ces points.

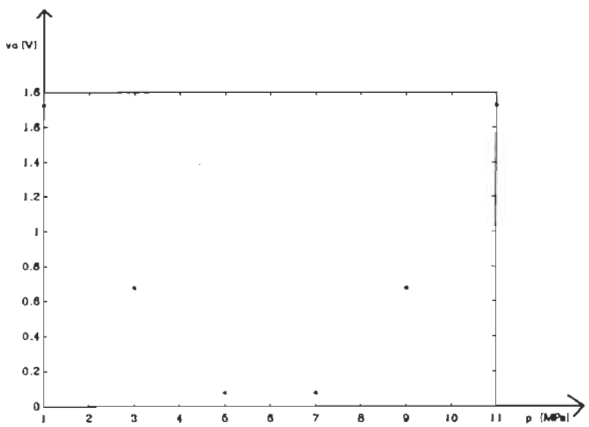

Figure 3.4 Tension en fonction de la pression pour le capteur choisit

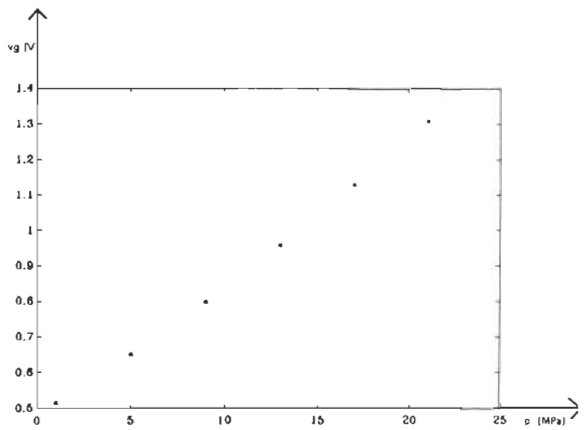

Figure 3.5 Tension en fonction de la pression pour le guide

Dans le cas particulier montré aux figures 3.2 à 3.5 si nous avons des mesures de tensions pour le capteur choisi de 1.1 volt et pour le guide de 0.8 volt, nous remarquons que pour la tension mesurée à la sortie du guide correspond une seule valeur de la pression. Par contre pour la tension mesurée à la sortie du capteur de référence correspondent deux valeurs de pression. Donc le guide ici sert à savoir laquelle de ces deux valeurs est adéquate.

Donc la première procédure d'étalonnage PCR1 est décrite comme suit:

1. choisir un capteur entre C1 et C2. Ce choix est basé sur la valeur de la pente de chaque capteur pour une pression locale. Le capteur possédant la pente la plus élevée sera choisit. Cette étape fournit la tension mesurée du capteur choisi vc_m, la tension mesurée du guide vg_m et la température mesurée t.

2. pour chaque valeur de référence p_i^{ETAL} (i=1, . . . , NP) il faut déterminer les vecteurs de paramètres $\underline{q}_i$ et $\underline{qg}_i$ des fonctions

$$vc = Spl(t, \underline{q}_i) \tag{3.8}$$

$$vg = Spl(t; \underline{qg}_i) \tag{3.9}$$

approximant respectivement

$$\left\{ t_j^{ETAL}, vc_{i,j}^{ETAL} \;\middle|\; j = 1, \ldots, NT \right\} \tag{3.10a}$$

$$\left\{ t_j^{ETAL}, vg_{i,j}^{ETAL} \;\middle|\; j = 1, \ldots, NT \right\} \tag{3.10b}$$

3. pour la valeur mesurée de $\hat{t}$, il faut calculer les séquences de valeurs

$$\hat{vc}_i = Spl(\hat{t}, \underline{q}_i) \qquad \text{pour i=1, . . . , NP} \tag{3.11}$$

$$\hat{vg}_i = Spl(\hat{t}; \underline{qg}_i) \qquad \text{pour i=1, . . ., NP} \tag{3.12}$$

et les utiliser pour déterminer les vecteurs de paramètres $\underline{q}(\hat{v}\,c)$ et $\underline{q}(\hat{v}\,g)$ de d'autres fonctions splines

$$\hat{vc} = Spl(p, \underline{q}(\hat{vc})) \tag{3.13}$$

$$\hat{vg} = Spl(p, \underline{q}(\hat{vg})) \tag{3.14}$$

approximant respectivement les séquences de points

$$\left\{ p_i^{ETAL}, \hat{vc}_i \middle| \; \mathrm{i} = 1, \ldots, \mathrm{NP} \right\} \tag{3.15a}$$

$$\left\{ p_i^{ETAL}, \hat{vg}_i \middle| \; \mathrm{i} = 1, \ldots, \mathrm{NP} \right\} \tag{3.15b}$$

4. calculer la valeur approximative de la pression mesurée $\hat{p}$ à l'aide de vg_m

$$\hat{p} = \arg_p \left\{ Spl(p, \underline{q}(\hat{vg})) = vg_m \right\} \tag{3.16}$$

5. calculer la valeur de $\hat{p}$ à l'aide de vc_m.

$$\hat{p} = \arg_p \left\{ Spl(p, \underline{q}(\hat{vc})) = vc_m \right\} \tag{3.17}$$

Comme la courbe de ce capteur est non monotone , il existe plusieurs valeurs satisfaisant (3.17). La discrimination de ces valeurs se fait à l'aide de l'expression (3.16) pour ainsi aboutir à une valeur qui représente plus fidèlement la pression réelle.

La procédure PCR2 est sensiblement pareille à la procédure PCR1. D'où pour cette procédure seulement la forme algorithmique est présentée. L'ensemble des programmes effectuant les deux procédures sont donnés en annexe.

La procédure PCR2 transforme les caractéristiques statiques des deux capteurs de référence en une seule monotone. Plus précisément la caractéristique de chacun des capteurs est transformée en une caractéristique monotone (figure 3.6) et à partir de ces deux caractéristiques monotones, une troisième est créée en utilisant les régions des deux caractéristiques servant au calcul de la pression (figure 3.7). Cette opération est effectuée pour chaque valeur de la température T_j^{ETAL}.

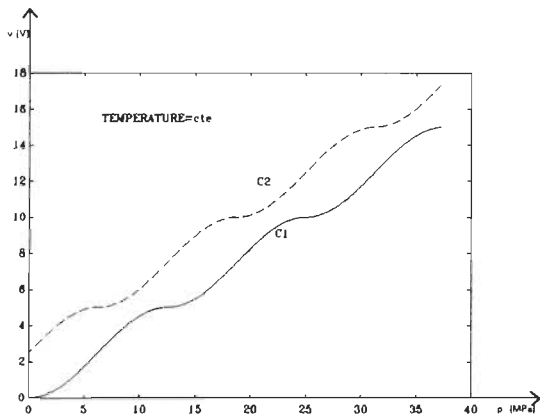

Figure 3.6 Caractéristiques monotones de deux capteurs de référence

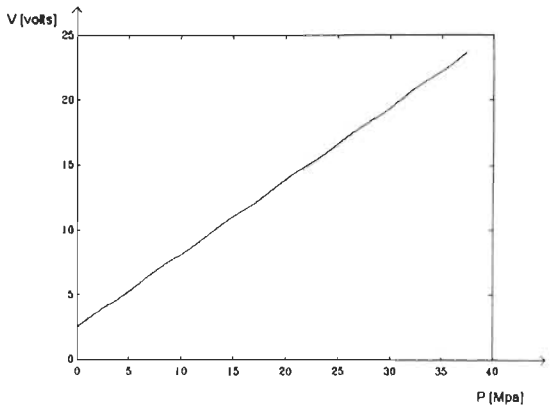

Figure 3.7 Caractéristique monotone combinée des deux capteurs

Ici le guide sert à connaître la plage de pression dans laquelle se situe la valeur exacte. Plus précisément le guide sert à calculer un décalage qu'il faut ajouter à la tension fournie par le capteur de référence.

Donc pour la caractéristique combinée nous avons un ensemble de valeurs de tensions d'étalonnage appelées vk qui proviennent de C1 ou C2 et qui ont la forme suivante:

$$vk_{1,j}^{ETAL} < vk_{2,j}^{ETAL} < \ldots < vk_{NP,j}^{ETAL} \qquad \text{pour j=1, . . . ,NT} \tag{3.18}$$

La procédure PCR2 peut donc être décrite comme suit:

1. choisir un capteur entre C1 et C2. Ce choix est basé sur la valeur de la pente de chaque capteur pour une pression locale. Le capteur possédant la pente la plus élevée sera choisi. Cette étape fournit la tension mesurée du capteur choisi vc_m, la tension mesurée du guide vg_m et la température mesurée t.

2. pour chaque valeur de mesure p_i^{ETAL} (i=1, . . . , NP), il faut déterminer les vecteurs de paramètres $\underline{q}_i$ et $\underline{\underline{qg}}_i$ des fonctions

$$vk = Spl(t, \underline{q}_i) \tag{3.19}$$

$$vg = Spl(t; \underline{\underline{qg}}_i) \tag{3.20}$$

approximant respectivement

$$\left\{ t_j^{ETAL}, vk_{i,j}^{ETAL} \;\middle|\; \mathrm{j} = 1, \ldots, \mathrm{NT} \right\} \tag{3.21a}$$

$$\left\{ t_j^{ETAL}, vg_{i,j}^{ETAL} \;\middle|\; \mathrm{j} = 1, \ldots, \mathrm{NT} \right\} \tag{3.21b}$$

3. pour la mesure de $\hat{t}$, les séquences de valeurs

$$\hat{vk}_i = Spl(\hat{t}, \underline{q}_i) \qquad \text{pour i=1, . . . , NP} \tag{3.22}$$

$$\hat{vg}_i = Spl(\hat{t}; \underline{\underline{qg}}_i) \qquad \text{pour i=1, . . ., NP} \tag{3.23}$$

sont calculées et utilisées pour déterminer les vecteurs de paramètre $\underline{q}(\hat{v}\,\mathrm{k})$ et $\underline{q}(\hat{v}\,\mathrm{g})$ de d'autres fonctions splines

$$\hat{vk} = Spl(p, \underline{q}(\hat{vk})) \tag{3.24}$$

$$\hat{vg} = Spl(p, \underline{q}(\hat{vg})) \tag{3.25}$$

approximant respectivement les séquences de points

$$\left\{ p_i^{ETAL}, \hat{vk}_i \middle| \, \mathrm{i} = 1, \ldots, \mathrm{NP} \right\} \tag{3.26a}$$

$$\left\{ p_i^{ETAL}, \hat{vg}_i \middle| \, \mathrm{i} = 1, \ldots, \mathrm{NP} \right\} \tag{3.26b}$$

4. calculer la valeur approximative de $\hat{p}$ à l'aide de vg_m

$$\hat{p} = \arg_p \left\{ Spl(p, \underline{q}(\hat{vg})) = vg_m \right\} \tag{3.27}$$

5. à l'aide de $\hat{p}$ estimé par le guide et connaissant le capteur qui a été choisi, calculer le décalage à ajouter à la tension mesurée vc_m

$$vc_m = vc_m + dec(\hat{p}) \tag{3.28}$$

où $dec(\hat{p})$ est le décalage fonction de $\hat{p}$.

6. calculer $\hat{p}$ à l'aide de vc_m déterminé à 5.

$$\hat{p} = \arg_p \left\{ Spl(p, \underline{q}(\hat{vk})) = vc_m \right\} \tag{3.29}$$

L'intérêt de cette procédure est dans la courbe quasi droite de la caractéristique combinée. Par contre vue que les données servant à l'étalonnage doivent être traitées pour créer la caractéristique combinée, il en résulte une plus faible maniabilité. Cet algorithme est très dépendant des données d'étalonnage. Donc pour remédier à ce problème une troisième procédure est développée. Elle est moins dépendante des données d'étalonnage que PCR2. Elle est un compromis entre PCR1 et PCR2.

La procédure PCR3 consiste à utiliser des fragments de la caractéristique des capteurs pour reconstruire le mesurande. Nous allons expliquer plus en détail ce procédé.

Pour une meilleur compréhension de la procédure PCR3 nous allons premièrement donner une expliquation qualitative. La première étape consiste à choisir le capteur qui sert à reconstruire la pression. Ensuite ont reconstruit la pression à l'aide du guide. Connaisant une première approximation sur la pression , il nous est possible de construire, à partir de fragments de caractéristiques, un ensemble de courbes. Une fois cet ensemble construit, à l'aide de la tension mesurée provenant d'un des capteurs de précision on reconstruit la pression. Les figures 3.8 et 3.9 montrent ce procédé.

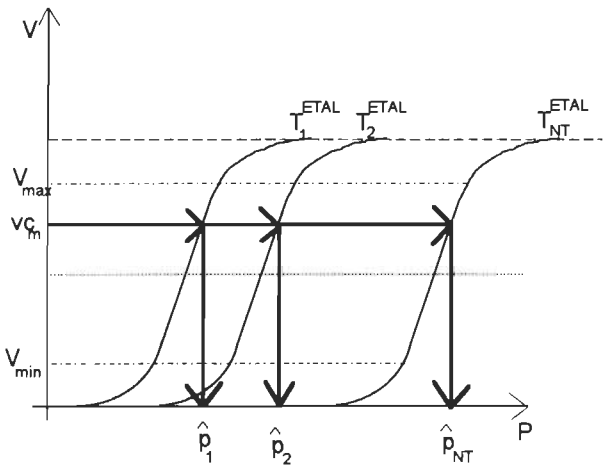

Figure 3.8 Graphique représentant différents fragments de caractéristique pour différentes températures

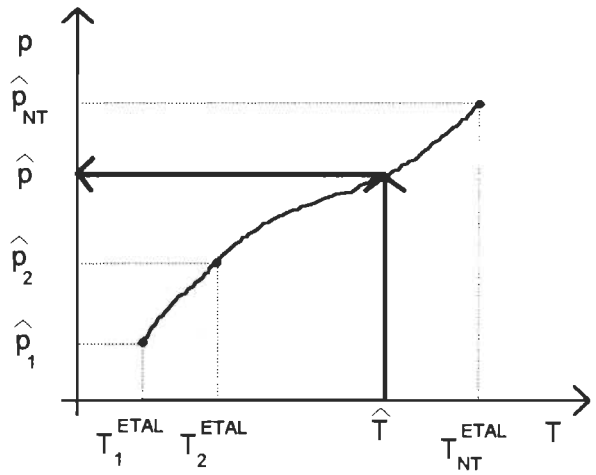

Figure 3.9 Graphique représentant la reconstruction de la pression à partir de la température

L'explication mathématique de la présentation qualitative peut se décrire comme suit. Premièrement il nous faut déterminer les données d'étalonnages. Ces données sont définies par l'équation (3.30).

$$D_k^{ETAL} = \bigcup_{m=1}^{Q_k} D_{k,m}^{ETAL} \qquad \text{pour k=1,2,G} \qquad (3.30)$$

où le sous-ensemble $D_{k,m}^{ETAL}$ représente un fragment de la caractéristique du capteur C_k par $p=f_{k,m}(v,T)$. Plus particulièrement $D_{k,m}^{ETAL}$ est définie par (3.31).

$$D_{k,m}^{ETAL} = \left\{ T_{k,m,j}^{ETAL}, \left\{ v_{k,m,i,j}^{ETAL} \middle| j = 1,\ldots,NV_k \right\} \middle| i = 1,\ldots,NT_k \right\} \qquad (3.31)$$

Si la plage de pression est définie par $[p_{min}, p_{max}]$, il faut, pour la procédure 3, que les conditions suivantes soient respectées.

$$\begin{aligned} \max\left\{ f_{k,1}(v_{min}, T_{k,1,i}^{ETAL}) \middle| i = 1,\ldots,NT_k \right\} &\le p_{min} \\ \min\left\{ f_{k,Q_k}(v_{max}, T_{k,Q_k,j}^{ETAL} \middle| i = 1,\ldots,NT_k \right\} &\ge p_{max} \end{aligned} \qquad (3.32)$$

pour k=1,2,G.

Donc à partir des ensembles de données définie par (3.30) et (3.31), ont peut présenter la procédure 3 (PCR3).

1. choisir un capteur entre C1 et C2. Ce choix est basé sur la valeur de la pente de chaque capteur pour une pression locale. Le capteur possédant la pente la plus élevée sera choisi. Cette étape fournit la

tension mesurée du capteur choisi vc_m, la tension mesurée du guide vg_m et la température mesurée T_m.

2. pour chaque valeur de mesure T_i^{ETAL} , avec i=1,. . . ,NT, les vecteurs de paramètres $\underline{q}_{i,m}$ et $\underline{qg}_i$ sont déterminés à partir des fonctions

$$p = Spl(vc, \underline{q}_{i,m}) \tag{3.33}$$

$$p = Spl(vg; \underline{qg}_i) \tag{3.34}$$

approximant respectivement

$$\left\{ vc_{m,i,j}^{ETAL}, p_{m,i,j}^{ETAL} \;\middle|\; \mathrm{j} = 1,\ldots,\mathrm{NV}\ ,\ \mathrm{m} = 1,\ldots,\mathrm{Q_k} \right\} \tag{3.35a}$$

$$\left\{ vg_{i,j}^{ETAL}, p_{i,j}^{ETAL} \;\middle|\; \mathrm{j} = 1,\ldots,\mathrm{NV} \right\} \tag{3.35b}$$

3. pour la mesure de vg_m , la séquence de valeurs

$$\hat{p}g_i = Spl(vg_m, \underline{qg}_i) \qquad \text{pour i=1, . . . , NP} \tag{3.36}$$

est calculée et utilisée pour déterminer le vecteur de paramètres $\underline{q}(\hat{p}g)$ d'une autre fonction spline

$$\hat{p}g = Spl(T, \underline{q}(\hat{p}g)) \tag{3.37}$$

approximant la séquence de points

$$\left\{ T_i^{ETAL}, \hat{p}g_i \;\middle|\; \mathrm{i} = 1,\ldots,\mathrm{NT} \right\} \tag{3.38}$$

4. $\hat{p}$ approximé par le guide est donné par

$$\hat{p} = Spl(T, \underline{q}(\hat{p}g)) \tag{3.39}$$

5. pour la mesure de vc_m, et connaissant $\hat{p}$ approximé par l'étape 4, nous déterminons la séquence de valeurs

$$\hat{pc}_i = Spl(vc_m, \underline{q}_{i,k}) \qquad \text{pour i=1, . . . ,NT et } 1 \le k \le Q_k \quad (3.40)$$

où le fragment k à utiliser est déterminé par la connaissance de $\hat{p}$ approximé par le guide. Ici k est une variable. Les valeurs de $\hat{pc}_i$ servent à déterminer le vecteur de paramètres $\underline{q}(\hat{pc})$ d'une fonction spline

$$\hat{pc} = Spl(T, \underline{q}(\hat{pc})) \quad (3.41)$$

approximant les points

$$\left\{ T_i^{ETAL}, \hat{pc}_i \middle| \; i = 1, \ldots, NT \right\} \quad (3.42)$$

6. finalement la pression $\hat{p}$ approximée par le capteur choisi est donnée par

$$\hat{p} = Spl(T_m, \underline{q}(\hat{pc})) \quad (3.43)$$

3.2) Description de la méthode d'évaluation des algorithmes

Les procédures PCR1, PCR2 et PCR3 ont été testées avec des données synthétiques générées à partir de la formule (3.44).

$$v = V_0 \cos^2\left[(Ap + BT + C)\pi\right] \quad (3.44)$$

avec les valeurs des paramètres A, B, C et V_0 correspondant aux capteurs C1, C2 et Cg données dans la table 1. Comme montré dans [BAR90a] , cette formule est un modèle qui idéalise le capteur de pression à fibre optique.

TABLE 1. Valeur des paramètres du modèle (3.44) pour les capteurs C1, C2, Cg

	A	B	C	V_0
C1	1/25	1/100	-1/4	5.0
C2	1/25	1/100	1/2	5.0
Cg	1/300	1/400	25/48	5.0

Les données exactes

$$\left\{\dot{p}_i, \left\{\dot{T}_{i,j}, \dot{v}_{1,i,j}, \dot{v}_{2,i,j}, \dot{v}_{G,i,j} \middle| \, \mathrm{j} = 1, \ldots, \mathrm{NT}\right\} \middle| \, i = 1, \ldots, NP\right\} \tag{3.45}$$

simulées à partir de (3.44) fûrent perturbées par l'addition de nombres pseudo-aléatoires { Δp_i }, { $\Delta T_{i,j}$ }, { $\Delta v_{1,i,j}$ }, { $\Delta v_{2,i,j}$ }, { $\Delta v_{g,i,j}$ } distribuées uniformément autour de zéro. Ceci fût réalisé pour introduire l'effet des erreurs de mesure lors de l'étalonnage , soit

$$
\begin{aligned}
p_i^{ETAL} &= \dot{p}_i + \Delta p_i & \left|\Delta \mathrm{p_i}\right| &\le \overline{\Delta} p^{ETAL} \\
T_{i,j}^{ETAL} &= \dot{T}_{i,j} + \Delta T_{i,j} & \left|\Delta \mathrm{T_{i,j}}\right| &\le \overline{\Delta} T^{ETAL} \\
v_{1,i,j}^{ETAL} &= \dot{v}_{1,i,j} + \Delta v_{1,i,j} & \left|\Delta \mathrm{v_{1,i,j}}\right| &\le \overline{\Delta} v^{ETAL} \\
v_{2,i,j}^{ETAL} &= \dot{v}_{2,i,j} + \Delta v_{2,i,j} & \left|\Delta \mathrm{v_{2,i,j}}\right| &\le \overline{\Delta} v^{ETAL} \\
v_{G,i,j}^{ETAL} &= \dot{v}_{G,i,j} + \Delta v_{G,i,j} & \left|\Delta \mathrm{v_{G,i,j}}\right| &\le \overline{\Delta} v^{ETAL}
\end{aligned}
$$

où $\overline{\Delta} p^{ETAL}$, $\overline{\Delta} T^{ETAL}$ et $\overline{\Delta} v^{ETAL}$ sont les erreurs limites absolues des mesures de pression, de la tension et de la température.

Pour valider les procédures PCR1, PCR2 et PCR3, un ensemble de données d'évaluation est créé

$$
\left\{ \dot{p}_i^{VAL}, \left\{ \dot{T}_{i,j}^{VAL}, \dot{v}_{1,i,j}^{VAL}, \dot{v}_{2,i,j}^{VAL}, \dot{v}_{G,i,j}^{VAL} \middle| j = 1,\ldots 3NT-3 \right\} \middle| i = 1,\ldots,3NP-3 \right\} \qquad (3.46)
$$

où les valeurs de pression et de températures sont choisies approximativement égales à

$$
0.75\dot{p}_i^{ETAL} + 0.25\dot{p}_{i+1}^{ETAL};\ \ 0.5\dot{p}_i^{ETAL} + 0.5\dot{p}_{i+1}^{ETAL};\ 0.25\dot{p}_i^{ETAL} + 0.75\dot{p}_{i+1}^{ETAL};
$$

$$
0.75\dot{T}_i^{ETAL} + 0.25\dot{T}_{i+1}^{ETAL};\ \ 0.5\dot{T}_i^{ETAL} + 0.5\dot{T}_{i+1}^{ETAL};\ 0.25\dot{T}_i^{ETAL} + 0.75\dot{T}_{i+1}^{ETAL};
$$

Un tel choix de points de mesures semble garantir le pire cas possible pour l'évaluation des algorithmes [BAR92]. Donc l'évaluation des procédures se fait par la méthodologie suivante:

1. Pour les quartets $\left\langle \dot{T}_{i,j}^{VAL}, \dot{v}_{1,i,j}^{VAL}, \dot{v}_{2,i,j}^{VAL}, \dot{v}_{G,i,j}^{VAL} \right\rangle$ les estimées $\hat{p}_{i,j}^{VAL}$ des pression $\dot{p}_i^{VAL}$ sont calculées par la procédure étudiée. De plus les erreurs limites locales d'étalonnage sont calculées.

$$\overline{\Delta}\hat{p}_{i,j} = \left| \hat{p}_{i,j}^{VAL} - \dot{p}_i^{VAL} \right| \tag{3.47}$$

2. L'erreur relative limite est déterminée selon la formule

$$\overline{\delta}\hat{p} = \max\left\{ \left| \frac{\overline{\Delta}\hat{p}_{i,j}}{\dot{p}_i^{VAL}} \right| \; i = 1,\ldots,3NP-3,\; j = 1,\ldots,3NT-3 \right\} \tag{3.48}$$

Le choix de la meilleure procédure se fera selon ces deux critères. Donc une évaluation des trois procédures est présentée dans la prochaine section.

3.3) Résultats obtenus pour les procédures PCR1, PCR2 et PCR3

Nous allons présenter les résultats pour chaque algorithmes décrits précédemment. Les gammes de pressions et de températures sont comprises à l'intérieur des limites suivantes:

p_{min}=1 MPa

T_{min}=0 °C

p_{max}=26 MPa

T_{max}=30 °C

Pour les procédures PCR1, PCR2 et PCR3 les simulations fûrent effectuées pour différentes valeurs d'erreurs sur les tensions, pressions et températures. De plus différents types de splines fûrent utilisées pour chacune de ces erreurs. Vue la grande quantité de résultats obtenus, seul les résultats les plus significatifs sont présentés aux figures 3.10 à 3.37.

Les résultats montrés aux figures 3.10 à 3.13 sont calculées en utilisant des splines cubiques d'interpolation. D'où les paramètres de la fonction $Spl(x;\underline{q})$

$$Spl(x;\underline{q}) = A_n + B_n(x - x_n) + C_n(x - x_n)^2 + D_n(x - x_n)^3 \qquad (3.49)$$

sont définies. De plus deux paramètres libres B_0 et B_n sont approximés à partir d'une dérivée analytique d'un polynôme de Lagrange de degré trois, au lieu d'être posés à zéro.

Les résultats montrés aux figures 3.14 à 3.17 sont calculées en utilisant des splines polynômiales (voir chap. 2) et les résultats montrés aux figures 3.18 à 3.21 utilisent des splines de lissage par la méthode de Reinsch [REI67].

Les erreurs introduites pour la température, pression et les tensions sont supposées suivre une loi uniforme et les simulations sont faites sur HPBASIC et MATLAB.

Les quatres premières figures présentent les résultats obtenues à partir de PCR1 et des splines cubiques d'interpolations

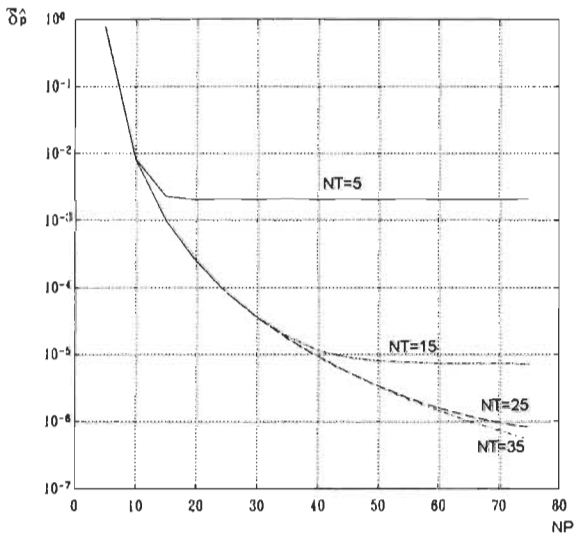

Figure 3.10 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR1: Δp=0.0 MPa ΔT=0.0 °C, Δv=0.0 V.

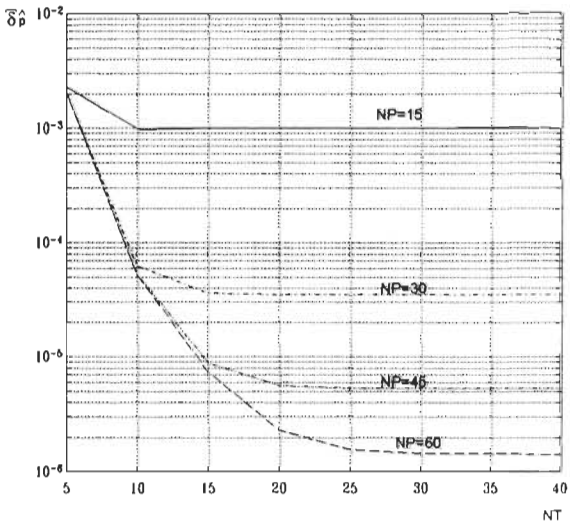

Figure 3.11 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR1: Δp=0.0 MPa ΔT=0.0 °C, Δv=0.0 V.

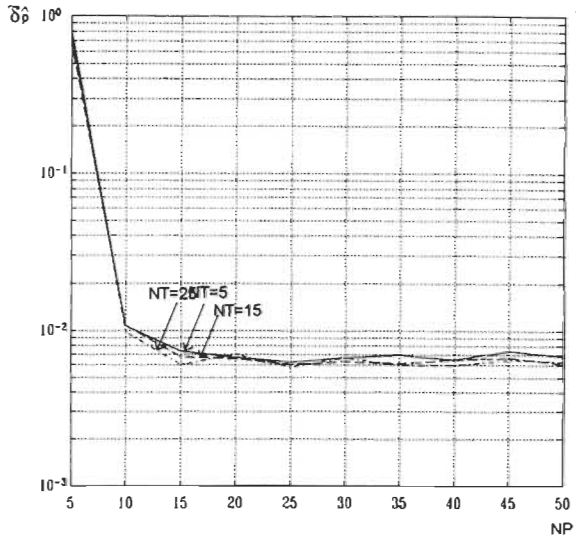

Figure 3.12 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR1: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

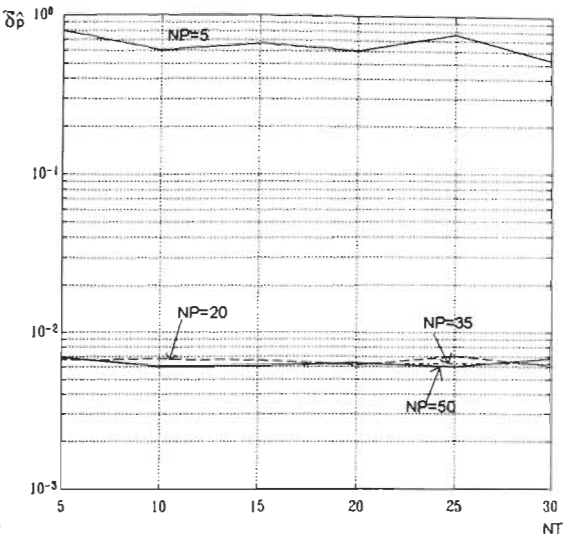

Figure 3.13 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR1: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

Les quatres prochaines figures présentent les résultats obtenus à partir de PCR1 et des splines polynômiales.

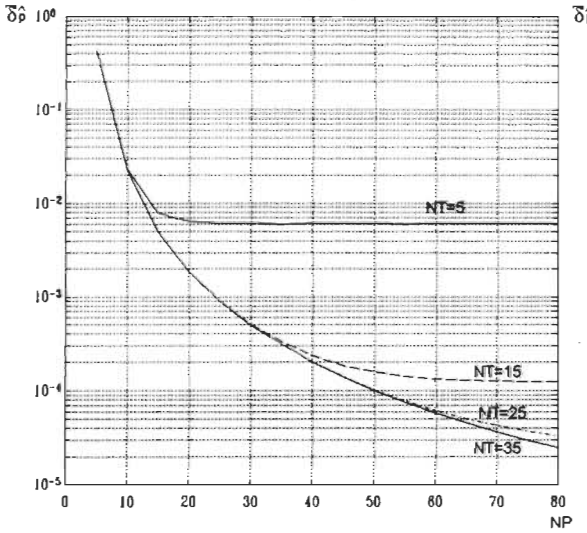

Figure 3.14 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR1: Δp=0.0 MPa ΔT=0.0 °C, Δv=0.0 V.

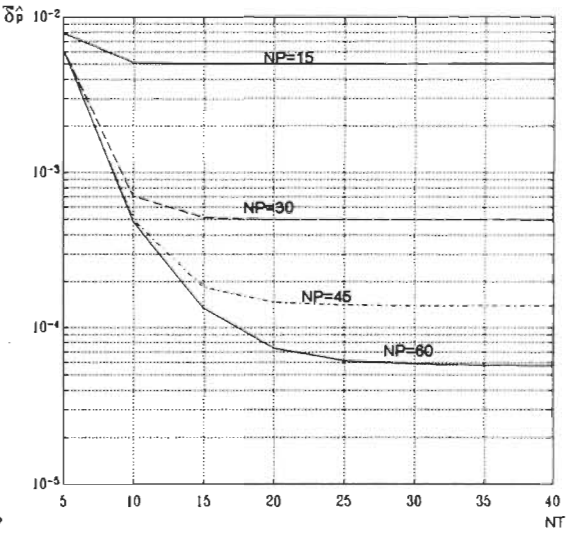

Figure 3.15 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR1: Δp=0.0 MPa ΔT=0.0 °C, Δv=0.0 V.

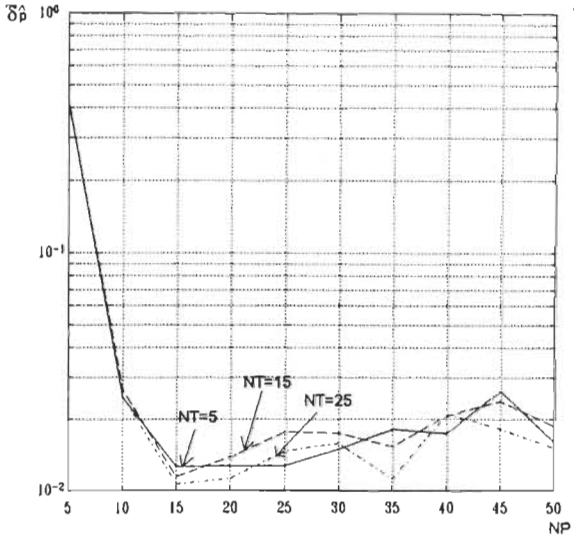

Figure 3.16 Dépendance de l'erreur relative limite en fonction du nombre de de données (NP) utilisées pour l'étalonnage de PCR1: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

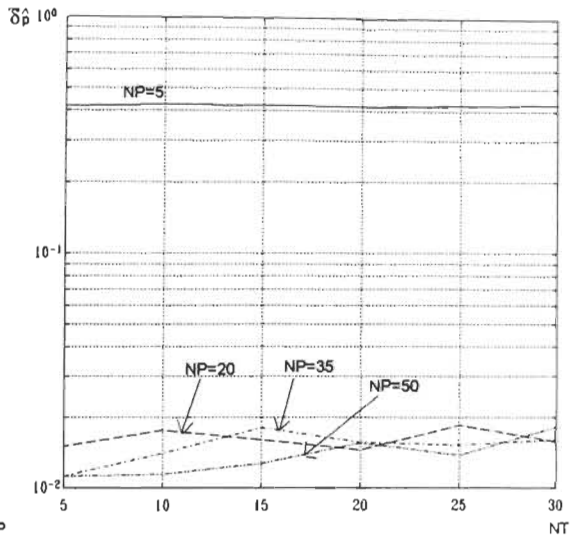

Figure 3.17 Dépendance de l'erreur relative limite en fonction du nombre de données (NT) utilisées pour l'étalonnage de PCR1: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

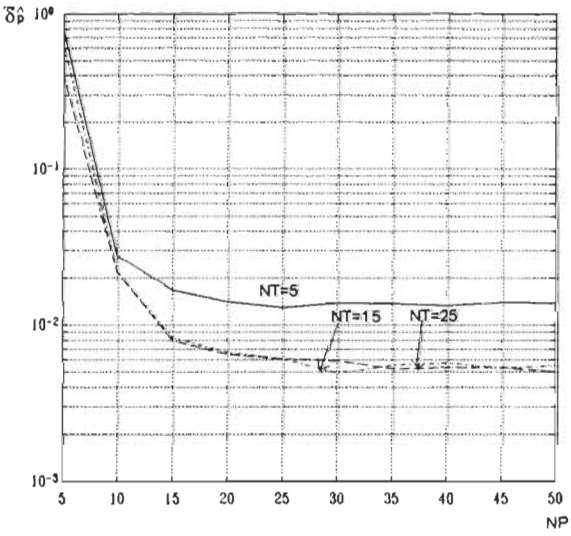

Figure 3.18 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR1: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

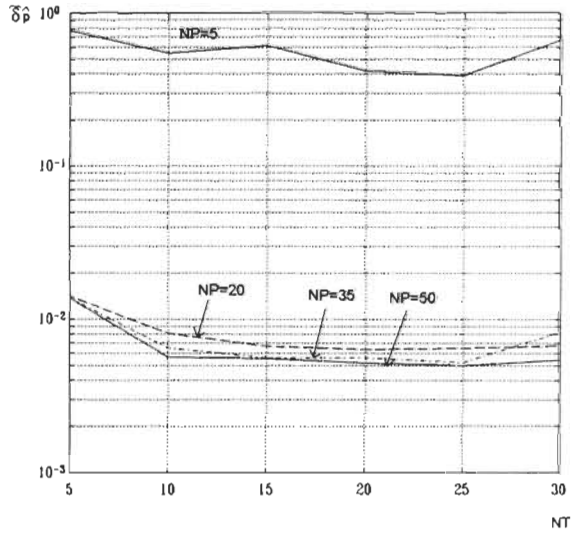

Figure 3.19 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR1: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

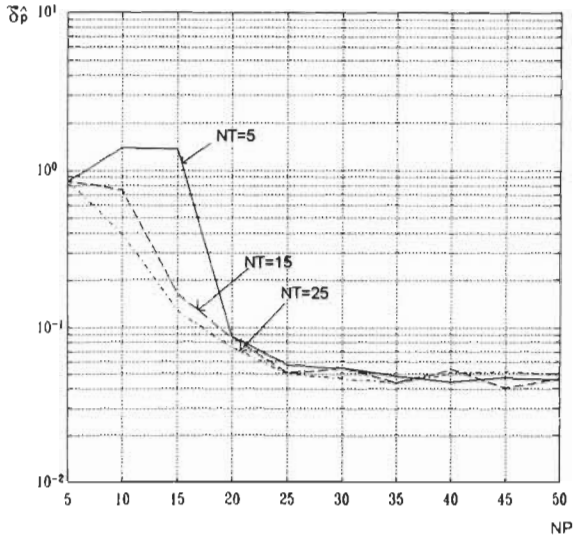

Figure 3.20 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR1: Δp=0.05 MPa ΔT=0.05 °C, Δv=0.002 V.

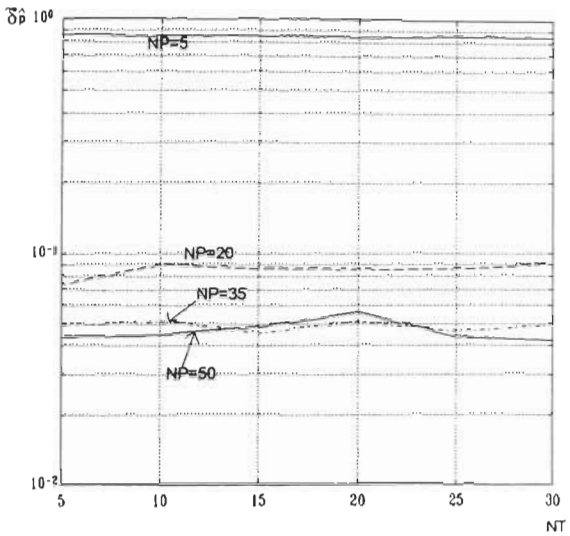

Figure 3.21 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR1: Δp=0.05 MPa ΔT=0.05 °C, Δv=0.002 V.

D'après les résultats obtenues à partir de PCR1 avec les données synthétiques, nous remarquons que l'emploi de splines cubiques donnent les meilleurs résultats. Il est par ce fait possible de dire que l'utilisation des splines de lissage doivent être de type cubique.

Les figures 3.22 à 3.33 montrent les résultats obtenus lorsqu'on utilise PCR2 avec différents type de splines. Les figures 3.22 à 3.25 montrent l'emploi de PCR2 avec des splines cubiques d'interpolation.

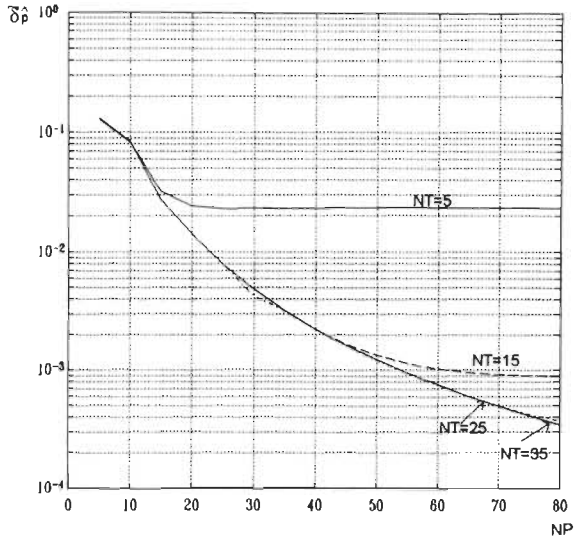

Figure 3.22 Dépendance de l'erreur relative limite en fonction du nombre de pressions (NP) utilisées pour l'étalonnage par PCR2: Δp=0.0 MPa ΔT=0.0 °C, Δv=0.0 V.

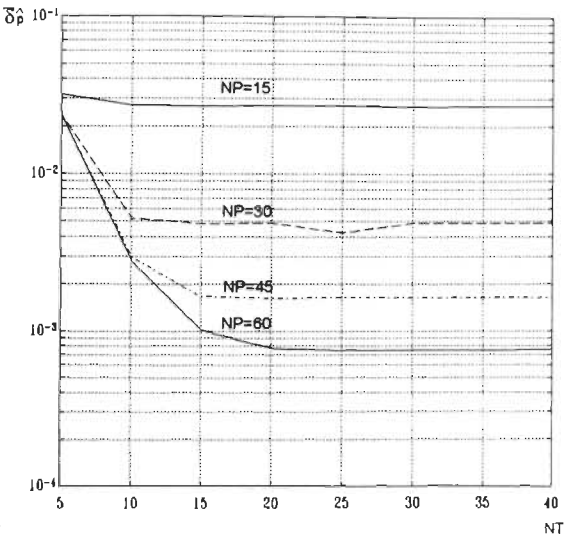

Figure 3.23 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR2: Δp=0.0 MPa ΔT=0.0 °C, Δv=0.0 V.

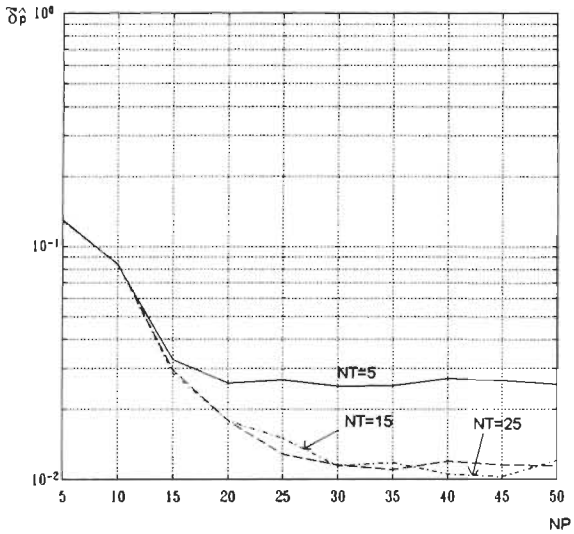

Figure 3.24 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR2: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

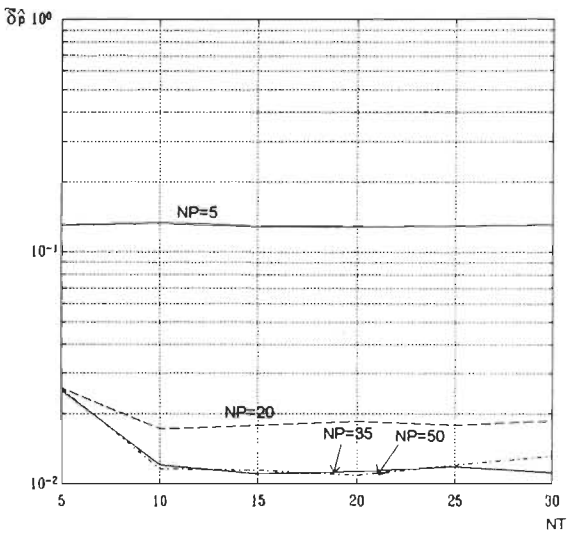

Figure 3.25 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR2: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

Les figurent 3.26 à 3.29 présentent les résultats obtenus à partir de PCR2 et des splines polynômiales.

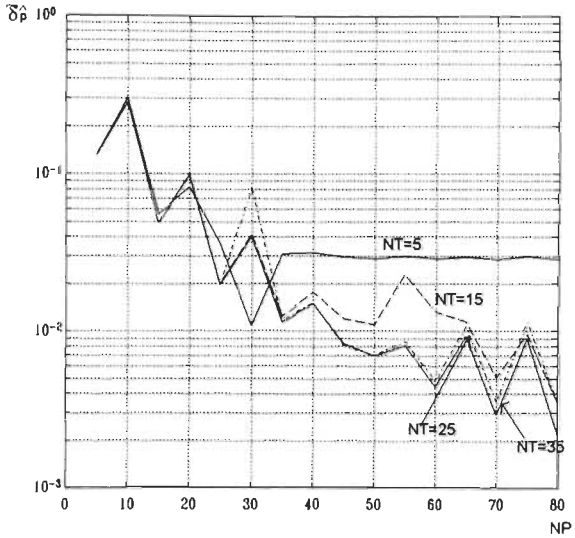

Figure 3.26 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR2: Δp=0.0 MPa ΔT=0.0 °C, Δv=0.0 V.

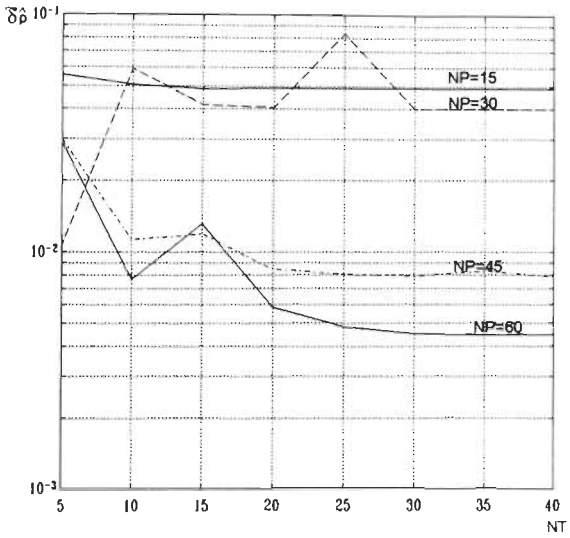

Figure 3.27 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR2: Δp=0.0 MPa ΔT=0.0 °C, Δv=0.0 V.

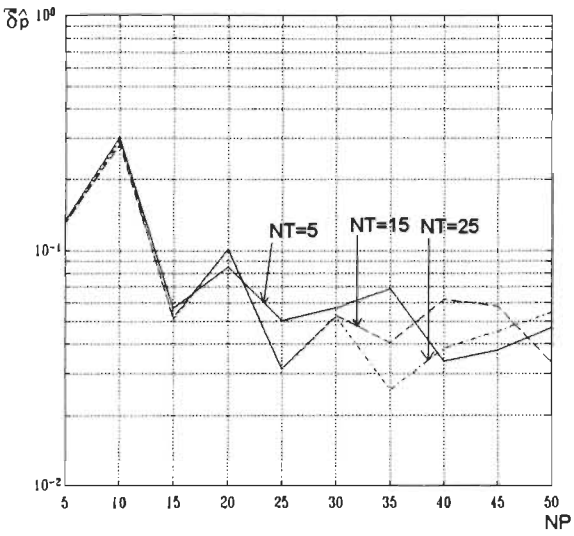

Figure 3.28 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR2: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

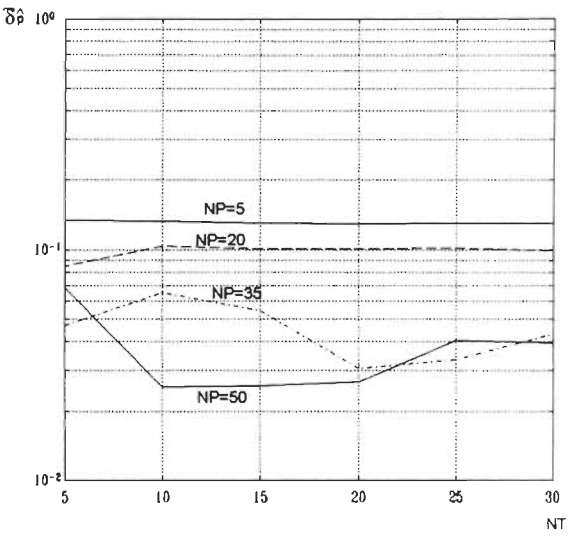

Figure 3.29 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR2: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

Les figures 3.30 à 3.33 montrent les résultats obtenus lorsqu'on utilise PCR2 avec des splines cubiques de lissage.

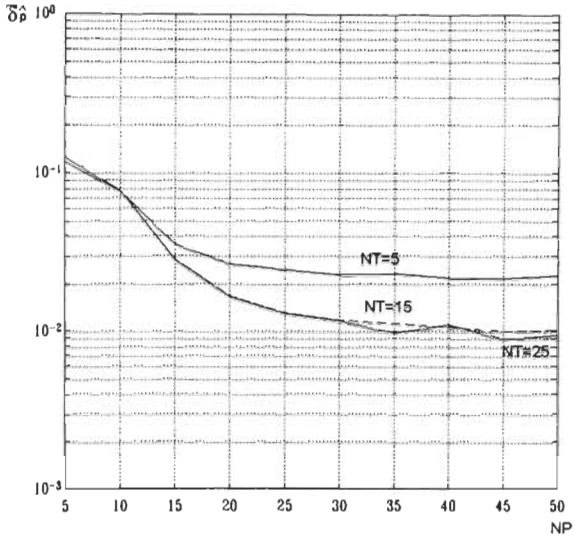

Figure 3.30 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR2: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

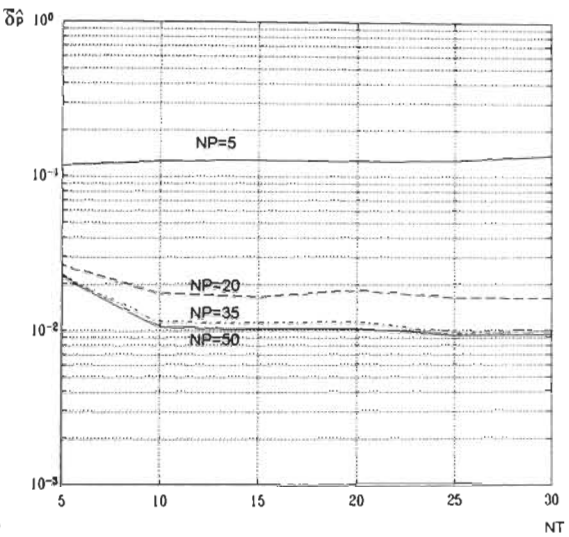

Figure 3.31 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR2: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

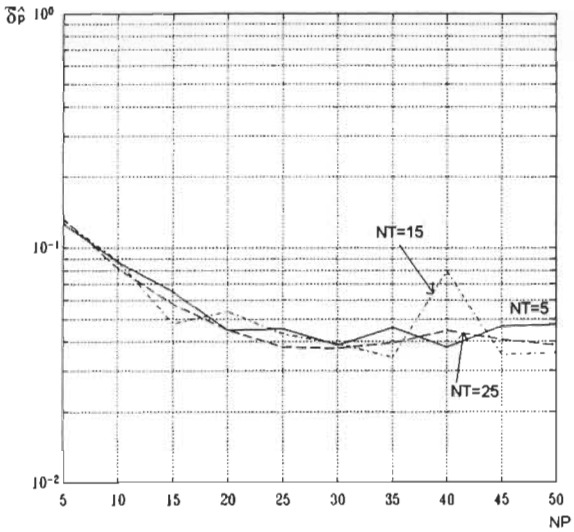

Figure 3.32 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR2: Δp=0.05 MPa ΔT=0.05 °C, Δv=0.002 V.

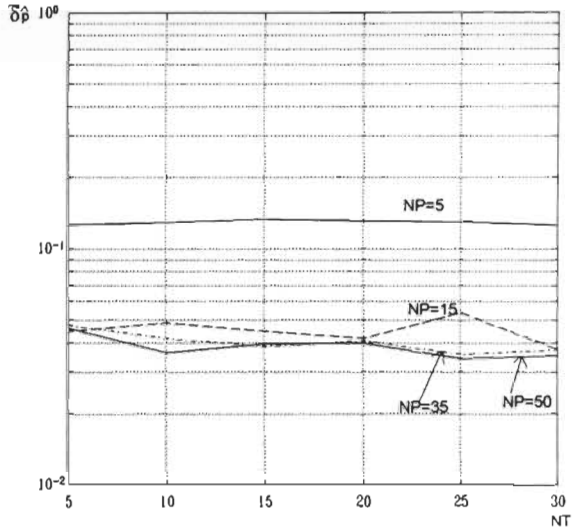

Figure 3.33 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR2: Δp=0.05 MPa ΔT=0.05 °C, Δv=0.002 V.

Après l'étude des procédures PCR1 et PCR2, nous remarquons que l'utilisation de splines cubiques donne de meilleurs résultats que l'utilisation de splines polynômiales. Par ailleurs les splines cubiques de lissage semblent peu performantes. Ceci est dû à un surlissage des données. Donc pour les simulations de PCR3 les splines cubiques de lissage sont utilisées pour l'étalonnage et les splines cubiques d'interpolation sont considérées pour la reconstruction. Pour PCR3 l'étape étalonnage correspond aux points 1 et 2 de PCR3 et l'étape reconstruction correspond aux points 3, 4, 5, 6 de PCR3.

Les figures 3.34 à 3.37 montrent les résultats de simulation obtenu à partir de PCR3 et de deux ensembles d'erreurs différents.

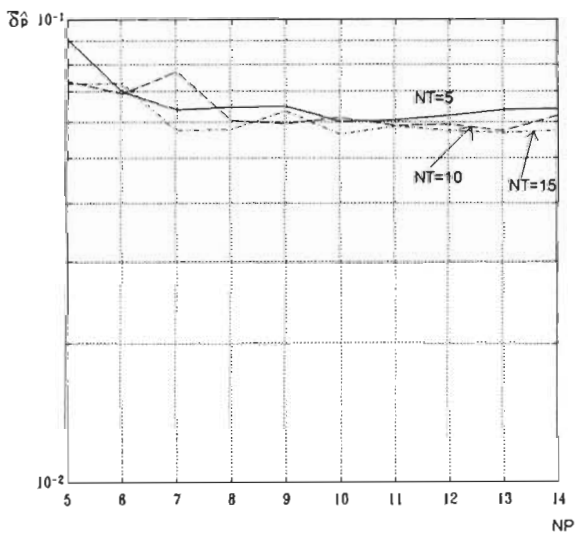

Figure 3.34 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR3: Δp=0.1 MPa ΔT=0.05 °C, Δv=0.002 V.

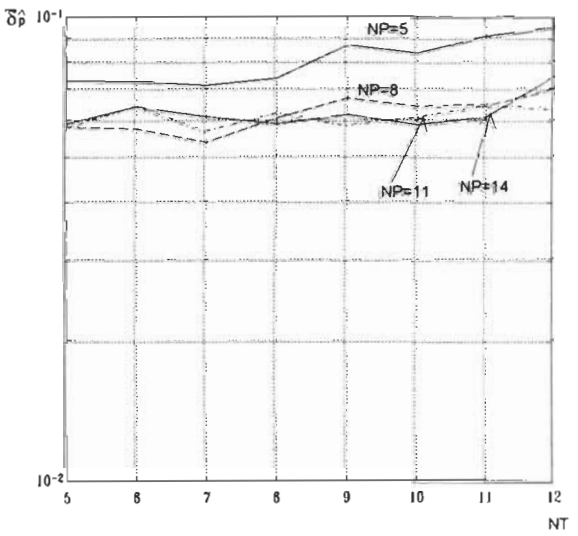

Figure 3.35 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR3: Δp=0.1 MPa ΔT=0.05 °C, Δv=0.002 V.

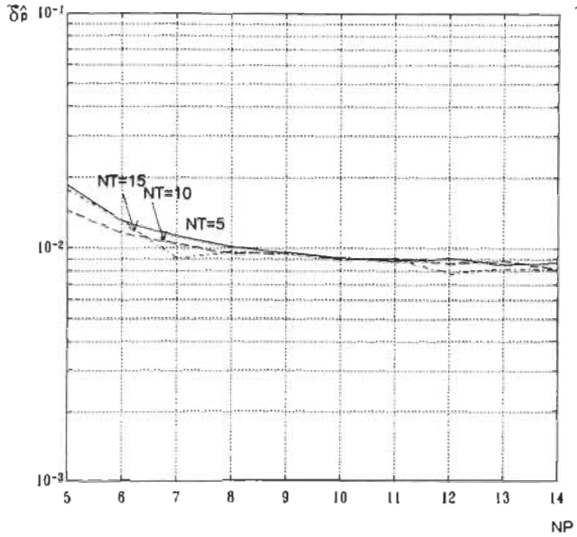

Figure 3.36 Dépendance de l'erreur relative limite en fonction du nombre de de pressions (NP) utilisées pour l'étalonnage par PCR3: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

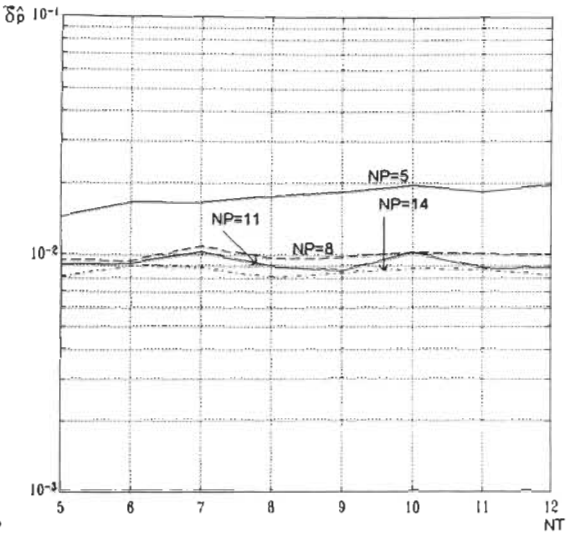

Figure 3.37 Dépendance de l'erreur relative limite en fonction du nombre de températures (NT) utilisées pour l'étalonnage par PCR3: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

Cette procédure semble être celle qui est le moins sensible aux erreurs de données de références utilisées pour l'étalonnage. De plus elle est souple ce qui lui permet de bien s'adapter aux variations des données d'étalonnage. Ici la souplesse de la procédure est caractérisée par le fait qu'elle utilise des fragments de la caractéristique qui sont quasi-linéaires et que les splines sont construites à partir de ces fragments de caractéristiques. Ceci fait que la procédure fonctionne bien même lorsque les données sont bruitées. Donc nous disons que cette procédure est souple.

3.4) Choix d'un algorithme

Parmi les critères possibles qui déterminent le choix d'une procédure, les deux critères qui semblent les plus importants pour le système de mesure traité sont :

- exactitude de mesure
- souplesse de l'algorithme

Si nous comparons les résultats obtenues pour chaque algorithme, nous remarquons que la procédure la moins sensible aux erreurs de données utilisées par l'étalonnage est PCR3. Vient par la suite PCR1 et PCR2.

Au niveau souplesse de la procédure, PCR1 semble être la plus souple car aucun prétraitement des données d'étalonnage est requis pour la faire converger à des valeurs de pressions adéquates. PCR2 semble être l'algorithme le moins souple dû à la construction de la caractéristique des capteurs à partir de caractéristiques auxiliaires.

Si nous regardons pour un algorithme en particulier répondant à une bonne exactitude et une bonne souplesse, PCR3 semble être le meilleur choix. D'où la vérification avec les données réelles se fera pour PCR3 seulement.

CHAPITRE 4

RESULTATS PRATIQUES

Ce chapitre a pour but de montrer l'exactitude d'une procédure choisie avec des données réelles. Dans ce chapitre nous commençons par la présentation des données réelles. À la section 4.2 nous présentons comment utiliser les données pour bien tester la procédure. À la section 4.3 les résultats obtenus à partir de la procédure sont présentés. Finalement une discussion sur les résultats obtenus est présenté à la section 4.4.

4.1) Présentation des données réelles

Les valeurs numériques des données réelles sont fournies à l'annexe B. Comme le montre les figures 4.1 et 4.2, les données réelles sont similaires aux données synthétiques. Pour les données synthétiques nous avons considérées une faible sensibilité du capteur aux variations de la température et des extrémums constants. Les données réelles montrent une forte sensibilité aux variations de la température et une variation des extrémums non négligeable.

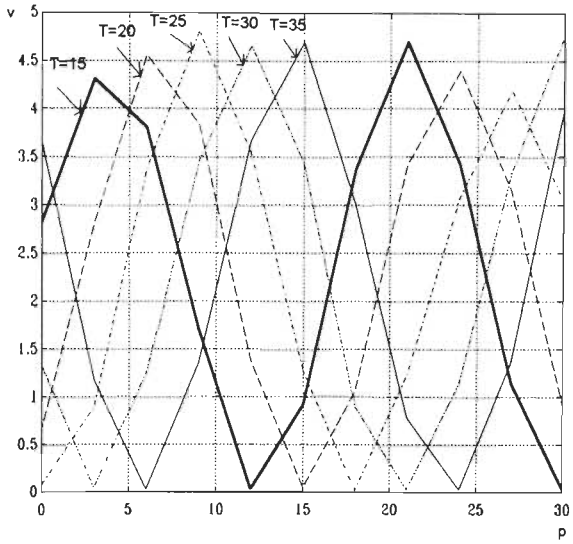

Figure 4.1 Données réelles pour un capteur de référence: tension de sortie en fonction de la pression pour différentes températures

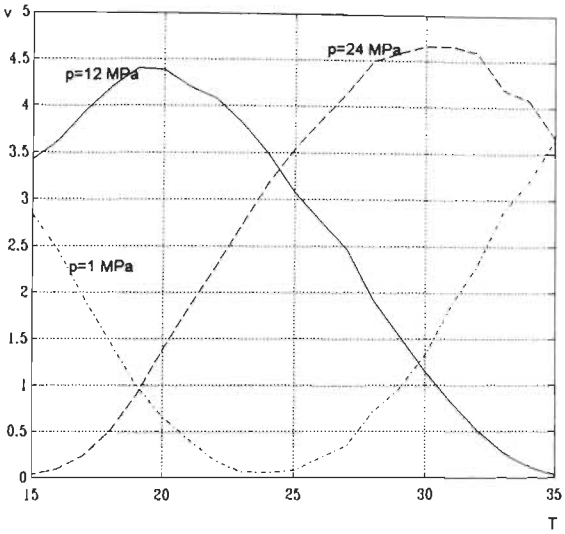

Figure 4.2 Données réelles pour un capteur de référence: tension de sortie en fonction de la température pour différentes pressions

Nous allons utiliser la procédure PCR3 avec des splines cubiques d'interpolation. Comme nous ne connaissons pas exactement les erreurs de mesure sur les données réelles, nous considérons que ces erreurs sont inférieure aux erreurs de mesure des données synthétiques. La prochaine section montre comment utiliser les données réelles pour les simulations.

4.2) Utilisation des données réelles

Les données réelles considérées sont pour un capteur de référence. Comme la seule différence entre les deux capteurs de référence est un déphasage de leurs caractéristiques statiques, alors les données réelles obtenues sont valides pour les deux capteurs de référence. Pour le guide nous allons considérer les données synthétiques parce qu'il n'influence pas l'exactitude de la mesure.

Pour fin de validation, les données réelles sont utilisées pour construire des splines d'interpolation. Ces splines servent de références. D'où pour NP valeurs de pressions et NT valeurs de températures on peut, à l'aide des courbes de références, déterminer [NPxNT] valeurs de tensions. Cet ensemble de pressions, températures et tensions sert à valider PCR3. Les figures 4.3 et 4.4 montrent les courbes de référence obtenues à partir des splines d'interpolation

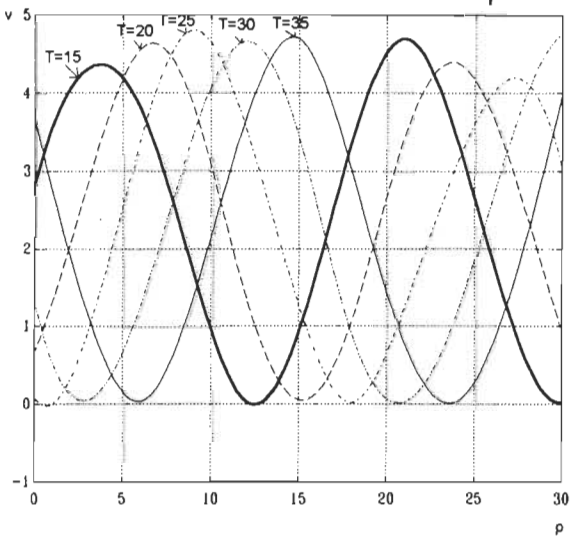

Figure 4.3 Courbes obtenues à partir de splines créées à partir de données réelles. Ces courbes représentent la tension en fonction de la pression pour différentes températures

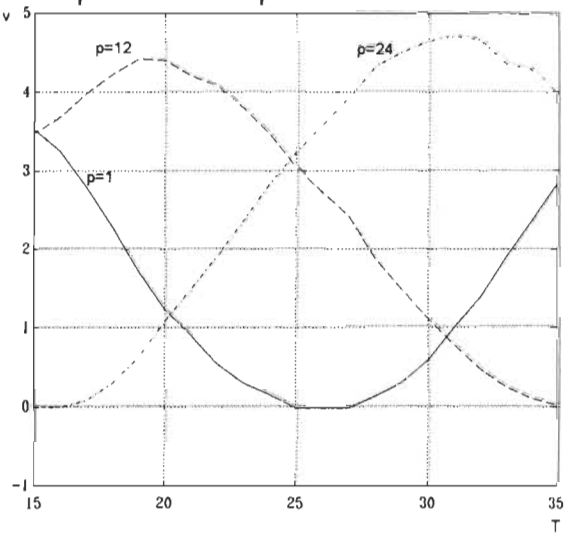

Figure 4.4 Courbes obtenues à partir de splines créées à partir de données réelles Ces courbes représentent le tension en fonction de la température pour différentes pressions

L'évaluation de la procédure se fait comme décrit au chapitre 3 à l'exception que les erreurs sont évaluées à l'aide des pressions, températures et tensions réelles.

Donc il nous est possible par la méthode décrite précédemment de vérifier PCR3 pour différents nombres de pressions et de températures.

Comme les données réelles sont considérées sans bruit, nous les affectons de différentes valeurs d'erreurs pour voir comment l'algorithme se comporte. La prochaine section montre les résultats obtenus à l'aide de PCR3.

4.3) Résultats pratiques obtenus pour la procédure PCR3

Les résultats obtenus pour PCR3 sont pour les gammes de pressions et de températures présentées au chapitre 3, soit:

p_{min}= 1 MPa p_{max}=26 MPa

T_{min}= 0 °C T_{max}=30 °C

Les résultats obtenus pour PCR3 avec les données réelles sont montrés aux figures 4.5 à 4.10.

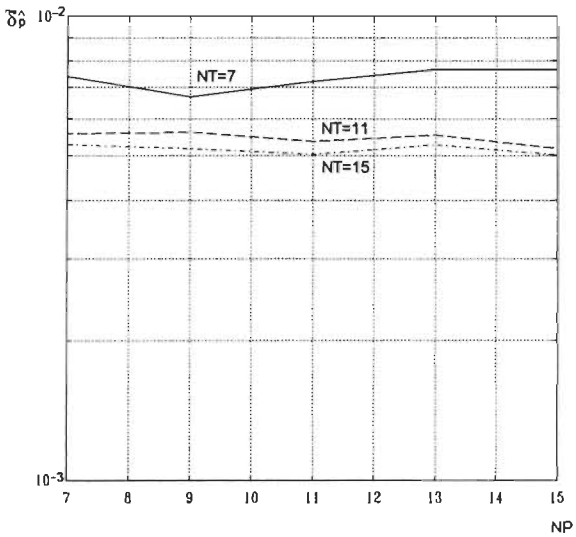

Figure 4.5 Dépendance de l'erreur relative limite en fonction du nombre de pressions réelles (NP) utilisées pour l'étalonnage: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

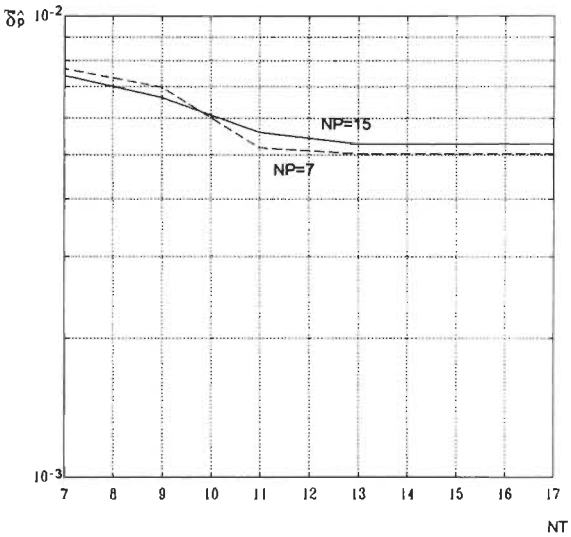

Figure 4.6 Dépendance de l'erreur relative limite en fonction du nombre de températures réelles (NT) utilisées pour l'étalonnage: Δp=0.01 MPa ΔT=0.01 °C, Δv=0.002 V.

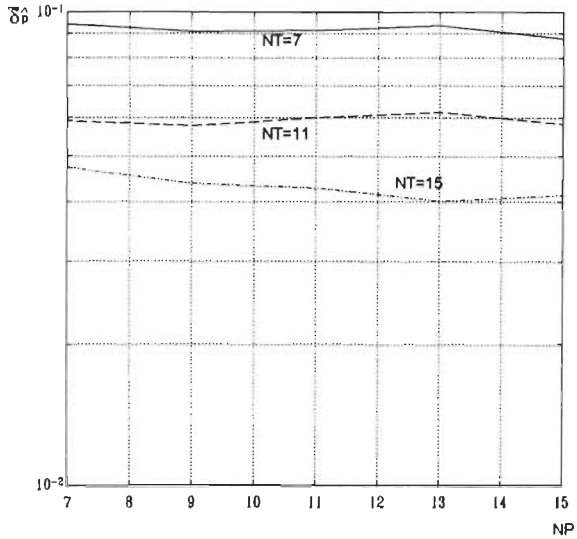

Figure 4.7 Dépendance de l'erreur relative limite en fonction du nombre de pressions réelles (NP) utilisées pour l'étalonnage: Δp=0.05 MPa ΔT=0.01 °C, Δv=0.002 V.

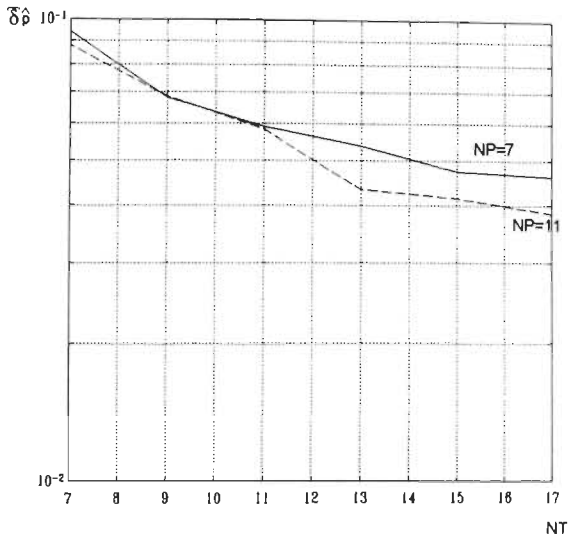

Figure 4.8 Dépendance de l'erreur relative limite en fonction du nombre de températures réelles (NT) utilisées pour l'étalonnage: Δp=0.05 MPa ΔT=0.01 °C, Δv=0.002 V.

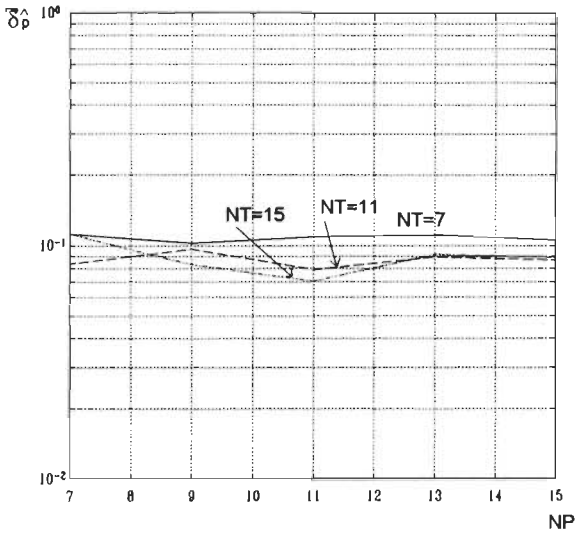

Figure 4.9 Dépendance de l'erreur relative limite en fonction du nombre de pressions réelles (NP) utilisées pour l'étalonnage: Δp=0.1 MPa ΔT=0.01 °C, Δv=0.002 V.

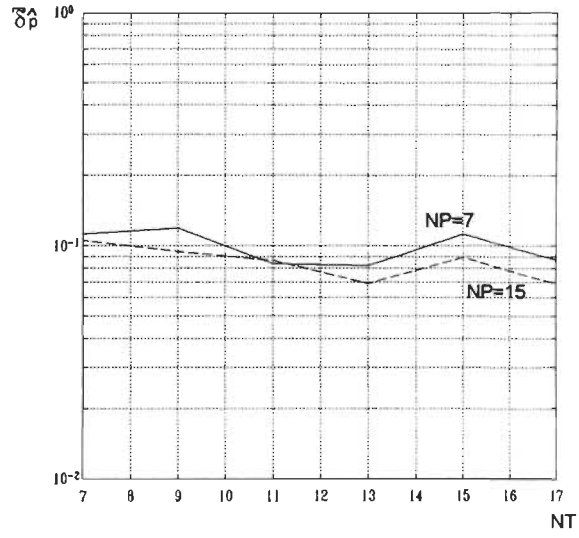

Figure 4.10 Dépendance de l'erreur relative limite en fonction du nombre de températures réelles (NT) utilisées pour l'étalonnage: Δp=0.1 MPa ΔT=0.01 °C, Δv=0.002 V.

4.4) Discussion

Les résultats nous montrent que la procédure fonctionne pour différentes erreurs de pression. Nous remarquons que la procédure 3 est plus sensible à l'augmentation du nombre de températures que du nombre de pressions. Ceci est probablement dû au fait que les données réelles sont entachées d'erreurs provenant de la mesure sur la température. Comme le niveau de cette erreur est inconnu, il nous est impossible d'incorporer cette information dans la procédure d'étalonnage. Donc pour pouvoir faire un meilleur étalonnage avec les données réelles, une estimation des variances des erreurs de mesure de la pression et de la température serait souhaitée.

Un phénomène montrant l'efficacité de la procédure PCR3 consiste à voir que l'erreur relative limite minimale est inférieure à l'erreur relative de la pression. Donc la procédure réussit à diminuer les erreurs de mesures. De plus nous remarquons que l'efficacité de la procédure augmente avec l'augmentation des erreurs. Ceci est dû à l'utilisation de splines de lissage dans la construction du modèle.

Si nous désirons implanter un modèle dans la mémoire d'un processeur et si les données réelles caractérisent adéquatement le capteur, un étalonnage avec un nombre de pressions (NP) de 11 et un nombre de températures égal à 11 donnerait de très bons résultats. Le choix du processeur dépendrait de la vitesse de reconstruction désirée. Comme pour le cas traité le temps de calcul n'est pas considéré, alors un processeur lent tel que le 68000 où un microcontrôleur tel que

le 68HC11 pourrait être employé. Pour obtenir une précision adéquate pour la reconstruction du mesurande, un processeur pouvant utiliser un point flottant avec une mantisse de 32 chiffres significatifs serait souhaité. Un processeur tel que le 68000 peut facilement utiliser un tel point flottant.

CONCLUSION

Le problème consistait à trouver une procédure d'étalonnage efficace pour un système de mesure de hautes pressions avec trois capteurs à fibre optique dont les caractéristiques statiques sont hautement non-linéaires. Ce travail a présenté trois procédures pouvant résoudre ce problème. Chacune de ces procédures peut être employées avec des fonctions splines d'interpolation ou avec des fonctions spline de lissage. Ces fonctions spline à une variable fûrent utilisées pour approximer des fonctions à deux variables.

L'étude des procédures développées (PCR1, PCR2 et PCR3) pour l'étalonnage du système de mesure de hautes pressions à fibre optique a été faite pour des données synthétiques et réelles où une acquisition équidistante de la pression et de la température a été réalisée. Ces algorithmes exigent un temps de calcul élevé pour l'étalonnage statique dû à sa complexité comparativement au temps de calcul pour la reconstitution de la pression.

Les résultats de calculs avec données synthétiques, présentés au chapitre 3, montrent la grande flexibilité des fonctions splines lorsque utilisées comme instrument de l'étalonnage d'un système de mesure. La flexibilité des fonctions spline est d'adapter ses paramètres aux données de références lors de l'étalonnage contrairement aux méthodes d'étalonnage dites paramétrées. Parmi les trois procédures, la procédure PCR3 qui consiste à appliquer les fonctions

spline sur des segments de la mesurande, semble être la plus performante. Ceci explique le choix de cette procédure pour une évaluation à l'aide de données réelles. Si les données sont affectées d'erreurs, nous remarquons que les fonctions spline de lissage s'avèrent plus efficaces que les fonctions spline d'interpolation. Ceci s'est vérifié surtout pour la procédure PCR3.

Lors de l'évaluation de la procédure PCR3 avec des données réelles, nous avons remarqué qu'une utilisation de 11 pressions équidistantes par segments monotone de la caractéristique statique et de 11 températures équidistantes pour une plage de pression de 1 à 100 MPa et de température de 0 à 30 °C donne de faibles erreurs relatives ($\overline{\delta \hat{p}}$=6*10^{-3}) comparitivement aux erreurs limites absolues de mesures ($\overline{\Delta}p = 0.01$ MPa, $\overline{\Delta}T = 0.01$ °C et $\overline{\Delta}v = 0.002$ V). La qualité des données utilisées pour l'estimation des paramètres de fonctions spline est essentielle pour la qualité de l'étalonnage.

Nous pouvons dire que l'objectif principal de ce travail qui consistait à trouver une procédure d'étalonnage adéquate pour le système de mesure de hautes pressions à fibre optique est atteint. Les procédures développées peuvent être également appliquée à d'autres cas où les caractéristiques statiques des capteurs sont fortement non-linéaire. Pour des développements futurs, une étude de la procédure avec d'autres types de fonctions spline (B-spline, β-spline) serait intéressante ainsi que le développement de nouvelles procédures utilisant des fonctions spline à deux variables.

BIBLIOGRAPHIE

[ASC87] G. Asch, Les capteurs en instrumentation industrielle, Dunod, Paris, 1987.

[BAR90a] A. Barwicz, W. J. Bock, "An Electronic High-Pressure Measuring System Using a Polarimetric Fiber-Optic Sensor," IEEE Trans. Instrum. Meas., vol 39, No 6, Dec 1990, pp 976-981.

[BAR90b] A. Barwicz, J. L. Dion, R. Z. Morawski, "Calibration of an Electronic Measuring System for Ultrasonic Analysis of Solutions," IEEE Trans. Instrum. and Meas., vol 39, Dec 1990, pp 1030-1033.

[BAR92] A. Barwicz, R. Z. Morawski, L. Lemire, J. Bock, "Calibration of an Electronic Multisensor System for Measuring High Pressures", IEEE Trans. on Instrum. & Measu., vol 41, No2, avril 1992, pp 269-273.

[BJÖ90] A Björck, Handbook of Numerical Analysis vol. 1, Finite difference methode (Part 1) - Solution of Equation in R^n (part 1), Elsevier Science Publishers B.V. , North-Holland, 1990.

[BOC89] W. J. Bock, A. Barwicz, "Electronic Measurement of High Pressure Using Birefringent Optical Fibers", Proc. Canadian Conf. Electrical and Computer Engineering, Sep 1989, pp 997-1000.

[BOC90] W. J. Bock, T. R. Wolinsky, A. Barwicz,"Developpement of a Polarimetric Optical Fiber Sensor for Electronic Measurement of High Pressure," IEEE Trans. Instrum.Meas., vol 39, Oct 1990, pp 715-721.

[DEB78] C. De Boor, A Practical Guide to Splines. New-York: Springer, 1978.

[EUB88] R. L. Eubank, Spline Smooting and Nonparametric Regression, New-York: Marcel Dekker, 1988.

[FES91] J. A. Fessler, "Nonparametric Fixed-Interval Smoothing with Vector Splines", IEEE Trans. Sign. Proces., vol 39, no 4, 1991.

[GIL78] P.E. Gill, W. Murray , "Algorithms for the Solution of the Nonlinear Least Squares Problem." SIAM J. Numer. Anal., vol 15, 1978, pp 977-992.

[HUT85] M. Hutchinson, F de Hoog, "Smoothing Noisy Data With Spline Functions", Numer. Math., vol.47,1985, pp 99-106.

[JAM85] M. L. James, G. M. Smith, J. C. Wolford, Applied Numerical Methods for Digital Computation (Third Edition), Harper & Row, publishers, New-York 1985.

[MEL82] J. M. Melvin, Numerical Analysis: A Practical Approch, Macmillan Publishing Co., Inc., New-York 1982.

[PRE86] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, Numerical Recipes: The art of Scientific Computing., Cambridge: Cambridge Univ. Press, 1986.

[REI67] C. H. Reinsch, "Smooting by Spline Functions", Numer. Math., vol. 10, 1967, pp177-183.

[REI71] C. H. Reinsch, "Smooting by Spline Functions II", Numer. Math., vol. 16, 1971, pp 451-454.

[ROB89] T. Robinson, R. Moyeed, "Making Robust the Cross-Validation Choice of Smoothing Parameter in Spline Smoothing Regression.", Commun. Statist. - Theory Math., vol. 18, no 2, 1989, pp 523-539.

[SCH81] L. L. Schumaker, Spline Functions: Basic Theory, New York: Wiley, 1981.

[WAH75] G. Wahba,"Smoothing Noisy Data with Spline Functions", Numer. Math., vol 24, 1975, pp 383-393.

[WAH78] P. Craven, G. Wahba, "Smoothing Noisy Data with Spline Functions", Numer. Math., vol. 31, 1979, pp 377-403.

[WAH80] G. Wahba, J. Wendelberger, "Some New Mathematical Methods for Variational Objective Analysis Using Splines and Cross Validation", Monthly Weather Rev., vol. 108, 1980, pp 1122-1143.

ANNEXE A

LISTE DES PROGRAMMES

Les procédures PCR1 et PCR2 sont écrit en HPBASIC et la procédure PCR3 est écrit pour le logiciel MATLAB.

Le programme principale pour la procédure PCR1 en HPBASIC:

```
10          CONTROL 32,2;1
20          STATUS 32,2;Status
30          PRINT "STATUS= ";Status
40          ! RTPRIO=0
50          ! ***************************************************************
60          !       PROGRAMME D'ETALLONNAGE
61          !
62          !       PROG1
70          !***************************************************************
71          !
80          OPTION BASE 1
90          DIM Donnee$[30],Donnee_sufx$[30],Er(5),Resul$[30]
100         INPUT "VOULEZ-VOUS RECUPERER UN FICHIER DE DONNES (O/N)",Rep$
110         !
120         !***************************************************************
130         !
140         !  LES FICHIERS DE LECTURE OU D'ECRITURE SONT DE TYPE ASCII
150         !
160         !***************************************************************
170         !
180         IF Rep$="O" THEN
190             LOOP
200                 Flag=0
210                 INPUT "ENTRER LE NOM DU FICHIER DE DONNEES A RECUPERER",Donnee$
220                 Donnee$="DONNEE.BAS/"&Donnee$
230                 ON ERROR GOSUB Esr2
240                 ASSIGN @F_1 TO Donnee$
250                 OFF ERROR
260             EXIT IF Flag=0
270             END LOOP
280             !
290             !***************************************************************
300             !
310             ! LE FICHIER DOIT CONTENIR DANS L'ORDRE SUIVANT 16 VALEURS REELS
320             ! DECRIVANT LE NOMBRE
330             ! D'ELEMENTS DES VECTEURS ET DES MATRICES, 8 VECTEURS, 10 MATRICES
340             ! ET 6 VALEURS REELS.
350             !
360             !***************************************************************
370             !
380             ENTER @F_1;K1,K2,K3,K4,V5,V6,V7,V8,M11,M12,M21,M22,M31,M32,M41,M42
390             ALLOCATE Vec_p1prim(V5),Vec_p2prim(V6)
400             ALLOCATE Vec_p1per(K1),Vec_p2per(K2),Vec_t1per(K3),Vec_t2per(K4)
410             ALLOCATE Vec_p1(V5),Vec_p2(V6),Vec_t1(V7),Vec_t2(V8)
420             ALLOCATE Mat_at1(M11,M12),Mat_bt1(M11,M12),Mat_ct1(M21,M22),Mat_dt1(M21,M22)
430             ALLOCATE Mat_at2(M21,M22),Mat_bt2(M21,M22),Mat_ct2(M21,M22),Mat_dt2(M21,M22)
440             ALLOCATE Mat_v2per(M41,M42),Mat_v1per(M31,M32),Mat_v1(2,2),Mat_v2(2,2)
450             ALLOCATE V1(K1),V2(K2),V3(K3),V4(K4),Ma1(M11,M12),Mb1(M11,M12)
```

```
460             ALLOCATE Mc1(M11,M12),Md1(M11,M12),Ma2(M21,M22),Mb2(M21,M22)
470             ALLOCATE Mc2(M21,M22),Md2(M21,M22),Mv1(M31,M32),Mv2(M41,M42)
480             ALLOCATE Vp1(V5),Vp2(V6),Vt1(V7),Vt2(V8)
490             !
500             !********************* ALLOCATION POUR DONNEE PRIMAIRE  ****************
510             !
520             ENTER @F_1;Vec_p1per(*),Vec_p2per(*),Vec_t1per(*),Vec_t2per(*)
530             ENTER @F_1;Vec_p1(*),Vec_p2(*),Vec_t1(*),Vec_t2(*)
540             ENTER @F_1;Mat_at1(*),Mat_bt1(*),Mat_ct1(*),Mat_dt1(*)
550             ENTER @F_1;Mat_at2(*),Mat_bt2(*),Mat_ct2(*),Mat_dt2(*)
560             ENTER @F_1;Mat_v1per(*),Mat_v2per(*)
570             ENTER @F_1;Chx_err,Erreur_v1,Erreur_v2,Erreur_t,Erreur_p,Chx_int
580             ASSIGN @F_1 TO *
590             Saute=1
600             Exis_fich=1
610         ELSE
620             INPUT "ENTRER Tmin",Tmin
630             INPUT "ENTER Tmax",Tmax
640             INPUT "ENTRER Pmin",Pmin
650             INPUT "ENTRER Pmax",Pmax
660             LOOP
670              INPUT "ERREUR POUR ETAL DONNEE PAR LOI NORMALE OU UNIFIRME (1/0)",Chx_err
680              IF Chx_err=1 THEN
690                     INPUT "ENTRER VARIANCE DE P",Var_p
700                     Erreur_p=Var_p
710                     INPUT "ENTRER VARIANCE DE T",Var_t
720                     Erreur_t=Var_t
730                     INPUT "ENTRER VARIANCE DE V1",Var_v1
740                     Erreur_v1=Var_v1
750                     INPUT "ENTRER VARIANCE DE V2",Var_v2
760                     Erreur_v2=Var_v2
770                     Sortir=1
780              ELSE
790                     IF Chx_err=0 THEN
800                         INPUT "ENTRER DELTA P",Delta_p
810                         Erreur_p=Delta_p
820                         INPUT "ENTRER DELTA T",Delta_t
830                         Erreur_t=Delta_t
840                         INPUT "ENTRER DELTA V1",Delta_v1
850                         Erreur_v1=Delta_v1
860                         INPUT "ENTRER DELTA V2",Delta_v2
870                         Erreur_v2=Delta_v2
880                         Sortir=1
890                     ELSE
900                         Sortir=0
910                     END IF
920              END IF
930             EXIT IF Sortir=1
940             END LOOP
950             Saute=0
960             Exis_fich=0
970         END IF
980         INPUT "ERREUR POUR EVAL DONNEE PAR LOI NORMALE OU UNIFIRME (1/0)",Cerr
990         INPUT "ENTRER LA VALEUR DE LA VARIANCE OU ERREUR UNIF POUR T",Err_t
1000 INPUT "ENTRER LA VALEUR DE LA VARIANCE OU ERREUR UNIF POUR V1",Err_v1
1010 INPUT "ENTRER LA VALEUR DE LA VARIANCE OU ERREUR UNIF POUR V2",Err_v2 1020  Er(1)=Cerr
1030 Er(2)=Err_t
1040 Er(3)=Err_v1
1050 Er(4)=Err_v2
1060 Er(5)=Erreur_p
1070 Sortir=0
1080 CLEAR SCREEN
1090 PRINT " LES INTERPOLATIONS POSSIBLES QUE PERMET CE PROGRAMME SONT:"
1100 PRINT
1110 PRINT "            1) INTSPL3"
1120 PRINT "            2) OPTSPL3"
```

```
1130 PRINT "            3) OPTSPL3_B"
1140 PRINT "            4) INTSPL2"
1150 PRINT "            5) OPTSPL2"
1160 PRINT "            6) SMOSPL3"
1170 PRINT "            7) SMOSPL2"
1180 LOOP
1190            INPUT " ENTRER VOTRE CHOIX",Chx_int
1200            IF Chx_int<=7 THEN
1210             IF Chx_int>=1 THEN
1220                    Sortir=1
1230             END IF
1240            END IF
1250 EXIT IF Sortir=1
1260 END LOOP
1270 CLEAR SCREEN
1280 INPUT "ENTRER LE NOMBRE D'ITERATION DE TEMPERATURE DESIREE",Nbr_temp
1290 INPUT "ENTRER LE NOMBRE D'ITERATION DE PRESSION DESIREE",Nbr_pres
1300 INPUT "ENTRER LE NOM DU FICHIER DE SAUVEGARDE DES DONNEES",Donnee$
1310 Donnee$="DONNEE.BAS/"&Donnee$
1320 INPUT "ENTRER LE NOM DU FICHIER POUR SAUVEGARDE DES RESULTATS",Resul$
1330 Resul$="RESULTAT.BAS/"&Resul$
1340 ON ERROR GOSUB Esr1
1350 CREATE ASCII Resul$,10
1360 OFF ERROR
1370 ASSIGN @F_2 TO Resul$
1380 Total_iter=Nbr_temp*Nbr_pres
1390 OUTPUT @F_2;Chx_int,Chx_err,Erreur_p,Erreur_t,Erreur_v1,Erreur_v2
1400 !
1410 !*****************************************************************
1420 !
1430 !    SI FICHIER DE DONNEE N'EXISTE PAS ALORS CALCUL DE
1440 !    VEC_P1, VEC_P2, MAT_V1, MAT_V2, MAT_V1PER
1450 !    MAT_V2PER, MAT_P1PER, MAT_P2PER, VEC_T1, VEC_T1PER, VEC_T2
1460 !    VEC_T2PER, MAT_AT1, MAT_BT1, MAT_CT1, MAT_DT1, MAT_AT2, MAT_BT2
1470 !    MAT_CT2, MAT_DT2
1480 !
1490 !*****************************************************************
1500 !
1510 IF Saute=0 THEN
1520            Dim_p=5
1530            Dim_t=5
1540            ALLOCATE Vec_p1prim(Dim_p),Vec_p2prim(Dim_p),Vec_p1(Dim_p),Vec_p2(Dim_p),Vec_p1per(Dim_p),Vec_p2per(Dim
1550            ALLOCATE Vec_t1(Dim_t),Vec_t2(Dim_t),Vec_t1per(Dim_t),Vec_t2per(Dim_t)
1560            ALLOCATE Mat_v1(Dim_p,Dim_t),Mat_v2(Dim_p,Dim_t),Mat_v1per(Dim_p,Dim_t),Mat_v2per(Dim_p,Dim_t)
1570            ALLOCATE Mat_at1(Dim_p,Dim_t),Mat_bt1(Dim_p,Dim_t),Mat_ct1(Dim_p,Dim_t),Mat_dt1(Dim_p,Dim_t)
1580            ALLOCATE Mat_at2(Dim_p,Dim_t),Mat_bt2(Dim_p,Dim_t),Mat_ct2(Dim_p,Dim_t),Mat_dt2(Dim_p,Dim_t)
1590            ALLOCATE V1(Dim_p),V2(Dim_p),V3(Dim_t),V4(Dim_t),Ma1(Dim_p,Dim_t),Mb1(Dim_p,Dim_t),Mc1(Dim_p,Dim_t)
1600            ALLOCATE Md1(Dim_p,Dim_t),Ma2(Dim_p,Dim_t),Mb2(Dim_p,Dim_t),Mc2(Dim_p,Dim_t),Md2(Dim_p,Dim_t),
Mv1(Dim_p,Dim_t)
1610            ALLOCATE Mv2(Dim_p,Dim_t),Vp1(Dim_p),Vp2(Dim_p),Vt1(Dim_t),Vt2(Dim_t)
1620 !
1630 !*****************************************************************
1640 !
1650 ! CALCUL DE V1, V2, T1 ET T2
1660 !
1670 !*****************************************************************
1680 !
1690            Pas_temp=(Tmax-Tmin)/(Dim_t-1)
1700            FOR I=1 TO Dim_t
1710             Vec_t1(I)=Tmin+(I-1)*Pas_temp
1720             Vec_t2(I)=Tmin+(I-1)*Pas_temp
1730            NEXT I
1740            Pas_pres=(Pmax-Pmin)/(Dim_p-1)
1750            FOR I=1 TO Dim_p
1760             Vec_p1(I)=Pmin+(I-1)*Pas_pres
1770             Vec_p2(I)=Pmin+(I-1)*Pas_pres
```

```
1780              NEXT I
1790              V5=Dim_p
1800              V7=Dim_t
1810  END IF
1820  DIM Rtr(8)
1830  OUTPUT @F_2;Err_t,Err_v1,Err_v2,Vec_p1(1),Vec_p1(V5),Vec_t1(1),Vec_t1(V7) 1840  OUTPUT @F_2;Total_iter
1850  Fin=0
1860  CLEAR SCREEN
1870  !
1880  !*************************************************************************
1890  !
1900  !     GARDE EN MEMOIRE DES DONNEES PRIMAIRE
1910  !
1920  !*************************************************************************
1930  !
1940  RANDOMIZE
1950  MAT Vec_p1prim= Vec_p1
1960  MAT Vec_p2prim= Vec_p2
1970  OUTPUT @F_2;Nbr_temp,Nbr_pres
1980  FOR I=1 TO Nbr_temp
1990   PRINT "NBR_TEMP=";I
2000   FOR J=1 TO Nbr_pres
2010    IF Saute=0 THEN
2020     CALL Det_v12(Vec_p1(*),Vec_t1(*),Vec_p2(*),Vec_t2(*),Mat_v1(*),Mat_v2(*))
2030     IF Chx_err=1 THEN
2040                          CALL Errnor(Vec_p1(*),Vec_p1per(*),Var_p)
2050                          Vec_p1per(1)=Vec_p1(1)-Var_p*RND/2
2060                          Transit=SIZE(Vec_p1per,1)
2070                          Vec_p1per(Transit)=Vec_p1(Transit)+Var_p*RND/2
2080                          CALL Errnor(Vec_p2(*),Vec_p2per(*),Var_p)
2090                          Vec_p2per(1)=Vec_p2(1)-Var_p*RND/2
2100                          Transit=SIZE(Vec_p2per,1)
2110                          Vec_p2per(Transit)=Vec_p2(Transit)+Var_p*RND/2
2120                          CALL Errnor(Vec_t1(*),Vec_t1per(*),Var_t)
2130                          Vec_t1per(1)=Vec_t1(1)-Var_t*RND/2
2140                          Transit=SIZE(Vec_t1per,1)
2150                          Vec_t1per(Transit)=Vec_t1(Transit)+Var_t*RND/2
2160                          CALL Errnor(Vec_t2(*),Vec_t2per(*),Var_t)
2170                          Vec_t2per(1)=Vec_t2(1)-Var_t*RND/2
2180                          Transit=SIZE(Vec_t2per,1)
2190                          Vec_t2per(Transit)=Vec_t2(Transit)+Var_t*RND/2
2200                          CALL Errnor(Mat_v1(*),Mat_v1per(*),Var_v1)
2210                          Transit1=SIZE(Mat_v1,1)
2220                          Transit2=SIZE(Mat_v1,2)
2230                          FOR Boucle=1 TO Transit2
2240                           Mat_v1per(1,Boucle)=Mat_v1(1,Boucle)-Var_v1*RND/2
2250                           Mat_v1per(Transit1,Boucle)=Mat_v1(Transit1,Boucle)+Var_v1*RND/2
2260                          NEXT Boucle
2270                          CALL Errnor(Mat_v2(*),Mat_v2per(*),Var_v2)
2280                          Transit1=SIZE(Mat_v2,1)
2290                          Transit2=SIZE(Mat_v2,2)
2300                          FOR Boucle=1 TO Transit2
2310                           Mat_v2per(1,Boucle)=Mat_v2(1,Boucle)-Var_v2*RND/2
2320                           Mat_v2per(Transit1,Boucle)=Mat_v2(Transit1,Boucle)+Var_v2*RND/2
2330                          NEXT Boucle
2340     ELSE
2350                          CALL Errunif(Vec_p1(*),Vec_p1per(*),Delta_p)
2360                          Vec_p1per(1)=Vec_p1(1)-Delta_p*RND/2
2370                          Transit=SIZE(Vec_p1per,1)
2380                          Vec_p1per(Transit)=Vec_p1(Transit)+Delta_p*RND/2
2390                          CALL Errunif(Vec_p2(*),Vec_p2per(*),Delta_p)
2400                          Vec_p2per(1)=Vec_p2(1)-Delta_p*RND/2
2410                          Transit=SIZE(Vec_p2per,1)
2420                          Vec_p2per(Transit)=Vec_p2(Transit)+Delta_p*RND/2
2430                          CALL Errunif(Vec_t1(*),Vec_t1per(*),Delta_t)
2440                          Vec_t1per(1)=Vec_t1(1)-Delta_t*RND/2
```

```
2450                Transit=SIZE(Vec_t1per,1)
2460                Vec_t1per(Transit)=Vec_t1(Transit)+Delta_t*RND/2
2470                CALL Errunif(Vec_t2(*),Vec_t2per(*),Delta_t)
2480                Vec_t2per(1)=Vec_t2(1)-Delta_t*RND/2
2490                Transit=SIZE(Vec_t2per,1)
2500                Vec_t2per(Transit)=Vec_t2(Transit)+Delta_t*RND/2
2510                CALL Errunif(Mat_v1(*),Mat_v1per(*),Delta_v1)
2520                Transit1=SIZE(Mat_v1,1)
2530                Transit2=SIZE(Mat_v1,2)
2540                FOR Boucle=1 TO Transit2
2550                  Mat_v1per(1,Boucle)=Mat_v1(1,Boucle)-Delta_v1*RND/2
2560                  Mat_v1per(Transit1,Boucle)=Mat_v1(Transit1,Boucle)+Delta_v1*RND/2
2570                NEXT Boucle
2580                CALL Errunif(Mat_v2(*),Mat_v2per(*),Delta_v2)
2590                Transit1=SIZE(Mat_v2,1)
2600                Transit2=SIZE(Mat_v2,2)
2610                FOR Boucle=1 TO Transit2
2620                    Mat_v2per(1,Boucle)=Mat_v2(1,Boucle)-Delta_v2*RND/2
2630                    Mat_v2per(Transit1,Boucle)=Mat_v2(Transit1,Boucle)+Delta_v2*RND/2
2640                NEXT Boucle
2650            END IF
2660            Longeur=SIZE(Vec_t1per,1)
2670            N=SIZE(Vec_p1per,1)
2680            ALLOCATE Var1(Longeur),Var2a(Longeur),Var2b(Longeur),Var2c(Longeur),Var2d(Longeur)
2690            FOR K=1 TO N
2700                MAT Var1= Mat_v1per(K,*)
2710                SELECT Chx_int
2720                CASE 1
2730                    CALL Intspl3_2(Vec_t1per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*))
2740                    MAT Mat_at1(K,*)= Var2a
2750                    MAT Mat_bt1(K,*)= Var2b
2760                    MAT Mat_ct1(K,*)= Var2c
2770                    MAT Mat_dt1(K,*)= Var2d
2780                CASE 2
2790                    CALL Optspl3_2(Vec_t1per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*))
2800                    MAT Mat_at1(K,*)= Var2a
2810                    MAT Mat_bt1(K,*)= Var2b
2820                    MAT Mat_ct1(K,*)= Var2c
2830                    MAT Mat_dt1(K,*)= Var2d
2840                CASE 3
2850                    CALL Optspl3b_2(Vec_t1per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*))
2860                    MAT Mat_at1(K,*)= Var2a
2870                    MAT Mat_bt1(K,*)= Var2b
2880                    MAT Mat_ct1(K,*)= Var2c
2890                    MAT Mat_dt1(K,*)= Var2d
2900                CASE 4
2910                    CALL Intspl2(Vec_t1per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*))
2920                    MAT Mat_at1(K,*)= Var2a
2930                    MAT Mat_bt1(K,*)= Var2b
2940                    MAT Mat_ct1(K,*)= Var2c
2950                CASE 5
2960                    CALL Optspl2(Vec_t1per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*))
2970                    MAT Mat_at1(K,*)= Var2a
2980                    MAT Mat_bt1(K,*)= Var2b
2990                    MAT Mat_ct1(K,*)= Var2c
3000                CASE 6
3010                    S=INT(Longeur+SQRT(2*Longeur))
3020                    Sigy=Erreur_v1/2
3030                    CALL Smospl3(Vec_t1per(*),Var1(*),S,Sigy,Var2a(*),Var2b(*),Var2c(*),Var2d(*))
3040                    MAT Mat_at1(K,*)= Var2a
3050                    MAT Mat_bt1(K,*)= Var2b
3060                    MAT Mat_ct1(K,*)= Var2c
3070                    MAT Mat_dt1(K,*)= Var2d
3080                CASE 7
3090                    S=INT(Longeur+SQRT(2*Longeur))
3100                    Sigy=Erreur_v1/2
```

```
3110                    IF Sigy=0 THEN
3120                      Sigy=.000001
3130                    END IF
3140                    CALL Smospl2a(Vec_t1per(*),Var1(*),Sigy,S,Var2a(*),Var2b(*),Var2c(*))
3150                    MAT Mat_at1(K,*)= Var2a
3160                    MAT Mat_bt1(K,*)= Var2b
3170                    MAT Mat_ct1(K,*)= Var2c
3180                END SELECT
3190            NEXT K
3200            DEALLOCATE Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*)
3210            Longeur=SIZE(Vec_t2per,1)
3220            N=SIZE(Vec_p2per,1)
3230            ALLOCATE Var1(Longeur),Var2a(Longeur),Var2b(Longeur),Var2c(Longeur),Var2d(Longeur)
3240            FOR K=1 TO N
3250                MAT Var1= Mat_v2per(K,*)
3260                SELECT Chx_int
3270                CASE 1
3280                    CALL Intspl3_2(Vec_t2per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*))
3290                    MAT Mat_at2(K,*)= Var2a
3300                    MAT Mat_bt2(K,*)= Var2b
3310                    MAT Mat_ct2(K,*)= Var2c
3320                    MAT Mat_dt2(K,*)= Var2d
3330                CASE 2
3340                    CALL Optspl3_2(Vec_t2per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*))
3350                    MAT Mat_at2(K,*)= Var2a
3360                    MAT Mat_bt2(K,*)= Var2b
3370                    MAT Mat_ct2(K,*)= Var2c
3380                    MAT Mat_dt2(K,*)= Var2d
3390                CASE 3
3400                    CALL Optspl3b_2(Vec_t2per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*))
3410                    MAT Mat_at2(K,*)= Var2a
3420                    MAT Mat_bt2(K,*)= Var2b
3430                    MAT Mat_ct2(K,*)= Var2c
3440                    MAT Mat_dt2(K,*)= Var2d
3450                CASE 4
3460                    CALL Intspl2(Vec_t2per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*))
3470                    MAT Mat_at2(K,*)= Var2a
3480                    MAT Mat_bt2(K,*)= Var2b
3490                    MAT Mat_ct2(K,*)= Var2c
3500                CASE 5
3510                    CALL Optspl2(Vec_t2per(*),Var1(*),Var2a(*),Var2b(*),Var2c(*))
3520                    MAT Mat_at2(K,*)= Var2a
3530                    MAT Mat_bt2(K,*)= Var2b
3540                    MAT Mat_ct2(K,*)= Var2c
3550                CASE 6
3560                    S=INT(Longeur+SQRT(2*Longeur))
3570                    Sigy=Erreur_v2/2
3580                    CALL Smospl3(Vec_t2per(*),Var1(*),S,Sigy,Var2a(*),Var2b(*),Var2c(*),Var2d(*))
3590                    MAT Mat_at2(K,*)= Var2a
3600                    MAT Mat_bt2(K,*)= Var2b
3610                    MAT Mat_ct2(K,*)= Var2c
3620                    MAT Mat_dt2(K,*)= Var2d
3630                CASE 7
3640                    S=INT(Longeur+SQRT(2*Longeur))
3650                    Sigy=Erreur_v2/2
3660                    IF Sigy=0 THEN
3670                      Sigy=.00001
3680                    END IF
3690                    CALL Smospl2a(Vec_t2per(*),Var1(*),Sigy,S,Var2a(*),Var2b(*),Var2c(*))
3700                    MAT Mat_at2(K,*)= Var2a
3710                    MAT Mat_bt2(K,*)= Var2b
3720                    MAT Mat_ct2(K,*)= Var2c
3730                END SELECT
3740            NEXT K
3750            DEALLOCATE Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*)
3760 !
```

```
3770 !
3780 !*****************************************************************
3790 !
3800 !                          SAUVEGARDE DES DONNEES
3810 !
3820 !*****************************************************************
3830 !
3840                       V5=SIZE(Vec_p1,1)
3850                       V6=SIZE(Vec_p2,1)
3860                       V7=SIZE(Vec_t1,1)
3870                       V8=SIZE(Vec_t2,1)
3880                       M11=SIZE(Mat_at1,1)
3890                       M12=SIZE(Mat_at1,2)
3900                       M21=SIZE(Mat_at2,1)
3910                       M22=SIZE(Mat_at2,2)
3920                       M31=SIZE(Mat_v1per,1)
3930                       M32=SIZE(Mat_v1per,2)
3940                     M41=SIZE(Mat_v2per,1)
3950                     M42=SIZE(Mat_v2per,2)
3960                   END IF
3970                   Saute=0
3980                   MAT V1= Vec_p1per
3990                   MAT V2= Vec_p2per
4000                   MAT V3= Vec_t1per
4010                   MAT V4= Vec_t2per
4020                   MAT Ma1= Mat_at1
4030                   MAT Mb1= Mat_bt1
4040                   MAT Mc1= Mat_ct1
4050                   MAT Md1= Mat_dt1
4060                   MAT Ma2= Mat_at2
4070                   MAT Mb2= Mat_bt2
4080                   MAT Mc2= Mat_ct2
4090                   MAT Md2= Mat_dt2
4100                   MAT Mv1= Mat_v1per
4110                   MAT Mv2= Mat_v2per
4120                   MAT Vp1= Vec_p1
4130                   MAT Vp2= Vec_p2
4140                   MAT Vt1= Vec_t1
4150                   MAT Vt2= Vec_t2
4160                   Cint=Chx_int
4170                   CALL Eval(V1(*),V2(*),V3(*),V4(*),Vp1(*),Vp2(*),Vt1(*),Vt2(*),Ma1(*),Mb1(*),Mc1(*),Md1(*),Ma2(*),Mb2(*),
Mc2(*),Md2(*),Mv1(*),Mv2(*),Cint,Er(*),Rtr(*))
4180                   P1n=Rtr(1)
4190                   T1n=Rtr(2)
4200                   P2n=Rtr(3)
4210                   T2n=Rtr(4)
4220                   Errmoy=Rtr(5)
4230                   Errmax=Rtr(6)
4240                   Ecart_reel=Rtr(7)
4250 !                 PRINT "ERRMAX= ";Errmax
4260 !                 PRINT "ERREUR RELAT. MAX.= ";Ecart_reel
4270 !                 PRINT "ERRMOY= ";Errmoy
4280 !                 PRINT "   "
4290 !                 PRINT "PRES NOV= ";P1n
4300 !                 PRINT "TEMP NOV= ";T1n
4310 !                 PRINT "   "
4320                   Fin=Rtr(8)
4330                   B=5
4340                   ALLOCATE Vec_p1n(V5+B),Vec_p2n(V6+B),Vec_t1n(V7),Vec_t2n(V8)
4350                   MAT Vec_t1n= Vec_t1
4360                   MAT Vec_t2n= Vec_t2
4370 !                 CALL Vec_nov(Vec_p1(*),Vec_p1n(*),P1n)
4380 !                 CALL Vec_nov(Vec_p2(*),Vec_p2n(*),P2n)
4390                   Pas_pres=(Pmax-Pmin)/(V5+B-1)
4400                   FOR Kk=1 TO V5+B
4410                     Vec_p1n(Kk)=Pmin+(Kk-1)*Pas_pres
```

```
4420                NEXT Kk
4430                Pas_pres=(Pmax-Pmin)/(V6+B-1)
4440                FOR Kk=1 TO V6+B
4450                  Vec_p2n(Kk)=Pmin+(Kk-1)*Pas_pres
4460                NEXT Kk
4470                ON ERROR GOSUB Esr3
4480                CREATE ASCII "TRANSIT",10
4490                OFF ERROR
4500                ASSIGN @F_3 TO "TRANSIT"
4510                OUTPUT @F_3;Vec_p1n(*),Vec_p2n(*),Vec_t1n(*),Vec_t2n(*)
4520                ASSIGN @F_3 TO *
4530                DEALLOCATE Vec_p1n(*),Vec_p2n(*),Vec_t1n(*),Vec_t2n(*)
4540                DEALLOCATE Vec_p1(*),Vec_p2(*),Vec_t1(*),Vec_t2(*),Vec_p1per(*)
4550                DEALLOCATE Vec_p2per(*),Vec_t1per(*),Vec_t2per(*),Mat_v1(*),Mat_v2(*)
4560                DEALLOCATE Mat_v1per(*),Mat_v2per(*),Mat_at1(*),Mat_bt1(*),Mat_ct1(*)
4570                DEALLOCATE Mat_dt1(*),Mat_at2(*),Mat_bt2(*),Mat_ct2(*),Mat_dt2(*)
4580                DEALLOCATE V1(*),V2(*),V3(*),V4(*),Ma1(*),Mb1(*),Mc1(*),Md1(*)
4590                DEALLOCATE Ma2(*),Mb2(*),Mc2(*),Md2(*),Mv1(*),Mv2(*),Vp1(*),Vp2(*),Vt1(*),Vt2(*)
4600                ALLOCATE Vec_p1(V5+B),Vec_p2(V6+B),Vec_p1per(V5+B),Vec_p2per(V6+B)
4610                ALLOCATE Vec_t1(V7),Vec_t2(V8),Vec_t1per(V7),Vec_t2per(V8)
4620                ALLOCATE Mat_v1(V5+B,V7),Mat_v2(V6+B,V8),Mat_v1per(V5+B,V7),Mat_v2per(V6+B,V8)
4630                ALLOCATE Mat_at1(V5+B,V7),Mat_bt1(V5+B,V7),Mat_ct1(V5+B,V7),Mat_dt1(V5+B,V7)
4640                ALLOCATE Mat_at2(V6+B,V8),Mat_bt2(V6+B,V8),Mat_ct2(V6+B,V8),Mat_dt2(V6+B,V8)
4650                ALLOCATE V1(V5+B),V2(V6+B),V3(V7),V4(V8),Ma1(V5+B,V7),Mb1(V5+B,V7),Mc1(V5+B,V7)
4660                ALLOCATE Md1(V5+B,V7),Ma2(V6+B,V8),Mb2(V6+B,V8),Mc2(V6+B,V8),Md2(V6+B,V8),Mv1(V5+B,V7)
4670                ALLOCATE Mv2(V6+B,V8),Vp1(V5+B),Vp2(V6+B),Vt1(V7),Vt2(V8)
4680                ASSIGN @F_3 TO "TRANSIT"
4690                ENTER @F_3;Vec_p1(*),Vec_p2(*),Vec_t1(*),Vec_t2(*)
4700                ASSIGN @F_3 TO *
4710              !
4720              !
4730              !*****************************************************************
4740              !
4750              !   ECRITURE DES RESULTATS DANS LE FICHIER ADEQUAT
4760              !
4770              !
4780              !*****************************************************************
4790              !
4800              !
4810                  OUTPUT @F_2;V5,V7,V6,V8,Errmax,Errmoy,Ecart_reel
4820            NEXT J
4830            V5=SIZE(Vec_p1prim,1)
4840            V6=SIZE(Vec_p2prim,1)
4850            ALLOCATE Vec_t1n(V7+B),Vec_t2n(V8+B)
4860 !  CALL Vec_nov(Vec_t1(*),Vec_t1n(*),T1n)
4870 !  CALL Vec_nov(Vec_t2(*),Vec_t2n(*),T2n)
4880            Pas_temp=(Tmax-Tmin)/(V7+B-1)
4890            FOR Kk=1 TO V7+B
4900                Vec_t1n(Kk)=Tmin+(Kk-1)*Pas_temp
4910            NEXT Kk
4920            Pas_temp=(Tmax-Tmin)/(V8+B-1)
4930            FOR Kk=1 TO V8+B
4940                Vec_t2n(Kk)=Tmin+(Kk-1)*Pas_temp
4950            NEXT Kk
4960            ASSIGN @F_3 TO "TRANSIT"
4970            OUTPUT @F_3;Vec_t1n(*),Vec_t2n(*)
4980            ASSIGN @F_3 TO *
4990            DEALLOCATE Vec_t1n(*),Vec_t2n(*)
5000            DEALLOCATE Vec_p1(*),Vec_p2(*),Vec_t1(*),Vec_t2(*),Vec_p1per(*)
5010            DEALLOCATE Vec_p2per(*),Vec_t1per(*),Vec_t2per(*),Mat_v1(*),Mat_v2(*)
5020            DEALLOCATE Mat_v1per(*),Mat_v2per(*),Mat_at1(*),Mat_bt1(*),Mat_ct1(*)
5030            DEALLOCATE Mat_dt1(*),Mat_at2(*),Mat_bt2(*),Mat_ct2(*),Mat_dt2(*)
5040            DEALLOCATE V1(*),V2(*),V3(*),V4(*),Ma1(*),Mb1(*),Mc1(*),Md1(*)
5050            DEALLOCATE Ma2(*),Mb2(*),Mc2(*),Md2(*),Mv1(*),Mv2(*)
5060            DEALLOCATE Vp1(*),Vp2(*),Vt1(*),Vt2(*)
5070            ALLOCATE Vec_p1(V5),Vec_p2(V6),Vec_p1per(V5),Vec_p2per(V6)
```

```
5080              ALLOCATE Vec_t1(V7+B),Vec_t2(V8+B),Vec_t1per(V7+B),Vec_t2per(V8+B)
5090              ALLOCATE Mat_v1(V5,V7+B),Mat_v2(V6,V8+B),Mat_v1per(V5,V7+B),Mat_v2per(V6,V8+B)
5100              ALLOCATE Mat_at1(V5,V7+B),Mat_bt1(V5,V7+B),Mat_ct1(V5,V7+B),Mat_dt1(V5,V7+B)
5110              ALLOCATE Mat_at2(V6,V8+B),Mat_bt2(V6,V8+B),Mat_ct2(V6,V8+B),Mat_dt2(V6,V8+B)
5120              ALLOCATE V1(V5),V2(V6),V3(V7+B),V4(V8+B),Ma1(V5,V7+B),Mb1(V5,V7+B),Mc1(V5,V7+B)
5130              ALLOCATE Md1(V5,V7+B),Ma2(V6,V8+B),Mb2(V6,V8+B),Mc2(V6,V8+B),Md2(V6,V8+B),Mv1(V5,V7+B)
5140              ALLOCATE Mv2(V6,V8+B),Vp1(V5),Vp2(V6),Vt1(V7+B),Vt2(V8+B)
5150              MAT Vec_p1= Vec_p1prim
5160              MAT Vec_p2= Vec_p2prim
5170              ASSIGN @F_3 TO "TRANSIT"
5180              ENTER @F_3;Vec_t1(*),Vec_t2(*)
5190              ASSIGN @F_3 TO *
5200 NEXT I
5210 ASSIGN @F_2 TO *
5220 STOP
5230 Esr1:IF ERRN=54 THEN
5240              CLEAR SCREEN
5250              PRINT "FICHIER EXISTE DEJA"
5260              PRINT "VOULEZ-VOUS CANGER DE NOM (O/N) ?"
5270              INPUT Chx$
5280              IF Chx$="N" THEN
5290                ERROR RETURN
5300              ELSE
5310                INPUT "ENTRER LE NOUVEAU NOM",Donnee$
5320                RETURN
5330              END IF
5340 END IF
5350 Esr2:IF ERRN=56 THEN
5360              CLEAR SCREEN
5370              PRINT "FICHIER N'EXISTE PAS DANS LE SOUS REPERTOIRE"
5380              PRINT "DONNEE.BAS"
5390              Flag=1
5400              ERROR RETURN
5410 END IF
5420 RETURN
5430 Esr3:IF ERRN=54 THEN
5440              ERROR RETURN
5450 END IF
5460 END
5470 CSUB Intspl2(X(*),Y(*),A(*),B(*),C(*))
5480 CSUB Optspl2(X(*),Y(*),A(*),B(*),C(*))
5490 CSUB Intspl3_2(X(*),Y(*),A(*),B(*),C(*),D(*))
5500 CSUB Optspl3b_2(X(*),Y(*),A(*),B(*),C(*),D(*))
5510 CSUB Optspl3_2(X(*),Y(*),A(*),B(*),C(*),D(*))
5520 CSUB Errunif(Entre(*),Sortie(*),Var)
5530 CSUB Erru1(Entre(*),Sortie(*),Boucle(*),Var)
5540 CSUB Erru2(Entre(*),Sortie(*),Boucle(*),Var)
5550 CSUB Erru3(Entre(*),Sortie(*),Boucle(*),Var)
5560 CSUB Errnor(Entre(*),Sortie(*),Var)
5570 CSUB Errn1(Entre(*),Sortie(*),Boucle(*),Var)
5580 CSUB Errn2(Entre(*),Sortie(*),Boucle(*),Var)
5590 CSUB Errn3(Entre(*),Sortie(*),Boucle(*),Var)
5600 CSUB Vec_nov(V1_anc(*),V1_nov(*),V1_pla)
5610 CSUB Valspl3_2(X(*),Y(*),A(*),B(*),C(*),D(*),Xm,Ym)
5620 CSUB Der_vspl3_2(X(*),Y(*),A(*),B(*),C(*),D(*),Xm,Ym)
5630 CSUB Valspl2(X(*),Y(*),A(*),B(*),C(*),Xm,Ym)
5640 CSUB Der_vspl2(X(*),Y(*),A(*),B(*),C(*),Xm,Ym)
5650 SUB Eval(V1(*),V2(*),V3(*),V4(*),Vp1(*),Vp2(*),Vt1(*),Vt2(*),Ma1(*),Mb1(*),Mc1(*),Md1(*),Ma2(*),Mb2(*),Mc2(*),Md2(*),
Mv1(*),Mv2(*),Chx_int,Er(*),Retr(*))
5660              OPTION BASE 1
5670 !
5680 !
5690 !*****************************************************************
5700 !
5710 !  PROGRAMME D'EVALUATION
5720 !
```

```
5730 !        ENTREE:  V1, V2, V3, V4, VP1, VP2, VT1, VT2, MA1, MB1
5740 !             MC1, MD1, MA2, MB2, MC2, MD2, MV1, MV2, CHX_INT
5750 !
5760 !        SORTIE: RETR
5770 !
5780 !*****************************************************************
5790 !
5800            G1=SIZE(V1,1)
5810            G2=SIZE(V2,1)
5820            G3=SIZE(V3,1)
5830            G4=SIZE(V4,1)
5840            G5=SIZE(Vp1,1)
5850            G6=G5+3*(G5-1)
5860            G7=SIZE(Vp2,1)
5870            G8=G7+3*(G7-1)
5880            G9=SIZE(Vt1,1)
5890            G10=G9+3*(G9-1)
5900            G11=SIZE(Vt2,1)
5910            G12=G11+3*(G11-1)
5920            M11=SIZE(Mv1,1)
5930            M12=SIZE(Mv1,2)
5940            M21=SIZE(Mv2,1)
5950            M22=SIZE(Mv2,2)
5960            Chx_err=Er(1)
5970            Err_t=Er(2)
5980            Err_v1=Er(3)
5990            Err_v2=Er(4)
6000            Err_p=Er(5)
6010            ALLOCATE Vec_p1per(G1),Vec_p2per(G2),Vec_t1per(G3),Vec_t2per(G4)
6020            ALLOCATE Vec_ptrans(G5),Vec_ttrans(G9)
6030            ALLOCATE Mat_v1per(M11,M12),Mat_v2per(M21,M22)
6040            MAT Vec_p1per= V1
6050            MAT Vec_p2per= V2
6060            MAT Vec_t1per= V3
6070            MAT Vec_t2per= V4
6080            MAT Mat_v1per= Mv1
6090            MAT Mat_v2per= Mv2
6100 !
6110 !
6120 !*****************************************
6130 ! CALCUL DE VEC_P, VEC_T
6140 !*****************************************
6150 !
6160 !
6170            MAT Vec_ptrans= Vp1
6180            Grosseur=G5
6190            FOR J=1 TO G7
6200                Incremente=0
6210                FOR I=1 TO G5
6220                    IF Vp1(I)=Vp2(J) THEN
6230                      Incremente=1
6240                    END IF
6250                NEXT I
6260                IF Incremente=0 THEN
6270                    ALLOCATE V_nov(Grosseur+1)
6280                    Val_nov=Vec_p2(J)
6290                    CALL Vec_nov(Vec_ptrans(*),V_nov(*),Val_nov)
6300                    DEALLOCATE Vec_ptrans(*)
6310                    Grosseur=Grosseur+1
6320                    ALLOCATE Vec_ptrans(Grosseur)
6330                    MAT Vec_ptrans= V_nov
6340                    DEALLOCATE V_nov(*)
6350                END IF
6360            NEXT J
6370            G5=SIZE(Vec_ptrans,1)
6380            G6=G5+1*(G5-1)
```

```
6390        ALLOCATE Vec_p(G6)
6400        FOR J=1 TO G5-1
6410            Vec_p(2*J-1)=Vec_ptrans(J)
6420            Vec_p(2*J)=(Vec_ptrans(J+1)-Vec_ptrans(J))/2+Vec_p(2*J-1)
6430        NEXT J
6440        Vec_p(2*G5-1)=Vec_ptrans(G5)
6450        MAT Vec_ttrans= Vt1
6460        Grosseur=G9
6470        FOR J=1 TO G11
6480            Incremente=0
6490            FOR I=1 TO G9
6500                IF Vt1(I)=Vt2(J) THEN
6510                  Incremente=1
6520                END IF
6530            NEXT I
6540            IF Incremente=0 THEN
6550                ALLOCATE V_nov(Grosseur+1)
6560                Val_nov=Vec_t2(J)
6570                CALL Vec_nov(Vec_ttrans(*),V_nov(*),Val_nov)
6580             DEALLOCATE Vec_ttrans(*)
6590             Grosseur=Grosseur+1
6600             ALLOCATE Vec_ttrans(Grosseur)
6610             MAT Vec_ttrans= V_nov
6620             DEALLOCATE V_nov(*)
6630            END IF
6640        NEXT J
6650        G9=SIZE(Vec_ttrans,1)
6660        G10=G9+1*(G9-1)
6670        ALLOCATE Vec_t(G10),Vec_tper(G10)
6680        FOR J=1 TO G9-1
6690            Vec_t(2*J-1)=Vec_ttrans(J)
6700            Vec_t(2*J)=(Vec_ttrans(J+1)-Vec_ttrans(J))/2+Vec_t(2*J-1)
6710        NEXT J
6720        Vec_t(2*G9-1)=Vec_ttrans(G9)
6730        ALLOCATE Mat_v1c(G6,G10),Mat_v2c(G6,G10),Mat_v1cper(G6,G10)
6740        ALLOCATE Mat_v2cper(G6,G10)
6750        CALL Det_v12(Vec_p(*),Vec_t(*),Vec_p(*),Vec_t(*),Mat_v1c(*),Mat_v2c(*))
6760        !
6770        !*************************************************************************
6780        !
6790        !   AJOUT DES PERTURBATIONS SUR T, V1, V2
6800        !
6810        !*************************************************************************
6820        !
6830        IF Chx_err=1 THEN
6840            CALL Errnor(Vec_t(*),Vec_tper(*),Err_t)
6850            CALL Errnor(Mat_v1c(*),Mat_v1cper(*),Err_v1)
6860            CALL Errnor(Mat_v2c(*),Mat_v2cper(*),Err_v2)
6870        ELSE
6880            CALL Errunif(Vec_t(*),Vec_tper(*),Err_t)
6890            CALL Errunif(Mat_v1c(*),Mat_v1cper(*),Err_v1)
6900            CALL Errunif(Mat_v2c(*),Mat_v2cper(*),Err_v2)
6910        END IF
6920        Vec_tper(1)=Vec_t(1)+Err_t*RND/2
6930        Transit=SIZE(Vec_tper,1)
6940        Vec_tper(Transit)=Vec_t(Transit)-RND/2*Err_t
6950        Transit1=SIZE(Mat_v1c,1)
6960        Transit2=SIZE(Mat_v1c,2)
6970        FOR Boucle=1 TO Transit2
6980            Mat_v1cper(1,Boucle)=Mat_v1c(1,Boucle)+Err_v1*RND/2
6990            Mat_v1cper(Transit1,Boucle)=Mat_v1c(Transit1,Boucle)-Err_v1*RND/2
7000        NEXT Boucle
7010        Transit1=SIZE(Mat_v2c,1)
7020        Transit2=SIZE(Mat_v2c,2)
7030        FOR Boucle=1 TO Transit2
7040            Mat_v2cper(1,Boucle)=Mat_v2c(1,Boucle)+Err_v2*RND/2
```

```
7050            Mat_v2cper(Transit1,Boucle)=Mat_v2c(Transit1,Boucle)-Err_v2*RND/2
7060        NEXT Boucle
7070        !
7080        !
7090        !*************************************************************
7100        !
7110        !  INTERPOLATION DE T
7120        !
7130        !*************************************************************
7140        !
7150        !
7160        ALLOCATE Mat_tp1(G1,G10),Mat_tp2(G2,G10),Xt1(G3),Yv1(G3)
7170        ALLOCATE Xt2(G4),Yv2(G4),Ta1(G3),Tb1(G3),Tc1(G3),Td1(G3),Ta2(G4),Tb2(G4)
7180        ALLOCATE Tc2(G4),Td2(G4)
7190        MAT Xt1= Vec_t1per
7200        MAT Xt2= Vec_t2per
7210        FOR J=1 TO G10
7220            Val_t=Vec_tper(J)
7230            FOR I=1 TO G1
7240                MAT Yv1= Mat_v1per(I,*)
7250                MAT Ta1= Ma1(I,*)
7260                MAT Tb1= Mb1(I,*)
7270                MAT Tc1= Mc1(I,*)
7280                MAT Td1= Md1(I,*)
7290                SELECT Chx_int
7300                CASE 1
7310                    CALL Valspl3_2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Val_t,Ym)
7320                CASE 2
7330                    CALL Valspl3_2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Val_t,Ym)
7340                CASE 3
7350                    CALL Valspl3_2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Val_t,Ym)
7360                CASE 4
7370                    CALL Valspl2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Val_t,Ym)
7380                CASE 5
7390                    CALL Valspl2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Val_t,Ym)
7400                CASE 6
7410                    CALL Valspl3_2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Val_t,Ym)
7420                CASE 7
7430                    CALL Valspl2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Val_t,Ym)
7440                END SELECT
7450                Mat_tp1(I,J)=Ym
7460            NEXT I
7470            FOR I=1 TO G2
7480                MAT Yv2= Mat_v2per(I,*)
7490                MAT Ta2= Ma2(I,*)
7500                MAT Tb2= Mb2(I,*)
7510                MAT Tc2= Mc2(I,*)
7520                MAT Td2= Md2(I,*)
7530                SELECT Chx_int
7540                CASE 1
7550                    CALL Valspl3_2(Xt2(*),Yv2(*),Ta2(*),Tb2(*),Tc2(*),Td2(*),Val_t,Ym)
7560                CASE 2
7570                    CALL Valspl3_2(Xt2(*),Yv2(*),Ta2(*),Tb2(*),Tc2(*),Td2(*),Val_t,Ym)
7580                CASE 3
7590                    CALL Valspl3_2(Xt2(*),Yv2(*),Ta2(*),Tb2(*),Tc2(*),Td2(*),Val_t,Ym)
7600                CASE 4
7610                    CALL Valspl2(Xt2(*),Yv2(*),Ta2(*),Tb2(*),Tc2(*),Val_t,Ym)
7620                CASE 5
7630                    CALL Valspl2(Xt2(*),Yv2(*),Ta2(*),Tb2(*),Tc2(*),Val_t,Ym)
7640                CASE 6
7650                    CALL Valspl3_2(Xt2(*),Yv2(*),Ta2(*),Tb2(*),Tc2(*),Td2(*),Val_t,Ym)
7660                CASE 7
7670                    CALL Valspl2(Xt2(*),Yv2(*),Ta2(*),Tb2(*),Tc2(*),Val_t,Ym)
7680                END SELECT
7690                Mat_tp2(I,J)=Ym
7700            NEXT I
```

```
7710            NEXT J
7720            DEALLOCATE Xt1(*),Yv1(*)
7730            DEALLOCATE Xt2(*),Yv2(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Ta2(*),Tb2(*)
7740            DEALLOCATE Tc2(*),Td2(*)
7750            DEALLOCATE Vec_t1per(*),Vec_t2per(*)
7760            DEALLOCATE Mat_v1c(*),Mat_v2c(*)
7770            DEALLOCATE Mat_v1per(*),Mat_v2per(*)
7780            ALLOCATE A1(G1,G10),B1(G1,G10),C1(G1,G10),D1(G1,G10),A2(G2,G10)
7790            ALLOCATE B2(G2,G10),C2(G2,G10),D2(G2,G10)
7800            ALLOCATE Xp1(G1),Yp1(G1),Xp2(G2),Yp2(G2),Va1(G1),Vb1(G1),Vc1(G1),Vd1(G1)
7810            ALLOCATE Va2(G2),Vb2(G2),Vc2(G2),Vd2(G2)
7820            MAT Yp1= Vec_p1per
7830            MAT Yp2= Vec_p2per
7840            S1=INT(G1+SQRT(2*G1))
7850            S2=INT(G2+SQRT(2*G2))
7860            Sigy1=Err_p/2
7870            Sigy2=Err_p/2
7880            FOR J=1 TO G10
7890             MAT Xp1= Mat_tp1(*,J)
7900                MAT Xp2= Mat_tp2(*,J)
7910                SELECT Chx_int
7920                CASE 1
7930                    CALL Intspl3_2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*),Vd1(*))
7940                    CALL Intspl3_2(Xp2(*),Yp2(*),Va2(*),Vb2(*),Vc2(*),Vd2(*))
7950                    MAT A1(*,J)= Va1
7960                    MAT B1(*,J)= Vb1
7970                    MAT C1(*,J)= Vc1
7980                    MAT D1(*,J)= Vd1
7990                    MAT A2(*,J)= Va2
8000                    MAT B2(*,J)= Vb2
8010                    MAT C2(*,J)= Vc2
8020                    MAT D2(*,J)= Vd2
8030                CASE 2
8040                    CALL Optspl3_2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*),Vd1(*))
8050                    CALL Optspl3_2(Xp2(*),Yp2(*),Va2(*),Vb2(*),Vc2(*),Vd2(*))
8060                    MAT A1(*,J)= Va1
8070                    MAT B1(*,J)= Vb1
8080                    MAT C1(*,J)= Vc1
8090                    MAT D1(*,J)= Vd1
8100                    MAT A2(*,J)= Va2
8110                    MAT B2(*,J)= Vb2
8120                    MAT C2(*,J)= Vc2
8130                    MAT D2(*,J)= Vd2
8140                CASE 3
8150                    CALL Optspl3b_2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*),Vd1(*))
8160                    CALL Optspl3b_2(Xp2(*),Yp2(*),Va2(*),Vb2(*),Vc2(*),Vd2(*))
8170                    MAT A1(*,J)= Va1
8180                    MAT B1(*,J)= Vb1
8190                    MAT C1(*,J)= Vc1
8200                    MAT D1(*,J)= Vd1
8210                    MAT A2(*,J)= Va2
8220                    MAT B2(*,J)= Vb2
8230                    MAT C2(*,J)= Vc2
8240                    MAT D2(*,J)= Vd2
8250                CASE 4
8260                    CALL Intspl2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*))
8270                    CALL Intspl2(Xp2(*),Yp2(*),Va2(*),Vb2(*),Vc2(*))
8280                    MAT A1(*,J)= Va1
8290                    MAT B1(*,J)= Vb1
8300                    MAT C1(*,J)= Vc1
8310                    MAT A2(*,J)= Va2
8320                    MAT B2(*,J)= Vb2
8330                    MAT C2(*,J)= Vc2
8340                CASE 5
8350                    CALL Optspl2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*))
8360                    CALL Optspl2(Xp2(*),Yp2(*),Va2(*),Vb2(*),Vc2(*))
```

```
8370                    MAT A1(*,J)= Va1
8380                    MAT B1(*,J)= Vb1
8390                    MAT C1(*,J)= Vc1
8400                    MAT A2(*,J)= Va2
8410                    MAT B2(*,J)= Vb2
8420                    MAT C2(*,J)= Vc2
8430                CASE 6
8440                    CALL Smospl3(Xp1(*),Yp1(*),S1,Sigy1,Va1(*),Vb1(*),Vc1(*),Vd1(*))
8450                    CALL Smospl3(Xp2(*),Yp2(*),S2,Sigy2,Va2(*),Vb2(*),Vc2(*),Vd2(*))
8460                    MAT A1(*,J)= Va1
8470                    MAT B1(*,J)= Vb1
8480                    MAT C1(*,J)= Vc1
8490                    MAT D1(*,J)= Vd1
8500                    MAT A2(*,J)= Va2
8510                    MAT B2(*,J)= Vb2
8520                    MAT C2(*,J)= Vc2
8530                    MAT D2(*,J)= Vd2
8540                CASE 7
8550                    IF Sigy1=0 THEN
8560                        Sigy1=.000001
8570                    END IF
8580                    IF Sigy2=0 THEN
8590                        Sigy2=.000001
8600                    END IF
8610                    CALL Smospl2a(Xp1(*),Yp1(*),Sigy1,S1,Va1(*),Vb1(*),Vc1(*))
8620                    CALL Smospl2a(Xp2(*),Yp2(*),Sigy2,S2,Va2(*),Vb2(*),Vc2(*))
8630                    MAT A1(*,J)= Va1
8640                    MAT B1(*,J)= Vb1
8650                    MAT C1(*,J)= Vc1
8660                    MAT A2(*,J)= Va2
8670                    MAT B2(*,J)= Vb2
8680                    MAT C2(*,J)= Vc2
8690                END SELECT
8700    NEXT J
8710                !
8720                !
8730                !*************************************************************
8740                !
8750                !           INTERPOLATION DE P
8760                !
8770                !*************************************************************
8780                !
8790                !
8800    ALLOCATE Xv1(G6),Xv2(G6),Mint_p(G6,G10)
8810    MAT Yp1= Vec_p1per
8820    MAT Yp2= Vec_p2per
8830    FOR J=1 TO G10
8840                MAT Xv1= Mat_v1cper(*,J)
8850                MAT Xv2= Mat_v2cper(*,J)
8860                FOR I=1 TO G6
8870                !
8880                !*********************************************************
8890                !
8900                !       CHOIX DU CAPTEUR
8910                !
8920                !*********************************************************
8930                !
8940                    Nb1=INT(Xv1(I)/5)
8950                    Nb2=INT(Xv2(I)/5)
8960                    Val_ajus1=Xv1(I)-Nb1*5
8970                    Val_ajus2=Xv2(I)-Nb2*5
8980                    Val_ajus1=ABS(Val_ajus1-2.5)
8990                    Val_ajus2=ABS(Val_ajus2-2.5)
9000                    IF Val_ajus1<Val_ajus2 THEN
9010                        Chx_v=1
9020                    ELSE
```

```
9030                    Chx_v=2
9040                END IF
9050                IF Chx_v=1 THEN
9060                    ALLOCATE V_int(G1),Yp(G1),A(G1),B(G1),C(G1),D(G1)
9070                    MAT V_int= Mat_tp1(*,J)
9080                    Gro_vec=G1
9090                    MAT Yp= Yp1
9100                    MAT A= A1(*,J)
9110                    MAT B= B1(*,J)
9120                    MAT C= C1(*,J)
9130                    MAT D= D1(*,J)
9140                    Xv=Xv1(I)
9150                ELSE
9160                    ALLOCATE V_int(G2),Yp(G2),A(G2),B(G2),C(G2),D(G2)
9170                    MAT V_int= Mat_tp2(*,J)
9180                    Gro_vec=G2
9190                    MAT Yp= Yp2
9200                    MAT A= A2(*,J)
9210                    MAT B= B2(*,J)
9220                    MAT C= C2(*,J)
9230                    MAT D= D2(*,J)
9240                    Xv=Xv2(I)
9250                END IF
9260                IF Xv>=V_int(1) AND Xv<=V_int(Gro_vec) THEN
9270                    !
9280                    !
9290                    !************************************************
9300                    !       CALCUL DE P
9310                    !************************************************
9320                    !
9330                    !
9340                    SELECT Chx_int
9350                    CASE 1
9360                        CALL Valspl3_2(V_int(*),Yp(*),A(*),B(*),C(*),D(*),Xv,Pcal)
9370                    CASE 2
9380                        CALL Valspl3_2(V_int(*),Yp(*),A(*),B(*),C(*),D(*),Xv,Pcal)
9390                    CASE 3
9400                        CALL Valspl3_2(V_int(*),Yp(*),A(*),B(*),C(*),D(*),Xv,Pcal)
9410                    CASE 4
9420                        CALL Valspl2(V_int(*),Yp(*),A(*),B(*),C(*),Xv,Pcal)
9430                    CASE 5
9440                        CALL Valspl2(V_int(*),Yp(*),A(*),B(*),C(*),Xv,Pcal)
9450                    CASE 6
9460                        CALL Valspl3_2(V_int(*),Yp(*),A(*),B(*),C(*),D(*),Xv,Pcal)
9470                    CASE 7
9480                        CALL Valspl2(V_int(*),Yp(*),A(*),B(*),C(*),Xv,Pcal)
9490                    END SELECT
9500                ELSE
9510                    Pcal=9.9E+99
9520                END IF
9530                IF Xv<V_int(1) OR Xv>V_int(Gro_vec) THEN
9540                    Mint_p(I,J)=9.9E+99
9550                ELSE
9560                    Mint_p(I,J)=Pcal
9570                END IF
9580                DEALLOCATE V_int(*),A(*),B(*),C(*),D(*),Yp(*)
9590            NEXT I
9600        NEXT J
9610            !
9620            !*****************************************************
9630            !
9640            !               CALCUL DES ERREURS
9650            !*****************************************************
9660            !
9670        N=0
9680        Errmax=0
```

```
9690        Errmoy=0
9700        Errmax_trans=0
9710        Pos_p=1
9720        Gr_trans=SIZE(Vec_ptrans,1)
9730        Ecart_reel=0
9740        Ecart_reeltrans=0
9750        FOR J=1 TO G6
9760            FOR I=1 TO G10
9770                M=1
9780                Transit2=(Mint_p(J,I)-Vec_p(J))^2
9790                Transit=ABS((Mint_p(J,I)-Vec_p(J)))/Vec_p(J)
9800                IF Mint_p(J,I)>=1.0E+99 THEN
9810                    Transit=0
9820                    Transit2=0
9830                    M=0
9840                    Mint_p(J,I)=Vec_p(J)
9850                END IF
9860                Errmoy=Errmoy+Transit2
9870                N=N+M
9880                Condition=0
9890                IF ABS(Mint_p(J,I)-Vec_p(J))/Vec_p(J)>Ecart_reeltrans THEN
9900                    FOR Z=1 TO Gr_trans
9910                        IF Vec_ptrans(Z)=Vec_p(J) THEN
9920                          Condition=1
9930                        END IF
9940                    NEXT Z
9950                    IF Condition=0 THEN
9960                        Ecart_reeltrans=ABS(Mint_p(J,I)-Vec_p(J))/Vec_p(J)
9970                        Pos_p=J
9980                        Pos_t=I
9990                    END IF
10000               END IF
10010               IF ABS((Mint_p(J,I)-Vec_p(J)))/Vec_p(J)>Ecart_reel THEN
10020                   Ecart_reel=ABS((Mint_p(J,I)-Vec_p(J)))/Vec_p(J)
10030               END IF
10040               IF ABS(Mint_p(J,I)-Vec_p(J))>Errmax THEN
10050                   Errmax=ABS(Mint_p(J,I)-Vec_p(J))
10060               END IF
10070           NEXT I
10080       NEXT J
10090       Gr_trans=SIZE(Vec_ttrans,1)
10100       Pos_t=1
10110       Err_temp=0
10120       FOR J=1 TO G6
10130         FOR I=1 TO G10
10140               Condition=0
10150               IF ABS(Mint_p(J,I)-Vec_p(J))/Vec_p(J)>Err_temp THEN
10160                   FOR Z=1 TO Gr_trans
10170                       IF Vec_ttrans(Z)=Vec_t(I) THEN
10180                         Condition=1
10190                       END IF
10200                   NEXT Z
10210                   IF Condition=0 THEN
10220                       Err_temp=ABS(Mint_p(J,I)-Vec_p(J))/Vec_p(J)
10230                       Pos_t=I
10240                   END IF
10250               END IF
10260         NEXT I
10270       NEXT J
10280       Errmoy=SQRT(Errmoy/N)
10290       Retr(1)=Vec_p(Pos_p)
10300       Retr(2)=Vec_t(Pos_t)
10310       Retr(3)=Vec_p(Pos_p)
10320       Retr(4)=Vec_t(Pos_t)
10330       Retr(5)=Errmoy
10340       Retr(6)=Errmax
```

```
10350           Retr(7)=Ecart_reel
10360           Retr(8)=1
10370 SUBEND
10380 CSUB Smospl3(REAL X(*),Y(*),S,Sigy,A(*),B(*),C(*),D(*)) 10390 CSUB
Det_v12(Va1(*),Va2(*),Vb1(*),Vb2(*),Ma(*),Mb(*)) 10400 CSUB Smospl2a(REAL
X(*),Y(*),Sigy,S,A(*),B(*),C(*)) 10410 CDEF FNSign(REAL I)
10420 CDEF FNDer1(REAL X0,X1,X2,X3,Y0,Y1,Y2,Y3)
```

Le programme principal pour la procédure 2 (PCR2) est:

```
10          CONTROL 32,2;1
20          STATUS 32,2;Status
30          PRINT "STATUS= ";Status
40          ! RTPRIO=0
50          ! ************************************************************
60          !       PROGRAMME D'ETALLONNAGE
61          !
62          !       PROG2
70          !************************************************************
71          !
80          OPTION BASE 1
90          DIM Donnee$[30],Donnee_sufx$[30],Er(5),Resul$[30]
100         INPUT "VOULEZ-VOUS RECUPERER UN FICHIER DE DONNES (O/N)",Rep$
110         !
120         !************************************************************
130         !
140         !  LES FICHIERS DE LECTURE OU D'ECRITURE SONT DE TYPE ASCII
150         !
160         !************************************************************
170         !
180         IF Rep$="O" THEN
190             LOOP
200                 Flag=0
210                 INPUT "ENTRER LE NOM DU FICHIER DE DONNEES A RECUPERER",Donnee$
220                 Donnee$="DONNEE.BAS/"&Donnee$
230                 ON ERROR GOSUB Esr2
240                 ASSIGN @F_1 TO Donnee$
250                 OFF ERROR
260             EXIT IF Flag=0
270             END LOOP
280             !
290             !************************************************************
300             !
310             ! LE FICHIER DOIT CONTENIR DANS L'ORDRE SUIVANT 16 VALEURS REELS
320             ! DECRIVANT LE NOMBRE
330             ! D'ELEMENTS DES VECTEURS ET DES MATRICES, 8 VECTEURS, 10 MATRICES
340             ! ET 6 VALEURS REELS.
350             !
360             !************************************************************
370             !
380             ENTER @F_1;K1,K2,K3,K4,V5,V6,V7,V8,M11,M12,M21,M22,M31,M32,M41,M42
390             ALLOCATE Vec_p1prim(V5),Vec_p2prim(V6)
400             ALLOCATE Vec_p1per(K1),Vec_p2per(K2),Vec_t1per(K3),Vec_t2per(K4)
410             ALLOCATE Vec_p1(V5),Vec_p2(V6),Vec_t1(V7),Vec_t2(V8)
420             ALLOCATE Mat_at1(M11,M12),Mat_bt1(M11,M12),Mat_ct1(M21,M22),Mat_dt1(M21,M22)
430             ALLOCATE Mat_at2(M21,M22),Mat_bt2(M21,M22),Mat_ct2(M21,M22),Mat_dt2(M21,M22)
440             ALLOCATE Mat_v2per(M41,M42),Mat_v1per(M31,M32),Mat_v1(2,2),Mat_v2(2,2)
450             ALLOCATE V1(K1),V2(K2),V3(K3),V4(K4),Ma1(M11,M12),Mb1(M11,M12)
```

```
460             ALLOCATE Mc1(M11,M12),Md1(M11,M12),Ma2(M21,M22),Mb2(M21,M22)
470             ALLOCATE Mc2(M21,M22),Md2(M21,M22),Mv1(M31,M32),Mv2(M41,M42)
480             ALLOCATE Vp1(V5),Vp2(V6),Vt1(V7),Vt2(V8)
490             !
500             !********************* ALLOCATION POUR DONNEE PRIMAIRE ***************
510             !
520             ENTER @F_1;Vec_p1per(*),Vec_p2per(*),Vec_t1per(*),Vec_t2per(*)
530             ENTER @F_1;Vec_p1(*),Vec_p2(*),Vec_t1(*),Vec_t2(*)
540             ENTER @F_1;Mat_at1(*),Mat_bt1(*),Mat_ct1(*),Mat_dt1(*)
550             ENTER @F_1;Mat_at2(*),Mat_bt2(*),Mat_ct2(*),Mat_dt2(*)
560             ENTER @F_1;Mat_v1per(*),Mat_v2per(*)
570             ENTER @F_1;Chx_err,Erreur_v1,Erreur_v2,Erreur_t,Erreur_p,Chx_int
580             ASSIGN @F_1 TO *
590             Saute=1
600             Exis_fich=1
610         ELSE
620             INPUT "ENTRER Tmin",Tmin
630             INPUT "ENTER Tmax",Tmax
640             INPUT "ENTRER Pmin",Pmin
650             INPUT "ENTRER Pmax",Pmax
660             LOOP
670              INPUT "ERREUR POUR ETAL DONNEE PAR LOI NORMALE OU UNIFIRME (1/0)",Chx_err
680              IF Chx_err=1 THEN
690                     INPUT "ENTRER VARIANCE DE P",Var_p
700                     Erreur_p=Var_p
710                     INPUT "ENTRER VARIANCE DE T",Var_t
720                     Erreur_t=Var_t
730                     INPUT "ENTRER VARIANCE DE V1",Var_v1
740                     Erreur_v1=Var_v1
750                     INPUT "ENTRER VARIANCE DE V2",Var_v2
760                     Erreur_v2=Var_v2
770                     Sortir=1
780              ELSE
790                     IF Chx_err=0 THEN
800                         INPUT "ENTRER DELTA P",Delta_p
810                         Erreur_p=Delta_p
820                         INPUT "ENTRER DELTA T",Delta_t
830                         Erreur_t=Delta_t
840                         INPUT "ENTRER DELTA V1",Delta_v1
850                         Erreur_v1=Delta_v1
860                         INPUT "ENTRER DELTA V2",Delta_v2
870                         Erreur_v2=Delta_v2
880                         Sortir=1
890                     ELSE
900                         Sortir=0
910                     END IF
920              END IF
930             EXIT IF Sortir=1
940             END LOOP
950             Saute=0
960             Exis_fich=0
970         END IF
980         INPUT "ERREUR POUR EVAL DONNEE PAR LOI NORMALE OU UNIFIRME (1/0)",Cerr
990         INPUT "ENTRER LA VALEUR DE LA VARIANCE OU ERREUR UNIF POUR T",Err_t
1000 INPUT "ENTRER LA VALEUR DE LA VARIANCE OU ERREUR UNIF POUR V1",Err_v1
1010 INPUT "ENTRER LA VALEUR DE LA VARIANCE OU ERREUR UNIF POUR V2",Err_v2 1020  Er(1)=Cerr
1030 Er(2)=Err_t
1040 Er(3)=Err_v1
1050 Er(4)=Err_v2
1060 Er(5)=Erreur_p
1070 Sortir=0
1080 CLEAR SCREEN
1090 PRINT " LES INTERPOLATIONS POSSIBLES QUE PERMET CE PROGRAMME SONT:"
1100 PRINT
1110 PRINT "            1) INTSPL3"
1120 PRINT "            2) OPTSPL3"
```

```
1130 PRINT "            3) OPTSPL3_B"
1140 PRINT "            4) INTSPL2"
1150 PRINT "            5) OPTSPL2"
1160 PRINT "            6) SMOSPL3"
1161 PRINT "            7) SMOSPL2"
1170 LOOP
1180            INPUT " ENTRER VOTRE CHOIX",Chx_int
1190            IF Chx_int<=7 THEN
1200             IF Chx_int>=1 THEN
1210                    Sortir=1
1220             END IF
1230            END IF
1240 EXIT IF Sortir=1
1250 END LOOP
1260 CLEAR SCREEN
1270 INPUT "ENTRER LE NOMBRE D'ITERATION DE TEMPERATURE DESIREE",Nbr_temp
1280 INPUT "ENTRER LE NOMBRE D'ITERATION DE PRESSION DESIREE",Nbr_pres
1290 INPUT "ENTRER LE NOM DU FICHIER DE SAUVEGARDE DES DONNEES",Donnee$
1300 Donnee$="DONNEE.BAS/"&Donnee$
1310 INPUT "ENTRER LE NOM DU FICHIER POUR SAUVEGARDE DES RESULTATS",Resul$
1320 Resul$="RESULTAT.BAS/"&Resul$
1330 ON ERROR GOSUB Esr1
1340 CREATE ASCII Resul$,10
1350 OFF ERROR
1360 ASSIGN @F_2 TO Resul$
1370 Total_iter=Nbr_temp*Nbr_pres
1380 OUTPUT @F_2;Chx_int,Chx_err,Erreur_p,Erreur_t,Erreur_v1,Erreur_v2
1390 !
1400 !*****************************************************************
1410 !
1420 !    SI FICHIER DE DONNEE N'EXISTE PAS ALORS CALCUL DE
1430 !    VEC_P1, VEC_P2, MAT_V1, MAT_V2, MAT_V1PER
1440 !    MAT_V2PER, MAT_P1PER, MAT_P2PER, VEC_T1, VEC_T1PER, VEC_T2
1450 !    VEC_T2PER, MAT_AT1, MAT_BT1, MAT_CT1, MAT_DT1, MAT_AT2, MAT_BT2
1460 !    MAT_CT2, MAT_DT2
1470 !
1480 !*****************************************************************
1490 !
1500 IF Saute=0 THEN
1510            Dim_p=5
1520            Dim_t=5
1530            ALLOCATE Vec_p1prim(Dim_p),Vec_p2prim(Dim_p),Vec_p1(Dim_p),Vec_p2(Dim_p),Vec_p1per(Dim_p),Vec_p2per(Dim
1540            ALLOCATE Vec_t1(Dim_t),Vec_t2(Dim_t),Vec_t1per(Dim_t),Vec_t2per(Dim_t)
1550            ALLOCATE Mat_v1(Dim_p,Dim_t),Mat_v2(Dim_p,Dim_t),Mat_v1per(Dim_p,Dim_t),Mat_v2per(Dim_p,Dim_t)
1560            ALLOCATE Mat_at1(Dim_p,Dim_t),Mat_bt1(Dim_p,Dim_t),Mat_ct1(Dim_p,Dim_t),Mat_dt1(Dim_p,Dim_t)
1570            ALLOCATE Mat_at2(Dim_p,Dim_t),Mat_bt2(Dim_p,Dim_t),Mat_ct2(Dim_p,Dim_t),Mat_dt2(Dim_p,Dim_t)
1580            ALLOCATE V1(Dim_p),V2(Dim_p),V3(Dim_t),V4(Dim_t),Ma1(Dim_p,Dim_t),Mb1(Dim_p,Dim_t),Mc1(Dim_p,Dim_t)
1590            ALLOCATE Md1(Dim_p,Dim_t),Ma2(Dim_p,Dim_t),Mb2(Dim_p,Dim_t),Mc2(Dim_p,Dim_t),Md2(Dim_p,Dim_t),Mv1(Dim_p
1600            ALLOCATE Mv2(Dim_p,Dim_t),Vp1(Dim_p),Vp2(Dim_p),Vt1(Dim_t),Vt2(Dim_t)
1610 !
1620 !*****************************************************************
1630 !
1640 ! CALCUL DE V1, V2, T1 ET T2
1650 !
1660 !*****************************************************************
1670 !
1680            Pas_temp=(Tmax-Tmin)/(Dim_t-1)
1690            FOR I=1 TO Dim_t
1700             Vec_t1(I)=Tmin+(I-1)*Pas_temp
1710             Vec_t2(I)=Tmin+(I-1)*Pas_temp
1720            NEXT I
1730            Pas_pres=(Pmax-Pmin)/(Dim_p-1)
1740            FOR I=1 TO Dim_p
1750             Vec_p1(I)=Pmin+(I-1)*Pas_pres
1760             Vec_p2(I)=Pmin+(I-1)*Pas_pres
1770            NEXT I
```

```
1780            V5=Dim_p
1790            V7=Dim_t
1800 END IF
1810 DIM Rtr(8)
1820 OUTPUT @F_2;Err_t,Err_v1,Err_v2,Vec_p1(1),Vec_p1(V5),Vec_t1(1),Vec_t1(V7)
1830 OUTPUT @F_2;Total_iter
1840 Fin=0
1850 CLEAR SCREEN
1860 !
1870 !*************************************************************************
1880 !
1890 !     GARDE EN MEMOIRE DES DONNEES PRIMAIRE
1900 !
1910 !*************************************************************************
1920 !
1930 RANDOMIZE
1940 MAT Vec_p1prim= Vec_p1
1950 MAT Vec_p2prim= Vec_p2
1960 OUTPUT @F_2;Nbr_temp,Nbr_pres
1970 FOR I=1 TO Nbr_temp
1980  PRINT "NBR_TEMP=";I
1990  FOR J=1 TO Nbr_pres
2000   IF Saute=0 THEN
2010                    CALL Det_v1(Vec_p1(*),Vec_t1(*),Mat_v1(*))
2020                    IF Chx_err=1 THEN
2030                        CALL Errnor(Vec_p1(*),Vec_p1per(*),Var_p)
2070                        CALL Errnor(Vec_p2(*),Vec_p2per(*),Var_p)
2110                        CALL Errnor(Vec_t1(*),Vec_t1per(*),Var_t)
2150                        CALL Errnor(Vec_t2(*),Vec_t2per(*),Var_t)
2260                    ELSE
2270                        CALL Errunif(Vec_p1(*),Vec_p1per(*),Delta_p)
2310                        CALL Errunif(Vec_p2(*),Vec_p2per(*),Delta_p)
2350                        CALL Errunif(Vec_t1(*),Vec_t1per(*),Delta_t)
2390                        CALL Errunif(Vec_t2(*),Vec_t2per(*),Delta_t)
2500                    END IF
2501                    CALL Det_v1(Vec_p1per(*),Vec_t1per(*),Mat_v1per(*))
2506                    IF Chx_err=1 THEN
2507                        CALL Errnor(Mat_v1per(*),Mat_v1per(*),Var_v1)
2508                        Transit1=SIZE(Mat_v1,1)
2509                        Transit2=SIZE(Mat_v1,2)
2510                        FOR Boucle=1 TO Transit2
2511                            Mat_v1per(1,Boucle)=Mat_v1(1,Boucle)-Var_v1*RND/2
2512                            Mat_v1per(Transit1,Boucle)=Mat_v1(Transit1,Boucle)+Var_v1*RND/2
2513                        NEXT Boucle
2514                    ELSE
2515                        CALL Errunif(Mat_v1per(*),Mat_v1per(*),Delta_v1)
2516                        Transit1=SIZE(Mat_v1,1)
2517                        Transit2=SIZE(Mat_v1,2)
2518                        FOR Boucle=1 TO Transit2
2519                            Mat_v1per(1,Boucle)=Mat_v1(1,Boucle)-Delta_v1*RND/2
2520                            Mat_v1per(Transit1,Boucle)=Mat_v1(Transit1,Boucle)+Delta_v1*RND/2
2521                        NEXT Boucle
2522                    END IF
2523                    Longeur=SIZE(Vec_t1per,1)
2524                    N=SIZE(Vec_p1per,1)
2525                    ALLOCATE Var1(Longeur),Var2a(Longeur),Var2b(Longeur),Var2c(Longeur),Var2d(Longeur)
2526                    FOR K=1 TO N
2527                        MAT Var1= Mat_v1per(K,*)
2528                        SELECT Chx_int
2529                        CASE 1
2530                            CALL Intspl3_2(Vec_t1(*),Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*))
2531                            MAT Mat_at1(K,*)= Var2a
2532                            MAT Mat_bt1(K,*)= Var2b
2533                            MAT Mat_ct1(K,*)= Var2c
2534                            MAT Mat_dt1(K,*)= Var2d
2535                        CASE 2
```

```
                        CALL Optspl3_2(Vec_t1(*),Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*))
                        MAT Mat_at1(K,*)= Var2a
                        MAT Mat_bt1(K,*)= Var2b
                        MAT Mat_ct1(K,*)= Var2c
                        MAT Mat_dt1(K,*)= Var2d
                    CASE 3
                        CALL Optspl3b_2(Vec_t1(*),Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*))
                        MAT Mat_at1(K,*)= Var2a
                        MAT Mat_bt1(K,*)= Var2b
                        MAT Mat_ct1(K,*)= Var2c
                        MAT Mat_dt1(K,*)= Var2d
                    CASE 4
                        CALL Intspl2(Vec_t1(*),Var1(*),Var2a(*),Var2b(*),Var2c(*))
                        MAT Mat_at1(K,*)= Var2a
                        MAT Mat_bt1(K,*)= Var2b
                        MAT Mat_ct1(K,*)= Var2c
                    CASE 5
                        CALL Optspl2(Vec_t1(*),Var1(*),Var2a(*),Var2b(*),Var2c(*))
                        MAT Mat_at1(K,*)= Var2a
                        MAT Mat_bt1(K,*)= Var2b
                        MAT Mat_ct1(K,*)= Var2c
                    CASE 6
                        S=INT(Longeur+SQRT(2*Longeur))
                        Sigy=Erreur_v1/2
                        CALL Smospl3(Vec_t1(*),Var1(*),S,Sigy,Var2a(*),Var2b(*),Var2c(*),Var2d(*))
                        MAT Mat_at1(K,*)= Var2a
                        MAT Mat_bt1(K,*)= Var2b
                        MAT Mat_ct1(K,*)= Var2c
                        MAT Mat_dt1(K,*)= Var2d
                    CASE 7
                        S=INT(Longeur+SQRT(2*Longeur))
                        Sigy=Erreur_v1/2
                        IF Sigy=0 THEN
                          Sigy=.000001
                        END IF
                        CALL Smospl2a(Vec_t1(*),Var1(*),Sigy,S,Var2a(*),Var2b(*),Var2c(*))
                        MAT Mat_at1(K,*)= Var2a
                        MAT Mat_bt1(K,*)= Var2b
                        MAT Mat_ct1(K,*)= Var2c
                    END SELECT
                NEXT K
                DEALLOCATE Var1(*),Var2a(*),Var2b(*),Var2c(*),Var2d(*)
!
!
!*****************************************************************
!
!                       SAUVEGARDE DES DONNEES
!
!*****************************************************************
!
                V5=SIZE(Vec_p1,1)
                V6=SIZE(Vec_p2,1)
                V7=SIZE(Vec_t1,1)
                V8=SIZE(Vec_t2,1)
                M11=SIZE(Mat_at1,1)
                M12=SIZE(Mat_at1,2)
                M21=SIZE(Mat_at2,1)
                M22=SIZE(Mat_at2,2)
                M31=SIZE(Mat_v1per,1)
                M32=SIZE(Mat_v1per,2)
!   IF J=Nbr_pres THEN
!               Sufx1$=VAL$(V7)
!               Sufx2$=VAL$(V5)
!               Sufx$=".T"&Sufx1$&"P"&Sufx2$
!               Donnee_sufx$=Donnee$&Sufx$
!               ON ERROR GOSUB Esr3
```

```
2605 !                  CREATE ASCII Donnee_sufx$,10
2606 !                  OFF ERROR
2607 !                  ASSIGN @F_1 TO Donnee_sufx$
2608 !                  OUTPUT @F_1;V5,V6,V7,V8,V5,V6,V7,V8,M11,M12,M21,M22,M31,M32
2609 !                  OUTPUT @F_1;M41,M42,Vec_p1per(*),Vec_p2per(*),Vec_t1per(*)
2610 !                  OUTPUT @F_1;Vec_t2per(*),Vec_p1(*),Vec_p2(*),Vec_t1(*),Vec_t2(*)
2611 !                  OUTPUT @F_1;Mat_at1(*),Mat_bt1(*),Mat_ct1(*),Mat_dt1(*),Mat_at2(*)
2612 !                  OUTPUT @F_1;Mat_bt2(*),Mat_ct2(*),Mat_dt2(*),Mat_v1per(*),Mat_v2per(*)
2613 !                  OUTPUT @F_1;Chx_err,Erreur_v1,Erreur_v2,Erreur_t,Erreur_p,Chx_int
2614 !                  ASSIGN @F_1 TO *
2615 !      END IF
2616      END IF
2617      Saute=0
2618      MAT V1= Vec_p1per
2619      MAT V2= Vec_p2per
2620      MAT V3= Vec_t1per
2621      MAT V4= Vec_t2per
2622                  MAT Ma1= Mat_at1
2623                  MAT Mb1= Mat_bt1
2624                  MAT Mc1= Mat_ct1
2625                  MAT Md1= Mat_dt1
2626                  MAT Mv1= Mat_v1per
2627                  MAT Vp1= Vec_p1
2628                  MAT Vp2= Vec_p2
2629                  MAT Vt1= Vec_t1
2630                  MAT Vt2= Vec_t2
2631                  Cint=Chx_int
2632                  CALL Eval(V1(*),V2(*),V3(*),V4(*),Vp1(*),Vp2(*),Vt1(*),Vt2(*),Ma1(*),Mb1(*),Mc1(*),Md1(*),Mv1(*),Cint,
Er(*),Rtr(*))
2633                  P1n=Rtr(1)
2634                  T1n=Rtr(2)
2635                  P2n=Rtr(3)
2636                  T2n=Rtr(4)
2637                  Errmoy=Rtr(5)
2638                  Errmax=Rtr(6)
2639                  Ecart_reel=Rtr(7)
2640                  !PRINT "ERRMAX= ";Errmax
2641                  !PRINT "ERREUR RELAT. MAX.= ";Ecart_reel
2642                  !PRINT "ERRMOY= ";Errmoy
2643                  !PRINT "   "
2644                  !PRINT "PRES NOV= ";P1n
2645                  !PRINT "TEMP NOV= ";T1n
2646                  !PRINT "   "
2647                  Fin=Rtr(8)
2648                  B=5
2649                  ALLOCATE Vec_p1n(V5+B),Vec_p2n(V6+B),Vec_t1n(V7),Vec_t2n(V8)
2650                  MAT Vec_t1n= Vec_t1
2651                  MAT Vec_t2n= Vec_t2
2652 !                CALL Vec_nov(Vec_p1(*),Vec_p1n(*),P1n)
2653 !                CALL Vec_nov(Vec_p2(*),Vec_p2n(*),P2n)
2654                  Pas_pres=(Pmax-Pmin)/(V5+B-1)
2655                  FOR Kk=1 TO V5+B
2656                    Vec_p1n(Kk)=Pmin+(Kk-1)*Pas_pres
2657                  NEXT Kk
2658                  Pas_pres=(Pmax-Pmin)/(V6+B-1)
2659                  FOR Kk=1 TO V6+B
2660                    Vec_p2n(Kk)=Pmin+(Kk-1)*Pas_pres
2661                  NEXT Kk
2662                  ON ERROR GOSUB Esr3
2663                  CREATE ASCII "TRANSIT",10
2664                  OFF ERROR
2665                  ASSIGN @F_3 TO "TRANSIT"
2666                  OUTPUT @F_3;Vec_p1n(*),Vec_p2n(*),Vec_t1n(*),Vec_t2n(*)
2667                  ASSIGN @F_3 TO *
2668                  DEALLOCATE Vec_p1n(*),Vec_p2n(*),Vec_t1n(*),Vec_t2n(*)
2669                  DEALLOCATE Vec_p1(*),Vec_p2(*),Vec_t1(*),Vec_t2(*),Vec_p1per(*)
```

```
2670                DEALLOCATE Vec_p2per(*),Vec_t1per(*),Vec_t2per(*),Mat_v1(*),Mat_v2(*)
2671                DEALLOCATE Mat_v1per(*),Mat_v2per(*),Mat_at1(*),Mat_bt1(*),Mat_ct1(*)
2672                DEALLOCATE Mat_dt1(*),Mat_at2(*),Mat_bt2(*),Mat_ct2(*),Mat_dt2(*)
2673                DEALLOCATE V1(*),V2(*),V3(*),V4(*),Ma1(*),Mb1(*),Mc1(*),Md1(*)
2674                DEALLOCATE Ma2(*),Mb2(*),Mc2(*),Md2(*),Mv1(*),Mv2(*),Vp1(*),Vp2(*),Vt1(*),Vt2(*)
2675                ALLOCATE Vec_p1(V5+B),Vec_p2(V6+B),Vec_p1per(V5+B),Vec_p2per(V6+B)
2676                ALLOCATE Vec_t1(V7),Vec_t2(V8),Vec_t1per(V7),Vec_t2per(V8)
2677                ALLOCATE Mat_v1(V5+B,V7),Mat_v2(V6+B,V8),Mat_v1per(V5+B,V7),Mat_v2per(V6+B,V8)
2678                ALLOCATE Mat_at1(V5+B,V7),Mat_bt1(V5+B,V7),Mat_ct1(V5+B,V7),Mat_dt1(V5+B,V7)
2679                ALLOCATE Mat_at2(V6+B,V8),Mat_bt2(V6+B,V8),Mat_ct2(V6+B,V8),Mat_dt2(V6+B,V8)
2680                ALLOCATE V1(V5+B),V2(V6+B),V3(V7),V4(V8),Ma1(V5+B,V7),Mb1(V5+B,V7),Mc1(V5+B,V7)
2681                ALLOCATE Md1(V5+B,V7),Ma2(V6+B,V8),Mb2(V6+B,V8),Mc2(V6+B,V8),Md2(V6+B,V8),Mv1(V5+B,V7)
2682                ALLOCATE Mv2(V6+B,V8),Vp1(V5+B),Vp2(V6+B),Vt1(V7),Vt2(V8)
2684                ASSIGN @F_3 TO "TRANSIT"
2685                ENTER @F_3;Vec_p1(*),Vec_p2(*),Vec_t1(*),Vec_t2(*)
2686                ASSIGN @F_3 TO *
2687   !
2688   !
2689             !*****************************************************************
2690             !
2691             !   ECRITURE DES RESULTATS DANS LE FICHIER ADEQUAT
2692             !
2693             !
2694             !*****************************************************************
2695             !
2696             !
2697               OUTPUT @F_2;V5,V7,V6,V8,Errmax,Errmoy,Ecart_reel
2698             NEXT J
2699             V5=SIZE(Vec_p1prim,1)
2700             V6=SIZE(Vec_p2prim,1)
2701             ALLOCATE Vec_t1n(V7+B),Vec_t2n(V8+B)
2702 !  CALL Vec_nov(Vec_t1(*),Vec_t1n(*),T1n) 2703 !  CALL
Vec_nov(Vec_t2(*),Vec_t2n(*),T2n) 2704 Pas_temp=(Tmax-
Tmin)/(V7+B-1)
2705             FOR Kk=1 TO V7+B
2706               Vec_t1n(Kk)=Tmin+(Kk-1)*Pas_temp
2707             NEXT Kk
2708             Pas_temp=(Tmax-Tmin)/(V8+B-1)
2709             FOR Kk=1 TO V8+B
2710               Vec_t2n(Kk)=Tmin+(Kk-1)*Pas_temp
2711             NEXT Kk
2712             ASSIGN @F_3 TO "TRANSIT"
2713             OUTPUT @F_3;Vec_t1n(*),Vec_t2n(*)
2714             ASSIGN @F_3 TO *
2715             DEALLOCATE Vec_t1n(*),Vec_t2n(*)
2716             DEALLOCATE Vec_p1(*),Vec_p2(*),Vec_t1(*),Vec_t2(*),Vec_p1per(*)
2717             DEALLOCATE Vec_p2per(*),Vec_t1per(*),Vec_t2per(*),Mat_v1(*),Mat_v2(*)
2718             DEALLOCATE Mat_v1per(*),Mat_v2per(*),Mat_at1(*),Mat_bt1(*),Mat_ct1(*)
2719             DEALLOCATE Mat_dt1(*),Mat_at2(*),Mat_bt2(*),Mat_ct2(*),Mat_dt2(*)
2720             DEALLOCATE V1(*),V2(*),V3(*),V4(*),Ma1(*),Mb1(*),Mc1(*),Md1(*)
2721             DEALLOCATE Ma2(*),Mb2(*),Mc2(*),Md2(*),Mv1(*),Mv2(*)
2722             DEALLOCATE Vp1(*),Vp2(*),Vt1(*),Vt2(*)
2723             ALLOCATE Vec_p1(V5),Vec_p2(V6),Vec_p1per(V5),Vec_p2per(V6)
2724             ALLOCATE Vec_t1(V7+B),Vec_t2(V8+B),Vec_t1per(V7+B),Vec_t2per(V8+B)
2725             ALLOCATE Mat_v1(V5,V7+B),Mat_v2(V6,V8+B),Mat_v1per(V5,V7+B),Mat_v2per(V6,V8+B)
2726             ALLOCATE Mat_at1(V5,V7+B),Mat_bt1(V5,V7+B),Mat_ct1(V5,V7+B),Mat_dt1(V5,V7+B)
2727             ALLOCATE Mat_at2(V6,V8+B),Mat_bt2(V6,V8+B),Mat_ct2(V6,V8+B),Mat_dt2(V6,V8+B)
2728             ALLOCATE V1(V5),V2(V6),V3(V7+B),V4(V8+B),Ma1(V5,V7+B),Mb1(V5,V7+B),Mc1(V5,V7+B)
2729             ALLOCATE Md1(V5,V7+B),Ma2(V6,V8+B),Mb2(V6,V8+B),Mc2(V6,V8+B),Md2(V6,V8+B),Mv1(V5,V7+B)
2730             ALLOCATE Mv2(V6,V8+B),Vp1(V5),Vp2(V6),Vt1(V7+B),Vt2(V8+B)
2731             MAT Vec_p1= Vec_p1prim
2732             MAT Vec_p2= Vec_p2prim
2733             ASSIGN @F_3 TO "TRANSIT"
2734             ENTER @F_3;Vec_t1(*),Vec_t2(*)
2735             ASSIGN @F_3 TO *
2736 NEXT I
```

```
2737 ASSIGN @F_2 TO *
2738 STOP
2739 Esr1:IF ERRN=54 THEN
2740              CLEAR SCREEN
2741              PRINT "FICHIER EXISTE DEJA"
2742              PRINT "VOULEZ-VOUS CANGER DE NOM (O/N) ?"
2743              INPUT Chx$
2744              IF Chx$="N" THEN
2745                ERROR RETURN
2746              ELSE
2747                INPUT "ENTRER LE NOUVEAU NOM",Donnee$
2748                RETURN
2749              END IF
2750 END IF
2751 Esr2:IF ERRN=56 THEN
2752              CLEAR SCREEN
2753              PRINT "FICHIER N'EXISTE PAS DANS LE SOUS REPERTOIRE"
2754              PRINT "DONNEE.BAS"
2755              Flag=1
2756              ERROR RETURN
2757 END IF
2758 RETURN
2759 Esr3:IF ERRN=54 THEN
2760              ERROR RETURN
2761 END IF
2762 END
2763 CSUB Intspl2(X(*),Y(*),A(*),B(*),C(*))
2764 CSUB Optspl2(X(*),Y(*),A(*),B(*),C(*))
2765 CSUB Intspl3_2(X(*),Y(*),A(*),B(*),C(*),D(*))
2766 CSUB Optspl3b_2(X(*),Y(*),A(*),B(*),C(*),D(*))
2767 CSUB Optspl3_2(X(*),Y(*),A(*),B(*),C(*),D(*))
2768 CSUB Errunif(Entre(*),Sortie(*),Var)
2769 CSUB Erru1(Entre(*),Sortie(*),Boucle(*),Var)
2770 CSUB Erru2(Entre(*),Sortie(*),Boucle(*),Var)
2771 CSUB Erru3(Entre(*),Sortie(*),Boucle(*),Var)
2772 CSUB Errnor(Entre(*),Sortie(*),Var)
2773 CSUB Errn1(Entre(*),Sortie(*),Boucle(*),Var)
2774 CSUB Errn2(Entre(*),Sortie(*),Boucle(*),Var)
2775 CSUB Errn3(Entre(*),Sortie(*),Boucle(*),Var)
2776 CSUB Vec_nov(V1_anc(*),V1_nov(*),V1_pla)
2777 CSUB Valspl3_2(X(*),Y(*),A(*),B(*),C(*),D(*),Xm,Ym)
2778 CSUB Der_vspl3_2(X(*),Y(*),A(*),B(*),C(*),D(*),Xm,Ym)
2779 CSUB Valspl2(X(*),Y(*),A(*),B(*),C(*),Xm,Ym)
2780 CSUB Der_vspl2(X(*),Y(*),A(*),B(*),C(*),Xm,Ym)
2781 SUB Eval(V1(*),V2(*),V3(*),V4(*),Vp1(*),Vp2(*),Vt1(*),Vt2(*),Ma1(*),Mb1(*),Mc1(*),Md1(*),Mv1(*),Chx_int,Er(*),Retr(*))
2782              OPTION BASE 1
2783 !
2784 !
2785 !*****************************************************************
2786 !
2787 !  PROGRAMME D'EVALUATION
2788 !
2789 !         ENTREE:  V1, V2, V3, V4, VP1, VP2, VT1, VT2, MA1, MB1
2790 !                  MC1, MD1, MA2, MB2, MC2, MD2, MV1, MV2, CHX_INT
2791 !
2792 !         SORTIE: RETR
2793 !
2794 !*****************************************************************
2795 !
2796              G1=SIZE(V1,1)
2797              G2=SIZE(V2,1)
2798              G3=SIZE(V3,1)
2799              G4=SIZE(V4,1)
2800              G5=SIZE(Vp1,1)
2801              G6=G5+3*(G5-1)
2802              G7=SIZE(Vp2,1)
```

```
2803            G8=G7+3*(G7-1)
2804            G9=SIZE(Vt1,1)
2805            G10=G9+3*(G9-1)
2806            G11=SIZE(Vt2,1)
2807            G12=G11+3*(G11-1)
2808            M11=SIZE(Mv1,1)
2809            M12=SIZE(Mv1,2)
2810            Chx_err=Er(1)
2811            Err_t=Er(2)
2812            Err_v1=Er(3)
2813            Err_v2=Er(4)
2814            Err_p=Er(5)
2815            ALLOCATE Vec_p1per(G1),Vec_p2per(G2),Vec_t1per(G3),Vec_t2per(G4)
2816            ALLOCATE Vec_p1(G5),Vec_p2(G7),Vec_t1(G9),Vec_t2(G11)
2817            ALLOCATE Vec_ptrans(G5),Vec_ttrans(G9)
2818            ALLOCATE Mat_v1per(M11,M12),Mat_v2per(M11,M12)
2819            MAT Vec_p1per= V1
2820            MAT Vec_p2per= V2
2821            MAT Vec_t1per= V3
2822            MAT Vec_t2per= V4
2823            MAT Mat_v1per= Mv1
2824            MAT Vec_p1= Vp1
2825            MAT Vec_p2= Vp2
2826            MAT Vec_t1= Vt1
2827            MAT Vec_t2= Vt2
2829 !
2830 !
2831 !*****************************************
2832 ! CALCUL DE VEC_P, VEC_T
2833 !*****************************************
2834 !
2835 !
2836            MAT Vec_ptrans= Vp1
2837            Grosseur=G5
2838            FOR J=1 TO G7
2839                Incremente=0
2840                FOR I=1 TO G5
2841                    IF Vp1(I)=Vp2(J) THEN
2842                      Incremente=1
2843                    END IF
2844                NEXT I
2845                IF Incremente=0 THEN
2846                    ALLOCATE V_nov(Grosseur+1)
2847                    Val_nov=Vec_p2(J)
2848                    CALL Vec_nov(Vec_ptrans(*),V_nov(*),Val_nov)
2849                    DEALLOCATE Vec_ptrans(*)
2850                    Grosseur=Grosseur+1
2851                    ALLOCATE Vec_ptrans(Grosseur)
2852                    MAT Vec_ptrans= V_nov
2853                    DEALLOCATE V_nov(*)
2854                END IF
2855            NEXT J
2856            G5=SIZE(Vec_ptrans,1)
2857            G6=G5+1*(G5-1)
2858            ALLOCATE Vec_p(G6)
2859            FOR J=1 TO G5-1
2860                Vec_p(2*J-1)=Vec_ptrans(J)
2861                Vec_p(2*J)=(Vec_ptrans(J+1)-Vec_ptrans(J))/2+Vec_p(2*J-1)
2862            NEXT J
2863            Vec_p(2*G5-1)=Vec_ptrans(G5)
2864            MAT Vec_ttrans= Vt1
2865            Grosseur=G9
2866            FOR J=1 TO G11
2867                Incremente=0
2868                FOR I=1 TO G9
2869                    IF Vt1(I)=Vt2(J) THEN
```

```
2870                Incremente=1
2871                END IF
2872            NEXT I
2873            IF Incremente=0 THEN
2874                ALLOCATE V_nov(Grosseur+1)
2875                Val_nov=Vec_t2(J)
2876                CALL Vec_nov(Vec_ttrans(*),V_nov(*),Val_nov)
2877                DEALLOCATE Vec_ttrans(*)
2878                Grosseur=Grosseur+1
2879                ALLOCATE Vec_ttrans(Grosseur)
2880                MAT Vec_ttrans= V_nov
2881                DEALLOCATE V_nov(*)
2882            END IF
2883        NEXT J
2884        G9=SIZE(Vec_ttrans,1)
2885        G10=G9+1*(G9-1)
2886        ALLOCATE Vec_t(G10),Vec_tper(G10)
2887        FOR J=1 TO G9-1
2888            Vec_t(2*J-1)=Vec_ttrans(J)
2889            Vec_t(2*J)=(Vec_ttrans(J+1)-Vec_ttrans(J))/2+Vec_t(2*J-1)
2890        NEXT J
2891        Vec_t(2*G9-1)=Vec_ttrans(G9)
2892        ALLOCATE Mat_v1c(G6,G10),Mat_v2c(G6,G10),Mat_v1cper(G6,G10)
2893        ALLOCATE Mat_v2cper(G6,G10)
2894        CALL Det_v12(Vec_p(*),Vec_t(*),Vec_p(*),Vec_t(*),Mat_v1c(*),Mat_v2c(*))
2895        !
2896        !*********************************************************************
2897        !
2898        !   AJOUT DES PERTURBATIONS SUR T, V1, V2
2899        !
2900        !*********************************************************************
2901        !
2902        IF Chx_err=1 THEN
2903            CALL Errnor(Vec_t(*),Vec_tper(*),Err_t)
2904            CALL Errnor(Mat_v1c(*),Mat_v1cper(*),Err_v1)
2905            CALL Errnor(Mat_v2c(*),Mat_v2cper(*),Err_v2)
2906        ELSE
2907            CALL Errunif(Vec_t(*),Vec_tper(*),Err_t)
2908            CALL Errunif(Mat_v1c(*),Mat_v1cper(*),Err_v1)
2909            CALL Errunif(Mat_v2c(*),Mat_v2cper(*),Err_v2)
2910        END IF
2911        Vec_tper(1)=Vec_t(1)+Err_t*RND/2
2912        Transit=SIZE(Vec_tper,1)
2913        Vec_tper(Transit)=Vec_t(Transit)-RND/2*Err_t
2914        Transit1=SIZE(Mat_v1c,1)
2915        Transit2=SIZE(Mat_v1c,2)
2916        FOR Boucle=1 TO Transit2
2917            Mat_v1cper(1,Boucle)=Mat_v1c(1,Boucle)+Err_v1*RND/2
2918            Mat_v1cper(Transit1,Boucle)=Mat_v1c(Transit1,Boucle)-Err_v1*RND/2
2919        NEXT Boucle
2920        Transit1=SIZE(Mat_v2c,1)
2921        Transit2=SIZE(Mat_v2c,2)
2922        FOR Boucle=1 TO Transit2
2923            Mat_v2cper(1,Boucle)=Mat_v2c(1,Boucle)+Err_v2*RND/2
2924            Mat_v2cper(Transit1,Boucle)=Mat_v2c(Transit1,Boucle)-Err_v2*RND/2
2925        NEXT Boucle
2926        !
2927        !
2928        !*********************************************************************
2929        !
2930        !   INTERPOLATION DE  T
2931        !
2932        !*********************************************************************
2933        !
2934        !
2935        ALLOCATE Mat_tp1(G1,G10),Mat_tp2(G2,G10),Xt1(G3),Yv1(G3)
```

```
2936            ALLOCATE Xt2(G4),Yv2(G4),Ta1(G3),Tb1(G3),Tc1(G3),Td1(G3),Ta2(G4),Tb2(G4)
2937            ALLOCATE Tc2(G4),Td2(G4)
2938            MAT Xt1= Vec_t1
2939            FOR J=1 TO G10
2940                Val_t=Vec_tper(J)
2941                FOR I=1 TO G1
2942                    MAT Yv1= Mat_v1per(I,*)
2943                    MAT Ta1= Ma1(I,*)
2944                    MAT Tb1= Mb1(I,*)
2945                    MAT Tc1= Mc1(I,*)
2946                    MAT Td1= Md1(I,*)
2947                    SELECT Chx_int
2948                    CASE 1
2949                     CALL Valspl3_2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Val_t,Ym)
2950                    CASE 2
2951                     CALL Valspl3_2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Val_t,Ym)
2952                    CASE 3
2953                     CALL Valspl3_2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Val_t,Ym)
2954                    CASE 4
2955                     CALL Valspl2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Val_t,Ym)
2956                    CASE 5
2957                     CALL Valspl2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Val_t,Ym)
2958                    CASE 6
2959                     CALL Valspl3_2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Val_t,Ym)
2960                    CASE 7
2961                     CALL Valspl2(Xt1(*),Yv1(*),Ta1(*),Tb1(*),Tc1(*),Val_t,Ym)
2962                    END SELECT
2963                    Mat_tp1(I,J)=Ym
2964                NEXT I
2965            NEXT J
2966            DEALLOCATE Xt1(*),Yv1(*)
2967            DEALLOCATE Xt2(*),Yv2(*),Ta1(*),Tb1(*),Tc1(*),Td1(*),Ta2(*),Tb2(*)
2968            DEALLOCATE Tc2(*),Td2(*)
2969            DEALLOCATE Vec_t1per(*),Vec_t2per(*)
2970            DEALLOCATE Mat_v1c(*),Mat_v2c(*)
2971            DEALLOCATE Mat_v1per(*),Mat_v2per(*)
2972            ALLOCATE A1(G1,G10),B1(G1,G10),C1(G1,G10),D1(G1,G10),A2(G2,G10)
2973            ALLOCATE B2(G2,G10),C2(G2,G10),D2(G2,G10)
2974            ALLOCATE Xp1(G1),Yp1(G1),Xp2(G2),Yp2(G2),Va1(G1),Vb1(G1),Vc1(G1),Vd1(G1)
2975            ALLOCATE Va2(G2),Vb2(G2),Vc2(G2),Vd2(G2)
2976            MAT Yp1= Vec_p1
2977 !  S1=INT(G1+SQRT(2*G1))
2978            S1=G1
2979            Sigy1=Err_p/2
2980            FOR J=1 TO G10
2981                MAT Xp1= Mat_tp1(*,J)
2982                SELECT Chx_int
2983                CASE 1
2984                    CALL Intspl3_2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*),Vd1(*))
2985                    MAT A1(*,J)= Va1
2986                    MAT B1(*,J)= Vb1
2987                    MAT C1(*,J)= Vc1
2988                    MAT D1(*,J)= Vd1
2989                CASE 2
2990                    CALL Optspl3_2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*),Vd1(*))
2991                    MAT A1(*,J)= Va1
2992                    MAT B1(*,J)= Vb1
2993                    MAT C1(*,J)= Vc1
2994                    MAT D1(*,J)= Vd1
2995                CASE 3
2996                    CALL Optspl3b_2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*),Vd1(*))
2997                    MAT A1(*,J)= Va1
2998                    MAT B1(*,J)= Vb1
2999                    MAT C1(*,J)= Vc1
3000                    MAT D1(*,J)= Vd1
3001                CASE 4
```

```
3002                CALL Intspl2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*))
3003                MAT A1(*,J)= Va1
3004                MAT B1(*,J)= Vb1
3005                MAT C1(*,J)= Vc1
3006            CASE 5
3007                CALL Optspl2(Xp1(*),Yp1(*),Va1(*),Vb1(*),Vc1(*))
3008                MAT A1(*,J)= Va1
3009                MAT B1(*,J)= Vb1
3010                MAT C1(*,J)= Vc1
3011            CASE 6
3012                CALL Smospl3(Xp1(*),Yp1(*),S1,Sigy1,Va1(*),Vb1(*),Vc1(*),Vd1(*))
3013                MAT A1(*,J)= Va1
3014                MAT B1(*,J)= Vb1
3015                MAT C1(*,J)= Vc1
3016                MAT D1(*,J)= Vd1
3017            CASE 7
3024                IF Sigy1=0 THEN
3025                  Sigy1=.000001
3026                END IF
3027                CALL Smospl2a(Xp1(*),Yp1(*),Sigy1,S1,Va1(*),Vb1(*),Vc1(*))
3028                MAT A1(*,J)= Va1
3029                MAT B1(*,J)= Vb1
3030                MAT C1(*,J)= Vc1
3031            END SELECT
3032        NEXT J
3033            !
3034            !
3035            !*************************************************************
3036            !
3037            !          INTERPOLATION DE P
3038            !
3039            !*************************************************************
3040            !
3041            !
3042        ALLOCATE Xv1(G6),Xv2(G6),Mint_p(G6,G10)
3043        Vmin=2.5-2.5*COS(PI/4)
3044        Vmax=2.5+2.5*SIN(PI/4)
3045        MAT Yp1= Vec_p1
3046        MAT Yp2= Vec_p2
3047        FOR J=1 TO G10
3048            MAT Xv1= Mat_v1cper(*,J)
3049            MAT Xv2= Mat_v2cper(*,J)
3050            FOR I=1 TO G6
3051            !
3052            !*********************************************************
3053            !
3054            !       CHOIX DU CAPTEUR
3055            !
3056            !*********************************************************
3057            !
3058                Nb1=INT(Xv1(I)/5)
3059                Nb2=INT(Xv2(I)/5)
3060                Val_ajus1=Xv1(I)-Nb1*5
3061                Val_ajus2=Xv2(I)-Nb2*5
3062                Val_ajus1=ABS(Val_ajus1-2.5)
3063                Val_ajus2=ABS(Val_ajus2-2.5)
3064                IF Val_ajus1<Val_ajus2 THEN
3065                    Chx_v=1
3066                ELSE
3067                    Chx_v=2
3068                END IF
3069                IF Chx_v=1 THEN
3070                    ALLOCATE V_int(G1),Yp(G1),A(G1),B(G1),C(G1),D(G1)
3071                    MAT V_int= Mat_tp1(*,J)
3072                    Gro_vec=G1
3073                    MAT Yp= Yp1
```

```
                MAT A= A1(*,J)
                MAT B= B1(*,J)
                MAT C= C1(*,J)
                MAT D= D1(*,J)
                Xv=Xv1(I)
                Xv=Xv+INT(Xv/5)*Vmax-INT(Xv/5)*3*Vmin
            ELSE
                ALLOCATE V_int(G2),Yp(G2),A(G2),B(G2),C(G2),D(G2)
                MAT V_int= Mat_tp1(*,J)
                Gro_vec=G1
                MAT Yp= Yp2
                MAT A= A1(*,J)
                MAT B= B1(*,J)
                MAT C= C1(*,J)
                MAT D= D1(*,J)
                Xv=Xv2(I)
                Xv=Xv+INT(Xv/5)*Vmax-INT(Xv/5)*3*Vmin+Vmax-Vmin
            END IF
            IF Xv>=V_int(1) AND Xv<=V_int(Gro_vec) THEN
                !
                !
                !*********************************************
                !       CALCUL DE P
                !*********************************************
                !
                !
                SELECT Chx_int
                CASE 1
                    CALL Valspl3_2(V_int(*),Yp(*),A(*),B(*),C(*),D(*),Xv,Pcal)
                CASE 2
                    CALL Valspl3_2(V_int(*),Yp(*),A(*),B(*),C(*),D(*),Xv,Pcal)
                CASE 3
                    CALL Valspl3_2(V_int(*),Yp(*),A(*),B(*),C(*),D(*),Xv,Pcal)
                CASE 4
                    CALL Valspl2(V_int(*),Yp(*),A(*),B(*),C(*),Xv,Pcal)
                CASE 5
                    CALL Valspl2(V_int(*),Yp(*),A(*),B(*),C(*),Xv,Pcal)
                CASE 6
                    CALL Valspl3_2(V_int(*),Yp(*),A(*),B(*),C(*),D(*),Xv,Pcal)
                CASE 7
                    CALL Valspl2(V_int(*),Yp(*),A(*),B(*),C(*),Xv,Pcal)
                END SELECT
            ELSE
                Pcal=9.9E+99
            END IF
            IF Xv<V_int(1) OR Xv>V_int(Gro_vec) THEN
                Mint_p(I,J)=9.9E+99
            ELSE
                Mint_p(I,J)=Pcal
            END IF
            DEALLOCATE V_int(*),A(*),B(*),C(*),D(*),Yp(*)
        NEXT I
    NEXT J
        !
        !*****************************************************
        !
        !               CALCUL DES ERREURS
        !*****************************************************
        !
    N=0
    Errmax=0
    Errmoy=0
    Errmax_trans=0
    Pos_p=1
    Gr_trans=SIZE(Vec_ptrans,1)
    Ecart_reel=0
```

```
3140            Ecart_reeltrans=0
3141            FOR J=1 TO G6
3142                FOR I=1 TO G10
3143                    M=1
3144                    Transit2=(Mint_p(J,I)-Vec_p(J))^2
3145                    Transit=ABS((Mint_p(J,I)-Vec_p(J)))/Vec_p(J)
3146                    IF Mint_p(J,I)>=1.0E+99 THEN
3147                        Transit=0
3148                        Transit2=0
3149                        M=0
3150                        Mint_p(J,I)=Vec_p(J)
3151                    END IF
3152                    Errmoy=Errmoy+Transit2
3153                    N=N+M
3154                    Condition=0
3155                    IF ABS(Mint_p(J,I)-Vec_p(J))/Vec_p(J)>Ecart_reeltrans THEN
3156                        FOR Z=1 TO Gr_trans
3157                            IF Vec_ptrans(Z)=Vec_p(J) THEN
3158                              Condition=1
3159                            END IF
3160                        NEXT Z
3161                        IF Condition=0 THEN
3162                            Ecart_reeltrans=ABS(Mint_p(J,I)-Vec_p(J))/Vec_p(J)
3163                            Pos_p=J
3164                            Pos_t=I
3165                        END IF
3166                    END IF
3167                    IF ABS((Mint_p(J,I)-Vec_p(J)))/Vec_p(J)>Ecart_reel THEN
3168                        Ecart_reel=ABS((Mint_p(J,I)-Vec_p(J)))/Vec_p(J)
3169                    END IF
3170                    IF ABS(Mint_p(J,I)-Vec_p(J))>Errmax THEN
3171                        Errmax=ABS(Mint_p(J,I)-Vec_p(J))
3172                    END IF
3173                NEXT I
3174            NEXT J
3175            Gr_trans=SIZE(Vec_ttrans,1)
3176            Pos_t=1
3177            Err_temp=0
3178            FOR J=1 TO G6
3179                FOR I=1 TO G10
3180                    Condition=0
3181                    IF ABS(Mint_p(J,I)-Vec_p(J))/Vec_p(J)>Err_temp THEN
3182                        FOR Z=1 TO Gr_trans
3183                            IF Vec_ttrans(Z)=Vec_t(I) THEN
3184                              Condition=1
3185                            END IF
3186                        NEXT Z
3187                        IF Condition=0 THEN
3188                            Err_temp=ABS(Mint_p(J,I)-Vec_p(J))/Vec_p(J)
3189                            Pos_t=I
3190                        END IF
3191                    END IF
3192                NEXT I
3193            NEXT J
3194            Errmoy=SQRT(Errmoy/N)
3195            Retr(1)=Vec_p(Pos_p)
3196            Retr(2)=Vec_t(Pos_t)
3197            Retr(3)=Vec_p(Pos_p)
3198            Retr(4)=Vec_t(Pos_t)
3199            Retr(5)=Errmoy
3200            Retr(6)=Errmax
3201            Retr(7)=Ecart_reel
3202            Retr(8)=1
3203 SUBEND
3204 CSUB Smospl3(REAL X(*),Y(*),S,Sigy,A(*),B(*),C(*),D(*))
3205 CSUB Det_v12(Va1(*),Va2(*),Vb1(*),Vb2(*),Ma(*),Mb(*))
```

```
3206 CSUB Det_v1(Pres1(*),Temp1(*),Ma(*))
3207 CDEF FNSign(REAL I)
3208 CDEF FNDer1(REAL X0,X1,X2,X3,Y0,Y1,Y2,Y3)
```

Les sous-programmes pour PCR1 et PCR2 sont:

```
170  END
180  SUB Det_v1(Pres1(*),Temp1(*),Ma(*))
190  !
200  !*************************************************************************
201  !
202  !       SOUS PROGRAMME DE MONOTONISATION DE DEUX SINUS
203  !
204  !                 REALISE PAR
205  !
206  !                SERGE AMPLEMAN
207  !
208  !
209  !       LE PROGRAMME TRANSFORME DEUX MATRICE DE TENSION
210  !          SINUSOIDALES EN SIGAUX MONOTONIQUES
211  !
212  !
213  !       ENTREE:   PRES1,TEMP1
214  !
215  !       SORTIE:   MA
216  !
217  !
218  !       PRES1 ET TEMP1 SONT DES VECTEURS
219  !       MA    EST UNE MATRICE DE TENSION
220  !
221  !
223  !*************************************************************************
224  !
225  !*************************************************************************
227  !         CARACTERISTIQUE DES ONDES SINUSOIDALES
228  !*************************************************************************
230  !
240  OPTION BASE 1
250  Nbre_temp1=SIZE(Temp1,1)
260  Nbre_pres1=SIZE(Pres1,1)
270  Dphas=PI/2
280  B=PI/50
290  Po=25
300  A=2*PI/Po
310  Dp=((PI/4)/A-(-PI/4)/A)
320  Vmax=SIN(PI/2-Dphas/2)
330  Vmin=SIN(3*PI/2-Dphas/2)
340  !
350  !*************************************************************************
360  ! RECHERCHE DE P1
370  !*************************************************************************
380  !
390  ALLOCATE P1(Nbre_temp1)
400  FOR J=1 TO Nbre_temp1
410    IF SIN(B*Temp1(J))>Vmax OR SIN(B*Temp1(J))<Vmin THEN
420      Deriv=COS(B*Temp1(J)-Dphas)
430      IF Deriv>=0 THEN
440        X=PI/4
450      ELSE
460        X=5*PI/4
470      END IF
480      C=-52
490      WHILE P1(J)<=0 OR P1(J)>Dp
```

```
500         P1(J)=(X+C*PI/2-B*Temp1(J)+Dphas)/A
510         C=C+2+(-1)^(C+1)
520       END WHILE
530     ELSE
540       Deriv=COS(B*Temp1(J))
550       IF Deriv>=0 THEN
560         X=PI/4
570       ELSE
580         X=5*PI/4
590       END IF
600       C=-52
610       WHILE P1(J)<=0 OR P1(J)>Dp
620         P1(J)=(X+C*PI/2-B*Temp1(J))/A
630         C=C+2+(-1)^(C+1)
640       END WHILE
650     END IF
660   NEXT J
670   !
680   !*************************************************************************
690   ! DETERMINATION DE LA FRACTION MONOTONIQUE N
700   !*************************************************************************
710   !
720   FOR J=1 TO Nbre_temp1
730     FOR I=1 TO Nbre_pres1
740       Z=(Pres1(I)-P1(J))/Dp
750       IF Z=INT(Z) THEN
760           N=Z
770       ELSE
780         IF INT(Z)<0 THEN
790           N=0
800         ELSE
810           N=INT(Z)+1
820         END IF
830       END IF
831       IF Z=-1 THEN N=0
840   !
850   !*************************************************************************
860   ! MONOTONISATION DES PARTIES COMPRISENT ENTRE VMIN ET VMAX
870   !*************************************************************************
880   !
890       IF SIN(A*Pres1(I)+B*Temp1(J))>Vmax OR SIN(A*Pres1(I)+B*Temp1(J))<Vmin THEN
900         Derive=COS(A*Pres1(I)+B*Temp1(J)-Dphas)
910         V=SIN(A*Pres1(I)+B*Temp1(J)-Dphas)
920       ELSE
930         Derive=COS(A*Pres1(I)+B*Temp1(J))
940         V=SIN(A*Pres1(I)+B*Temp1(J))
950       END IF
960       IF Derive>=0 THEN
970         Pente=0
980       ELSE
990         Pente=1
1000      END IF
1010      Vamax=2.5+2.5*Vmax
1020      Vamin=2.5+2.5*Vmin
1030      IF Pente=1 THEN
1040        Valeur=2.5+2.5*V
1050        Ma(I,J)=((N+1)*Vamax)-((N-1)*Vamin)-Valeur
1060      ELSE
1070        Valeur=2.5+2.5*V
1080        Ma(I,J)=(N*Vamax)-(N*Vamin)+Valeur
1090      END IF
1100    NEXT I
1110  NEXT J
1120  !
1130  !*************************************************************************
1140  ! MONOTONISATION DE LA TEMPERATURE POUR POUVOIR APPLIQUER LE SPLINE
```

```
1150  !*************************************************************************
1160  !
1170  FOR J=1 TO Nbre_temp1
1180    FOR I=1 TO Nbre_pres1
1190      Z=Temp1(J)/25+.5
1200      N=INT(Z)
1210      Ma(I,J)=Ma(I,J)+N*(5*SIN(PI/4))
1220    NEXT I
1230  NEXT J
1430  SUBEND


178  END
179  SUB Det_v12(Va1(*),Va2(*),Vb1(*),Vb2(*),Ma(*),Mb(*))
180   OPTION BASE 1
181  !
182  !*************************************************************
183  !
184  !  SOUS-PROGRAMME QUI TROUVE DEUX MATRICES DE TENSION
185  !
186  !  CES TENSION REPRESENTENT LES PRESSIONS DES DEUX CAPTEURS
187  !
188  !        ENTREE:  - VA1,VA2  OU VA1= PRESSION DU CAPTEUR1
189  !                          VA2= TEMPERATURE DU CAPTEUR 1
190  !                 - VB1,VB2  OU VB1= PRESSION DU CAPTEUR 2
191  !                          VB2= TEMPERATURE DU CAPTEUR 2
192  !
193  !*************************************************************
194  !
195   Nbre_temp1=SIZE(Va2,1)
196   Nbre_pres1=SIZE(Va1,1)
197   Nbre_temp2=SIZE(Vb2,1)
198   Nbre_pres2=SIZE(Vb1,1)
199  !
200  !  LE SINUS EST DE LA FORME SIN(A*P + B*T)
201  !
202  !   OU A=2*PI/Po
203  !
204   B=PI/50
205   Po=25
206   A=2*PI/Po
207   Demi_p=Po/2
208   N=0
209  !
210  !
211  !  CLACUL DE LA PREIMIERE MATRICE
212  !
213  !
214   FOR I=1 TO Nbre_temp1
215    Tmin=0
216    Tmax=100/4
217    N=1
218    Sortir=1
219    WHILE Sortir=1
220     IF Tmin<=Va2(I) AND Va2(I)<Tmax THEN
221       Sortir=0
222       Pos1=N
223     END IF
224     Tmin=Tmax
225     Tmax=Tmax+100/2
226     N=N+1
227    END WHILE
228    Ecart=5*(Pos1-1)
229    Derive=PROUND(COS(PI/50*Va2(I)),-15)
230    IF Derive=0 THEN
```

```
231       IF PROUND(SIN(B*Va2(I)),-15)>0 THEN
232         Penteneg=1
233       ELSE
234         Penteneg=0
235       END IF
236     ELSE
237       IF Derive>0 THEN
238         Penteneg=0
239       ELSE
240         Penteneg=1
241       END IF
242     END IF
243     N=0
244     Sortir=1
245     WHILE Sortir=1
246       P1=(2*N+1)*PI/(2*A)-B*Va2(I)/A
247       IF P1>=0 THEN
248         Sortir=0
249       END IF
250       N=N+1
251     END WHILE
252     FOR J=1 TO Nbre_pres1
253      Vtrans=2.5+2.5*PROUND(SIN(A*Va1(J)+B*Va2(I)),-15)
254      N=1
255      Pmin=0
256      Pmax=P1
257      Sortir=1
258      WHILE Sortir=1
259        IF Pmin<=Va1(J) AND Va1(J)<Pmax THEN
260          Sortir=0
261          Pos1=N
262        END IF
263        Pmin=Pmax
264        Pmax=Po/2+Pmax
265        N=N+1
266      END WHILE
267      IF Penteneg=0 THEN
268        Ma(J,I)=Ecart+INT(Pos1/2)*10+(-1)^(Pos1-1)*Vtrans
269      ELSE
270        Ma(J,I)=Ecart+5+INT((Pos1-1)/2)*10+(-1)^Pos1*Vtrans
271      END IF
272    NEXT J
273   NEXT I
306   !
307   ! CALCUL DE LA DEUXIEME MATRICE
308   !
309   !
310   FOR I=1 TO Nbre_temp2
311     Tmin=0
312     Tmax=100/2
313     N=1
314     Sortir=1
315     WHILE Sortir=1
316       IF Tmin<=Vb2(I) AND Vb2(I)<Tmax THEN
317         Sortir=0
318         Pos1=N
319       END IF
320       Tmin=Tmax
321       Tmax=Tmax+100/2
322       N=N+1
323     END WHILE
324     Ecart=5*(Pos1-1)
325     Derive=PROUND(SIN(PI/50*Vb2(I)),-15)
326     IF Derive=0 THEN
327       IF PROUND(-COS(B*Vb2(I)),-15)>0 THEN
328         Penteneg=1
```

```
329       ELSE
330         Penteneg=0
331       END IF
332     ELSE
333       IF Derive>0 THEN
334         Penteneg=0
335       ELSE
336         Penteneg=1
337       END IF
338     END IF
339     N=1
340     Sortir=1
341     WHILE Sortir=1
342       P1=N*PI/A-B*Vb2(I)/A
343       IF P1>=0 THEN
344         Sortir=0
345       END IF
346       N=N+1
347     END WHILE
354     CLEAR SCREEN
356     FOR J=1 TO Nbre_pres2
357       Vtrans=2.5-2.5*PROUND(COS(A*Vb1(J)+B*Vb2(I)),-15)
358       N=1
359       Pmin=0
360       Pmax=P1
361       Sortir=1
362       WHILE Sortir=1
363         IF Pmin<=Vb1(J) AND Vb1(J)<Pmax THEN
364           Sortir=0
365           Pos1=N
366         END IF
367         Pmin=Pmax
368         Pmax=Po/2+Pmax
369         N=N+1
370       END WHILE
371       IF Penteneg=0 THEN
372         Mb(J,I)=Ecart+INT(Pos1/2)*10+(-1)^(Pos1-1)*Vtrans
373       ELSE
374         Mb(J,I)=Ecart+5+INT((Pos1-1)/2)*10+(-1)^Pos1*Vtrans
375       END IF
376     NEXT J
377   NEXT I
378 SUBEND


220 END
230 SUB Det_vg(Va1(*),Va2(*),Ma(*))
240   OPTION BASE 1
250 !
260 !*****************************************************************
270 !
280 !  SOUS-PROGRAMME QUI TROUVE UNE MATRICE DE TENSION
290 !
300 !  CETTE TENSION REPRESENTE LA PRESSION DU DUIDE
310 !
320 !         ENTREE:   - VA1,VA2  OU VA1= PRESSION DU GUIDE
330 !                           VA2= TEMPERATURE DU GUIDE
360 !
370 !*****************************************************************
380 !
390   Nbre_temp1=SIZE(Va2,1)
400   Nbre_pres1=SIZE(Va1,1)
430 !
440 !  LE SINUS EST DE LA FORME SIN(A*P + B*T)
450 !
```

```
460  !   OU A=2*PI/Po
470  !
480   B=PI/200
490   Po=300
500   A=2*PI/Po
510   Demi_p=Po/2
520   N=0
530  !
540  !
550  !  CLACUL DE LA PREIMIERE MATRICE
560  !
570  !
580   FOR I=1 TO Nbre_temp1
590    FOR J=1 TO Nbre_pres1
600     Ma(J,I)=2.5+2.5*SIN(A*Va1(J)+B*Va2(I)-PI/2+PI/24)
1070   NEXT J
1080  NEXT I
1630 SUBEND



190  END
200  SUB Smospl3(REAL X(*),Y(*),S,Sigy,A(*),B(*),C(*),D(*),INTEGER N,N0)
210  !_____________________________________________________________________
220  !
230  !   APPROXIMATION DU LISSAGE DES DONNEES X(*),Y(*) UTILISANT LES
240  !   SPLINES CUBIQUES.
250  !
260  !   ENTREE: X,Y   COORDONNEE DES POINTS DE MESURE.
270  !         Sigy  DEVIATION STANDARD DE L'ERREUR SUR Y
280  !         S     PARAMETRE DE REGULARISATION
290  !         N     NOMBRE DE POINTS DE DONNEES
300  !         N0    INDEX DU PREMIER POINT DE DONNEE
310  !
320  !   SORTIE: A,B,C,D  COEFFICIENT DE LA FONCTION SPLINE
330  !_____________________________________________________________________
340     INTEGER N2,Li,M1,M2,I
350     REAL P,H,F,F2,G,E,Sigy2
360     ALLOCATE REAL R(-1:N),R1(-1:N),R2(-1:N),T0(-1:N),T1(-1:N)
370     ALLOCATE REAL U(-1:N),V(-1:N)
380     N2=N0+N-1
390     M1=N0+1
400     M2=N2-1
410     P=0        ! PARAMETRE P DE LAGRANGE
420     H=X(M1)-X(N0)
430     F=(Y(M1)-Y(N0))/H
440     FOR I=M1 TO M2
450       G=H
460       H=X(I+1)-X(I)
470       E=F
480       F=(Y(I+1)-Y(I))/H
490       D(I)=F-E
500  ! LES VECTEURS T0 ET T1 CONTIENT LES DIAGONALES DE LA MATRICE T
510       T0(I)=2*(H+G)/3
520       T1(I)=H/3
530  ! LES VECTEURS R,R1 ET R2 CONTIENNENT LA DIAGONALE DE LA MATRICE Sigy*Q
540       R2(I)=Sigy/G
550       R(I)=Sigy/H
560       R1(I)=-R2(I)-R(I)
570     NEXT I
580  ! CALCUL LE PRODUIT DE Sigy*Q ET LA TRANSPOSE
590     FOR I=M1 TO M2
600       C(I)=R(I)^2+R1(I)^2+R2(I)^2
610       B(I)=R(I)*R1(I+1)+R1(I)*R2(I+1)
620       A(I)=R(I)*R2(I+2)
630     NEXT I
```

```
640     F2=-S
650 ! DEPART DE L'ITERATION
660     Li=0
670 ! CALCUL DU VECTEUR U AVEC L'UTILISATION DE LA DECOMPOSITION DE CHOLESKY
680     REPEAT
690       FOR I=M1 TO M2
700         R1(I-1)=F*R(I-1)
710         R2(I-2)=G*R(I-2)
720         R(I)=1/(P*C(I)+T0(I)-F*R1(I-1)-G*R2(I-2))
730         U(I)=D(I)-R1(I-1)*U(I-1)-R2(I-2)*U(I-2)
740         F=P*B(I)+T1(I)-H*R1(I+1)
750         G=H
760         H=A(I)*P
770       NEXT I
780       FOR I=M2 TO M1 STEP -1
790         U(I)=R(I)*U(I)-R1(I)*U(I+1)-R2(I)*U(I+2)
800       NEXT I
810       H=0
820       E=0
830       Sigy2=Sigy^2
840 !  CALCUL LE VECTEUR V=Sigy^Q*U ET LE CARRE DE LA SECONDE NORME E
850       FOR I=N0 TO M2
860         G=H
870         H=(U(I+1)-U(I))/(X(I+1)-X(I))
880         V(I)=(H-G)*Sigy2
890         E=E+V(I)*(H-G)
900       NEXT I
910       G=-H*Sigy2
920       V(N2)=G
930       E=E-G*H
940       G=F2
950       F2=E*P*P
960 !   SI FIN DES ITERATIONS, ALORS CALCUL LES COEFFICIENTS
970       IF (F2>=S) OR (F2<G) THEN GOTO 1120
980       F=0
990       H=(V(M1)-V(N0))/(X(M1)-X(N0))
1000      FOR I=M1 TO M2
1010        G=H
1020        H=(V(I+1)-V(I))/(X(I+1)-X(I))
1030        G=H-G-R1(I-1)*R(I-1)-R2(I-2)*R(I-2)
1040        F=F+G*R(I)*G
1050        R(I)=G
1060      NEXT I
1070      H=E-P*F
1080      IF H<=0 THEN GOTO 1120
1090      P=P+(S-F2)/((SQRT(S/E)+P)*H)
1100      Li=Li+1
1110    UNTIL Li>100
1120 ! CALCUL LES COEFFICIENTS
1130    FOR I=N0 TO N2
1140      D(I)=Y(I)-P*V(I)
1150      B(I)=U(I)
1160    NEXT I
1170    FOR I=N0 TO M2
1180      H=X(I+1)-X(I)
1190      A(I)=(B(I+1)-B(I))/(3*H)
1200      C(I)=(D(I+1)-D(I))/H-(H*A(I)+B(I))*H
1210    NEXT I
1220    DEALLOCATE R(*),R1(*),R2(*),T0(*),T1(*),U(*),V(*)
1230  SUBEND
1240  !
1250  !


2620  SUB Smospl2c(REAL X(*),Y(*),Sigy,S,A(*),B(*),C(*))
2630  !_____________________________________________________________
```

```
2640 !
2650 !  CALCUL LES COEFFICIENTS A,B,C DE LA FONCTION SPLINE PARABOLIQUE
2660 !  DE LISSAGE A L'AIDE DES POINTS DE DONNEES X(N) ET Y(N), N=1,...,Nn
2670 !
2680 !
2690 !  C(1)+C(2)+...+C(Nn)=Y(1)+Y(2)+...+Y(Nn)
2700 !  SQR(C(1)-Y(1))+SQR(C(2)-Y(2))+...+SQR(C(Nn)-Y(Nn))<S*SQR(Sigy)
2710 !
2720 !  ENTREE: X,Y  LES POINTS DE DONNEES
2730 !          Sigy DEVIATION STANDARD DES DONNEES DE Y
2740 !          S   PARAMETRE DE REGULARISATION  (S=Nn OU S=Nn-1)
2750 !          Nn   LE NOMBRE DE POINTS DE DONNEES
2760 !
2770 !  SORTIE: A,B,C  LES COEFFICIENTS DE LA FONCTION SPLINE PARABOLIQUE
2780 !                 DE LISSAGE.
2790 !_____________________________________________________________________
2800   INTEGER N
2810   REAL Da,Dj,Dj_db1,J,La,La0,La1,La2,Sn,Sd,W0,W1,W2
2811   N=SIZE(X,1)
2812   Nn=N
2820   ALLOCATE REAL Dj_dc(-1:N),Dj_dq(-1:N)
2830! VALEURS INITIALES DES PARAMETRES
2840   B(1)=FNDer1(X(1),X(2),X(3),X(4),Y(1),Y(2),Y(3),Y(4))
2850   FOR N=1 TO Nn-1
2860     X(N)=X(N+1)-X(N)    ! DX[N] --> X[N]
2870     C(N)=Y(N)
2880   NEXT N
2890   C(Nn)=Y(Nn)
2891   FOR N=1 TO Nn-1
2892     A(N)=(C(N+1)-C(N))/X(N)-B(N)
2893     B(N+1)=B(N)+2*A(N)
2894     A(N)=A(N)/X(N)
2895   NEXT N
2910! VALEUR INITIAL DE LA NORME L2 DE LA SECONDE DERIVE DU SPLINE
2920   J=0
2930   FOR N=1 TO Nn-1
2940     J=J+(A(N)^2)*X(N)
2950   NEXT N
2960   J=J/2
2970! VALEURS INITIALES DES PARAMETRES D'ITERATION
2980   W0=-S*Sigy^2
2990   W1=0
3000   W2=0
3010   La=0
3020   REPEAT
3030     Dj_db1=0
3040     FOR N=1 TO Nn-1
3050       IF (N MOD 2)=1 THEN
3060         Dj_db1=Dj_db1-A(N)
3070       ELSE
3080         Dj_db1=Dj_db1+A(N)
3090       END IF
3100     NEXT N
3110! CALCUL dJ_dQ(N) POUR N=1,2,...,NN
3120     Dj_dq(Nn-1)=A(Nn-1)
3130     FOR N=Nn-2 TO 2 STEP -1
3140       Dj_dq(N)=A(N)-A(N+1)-Dj_dq(N+1)
3150     NEXT N
3160     Dj_dq(1)=A(1)
3170! CALCUL La0
3180     Da=Dj_dq(1)-Dj_db1
3190     Sn=Da*A(1)
3200     Sd=(Da^2)/X(1)
3210     FOR N=2 TO Nn-1
3220       Da=Dj_dq(N)-Dj_dq(N-1)-Da
3230       Sn=Sn+Da*A(N)
```

```
3240        Sd=Sd+(Da^2)/X(N)
3250      NEXT N
3260      La0=-Sn/Sd
3270! CALCUL dJ_dC(N) POUR N=1,2,...,NN
3280      Dj_dc(1)=0
3290      FOR N=1 TO Nn-1
3300        Dj_dc(1)=Dj_dc(1)-(1-N/Nn)*X(N)*Dj_dq(N)
3310      NEXT N
3320      FOR N=2 TO Nn
3330        Dj_dc(N)=Dj_dc(N-1)+Dj_dq(N-1)*X(N-1)
3340      NEXT N
3350! RESOLUTION DE L'EQUATION QUADRATIQUE AVEC LA1 ET LA2 RESPECTEES
3360      W0=W0+La*(La*W2+2*W1)
3370      W1=0
3380      W2=0
3390      FOR N=1 TO Nn
3400        W1=W1+Dj_dc(N)*(C(N)-Y(N))
3410        W2=W2+Dj_dc(N)^2
3420      NEXT N
3430      La2=(-W1-FNSign(W1)*SQRT(W1^2-W0*W2))/W2
3440      IF La2=0 THEN
3450        La1=0
3460      ELSE
3470        La1=W0/W2/La2
3480      END IF
3490! CHOIX DE La DANS La0,La1 ET La2
3500      IF La1>La2 THEN
3510        La=La1
3520        La1=La2
3530        La2=La
3540      END IF
3550      IF La0<0 THEN
3560        La=.9999*MAX(La1,La0)
3570      ELSE
3580        La=.9999*MIN(La0,La2)
3590      END IF
3600! CALCUL LES PARAMETRES DU SPLINE
3610      FOR N=1 TO Nn
3620        C(N)=C(N)+La*Dj_dc(N)
3630      NEXT N
3640      B(1)=B(1)+La*Dj_db1
3641      FOR N=1 TO Nn-1
3642        A(N)=(C(N+1)-C(N))/X(N)-B(N)
3643        B(N+1)=B(N)+2*A(N)
3644        A(N)=A(N)/X(N)
3645      NEXT N
3660    UNTIL ABS(La*(Sn+La*Sd/2))<.001*J   ! LE NOUVEAU J EST PLUS PETIT
3670                                        ! QUE 0.1% DE SA VALEUR INITIAL
3680  DEALLOCATE Dj_dc(*),Dj_dq(*)
3690  SUBEND


3700  !
3710  !
3720  SUB Compab(REAL A(*),B(*),C(*),X(*),INTEGER Nn)
3730  !_____________________________________________________________________
3740  !
3750  !  ROUTINE UTILE POUR SMOSPL2C
3760  !
3770  !  ENTREE: A,B,C   LES COEFFICIENTS DE LA FONCTION SPLINE PARABOLIQUE
3780  !          X       LES DONNEES EN X
3790  !          Nn      LE NOMBRE DE DONNEES
3800  !
3810  !  SORTIE: A,B,C   LES COEFFICIENTS DE LA FONCTION SPLINE PARABOLIQUE
3820  !_____________________________________________________________________
3830    INTEGER N
```

```
3840   FOR N=1 TO Nn-1
3850     A(N)=(C(N-1)-C(N))/X(N)-B(N)
3860     B(N+1)=B(N)+2*A(N)
3870     A(N)=A(N)/X(N)
3880   NEXT N
3890 SUBEND


3900 !
3910 !
3920 SUB Valspl3(INTEGER N,REAL X(*),A(*),B(*),C(*),D(*),Vx,Vy,Vy1)
3930 !_____________________________________________________________
3940 !
3950 ! CALCUL LA VALEUR Vy ET SA PREMIERE DERIVEE Vy1 DE LA FONCTION
3960 ! SPLINE CUBIQUE DEFINIE PAR X ET LES COEFFICIENTS A,B,C,D
3970 !
3980 ! ENTREE: X  LES VALEURS DE DE L'AXE DES X
3990 !        A,B,C,D   LES COEFFICIENTS DE LA SPLINE CUBIQUE
4000 !        N  LE NOMBRE DE POINTS DE DONNEES
4010 !        Vx LA VALEUR SITUE DANS L'INTERVALLE [Xi,Xi+1]
4020 !
4030 ! SORTIE: Vy LA VALEUR DE L'AXE DES Y SITUE DANS [Yi,Yi+1]
4040 !        Vy1 LA DERIVE DE Vy
4050 !_____________________________________________________________
4060   INTEGER I,Il,Ih
4070   REAL Dx,H
4080   Il=1
4090   Ih=N
4100   WHILE Ih>Il+1
4110     I=(Ih+Il) DIV 2
4120     IF X(I)>Vx THEN
4130       Ih=I
4140     ELSE
4150       Il=I
4160     END IF
4170   END WHILE
4180     IF Il=Ih THEN
4190       IF Il=N THEN
4200         Il=Il-1
4210       ELSE
4220         Ih=Ih+1
4230       END IF
4240     END IF
4250     H=X(Ih)-X(Il)
4260     Dx=Vx-X(Il)
4270     Vy=((A(Il)*Dx+B(Il))*Dx+C(Il))*Dx+D(Il)
4280     Vy1=(3*A(Il)*Dx+2*B(Il))*Dx+C(Il)
4290   SUBEND


4300   !
4310   !
4320 DEF FNSign(REAL I)
4330 !_____________________________________________________________
4340 !
4350 !  DETERMINE LE SIGNE DE LA VALEUR DE I
4360 !
4370 !  ENTREE: I  UNE VALEUR REELLE
4380 !
4390 !  SORTIE: Rep  LA REPONSE ENTRIERE (1 OU -1)
4400 !_____________________________________________________________
4410   INTEGER Rep
4420   IF I>0 THEN
4430     Rep=1
4440   ELSE
4450     Rep=-1
```

```
4460    END IF
4470    RETURN Rep
4480  FNEND


4490  !
4500  !
4720  SUB Valspl2(INTEGER N,REAL X(*),A(*),B(*),C(*),Vx,Vy,Vy1)
4730  !_____________________________________________________________________
4740  !
4750  !  CALCUL LA VALEUR Vy ET SA PREMIERE DERIVEE Vy1 DE LA FONCTION
4760  !  SPLINE PARABOLIQUE DEFINIE PAR X ET LES COEFFICIENTS A,B,C
4770  !
4780  !  ENTREE: X  LES VALEURS DE DE L'AXE DES X
4790  !          A,B,C   LES COEFFICIENTS DE LA SPLINE PARABOLIQUE
4800  !          N  LE NOMBRE DE POINTS DE DONNEES
4810  !          Vx LA VALEUR SITUE DANS L'INTERVALLE [Xi,Xi+1]
4820  !
4830  !  SORTIE: Vy LA VALEUR DE L'AXE DES Y SITUE DANS [Yi,Yi+1]
4840  !          Vy1 LA DERIVE DE Vy
4850  !_____________________________________________________________________
4860    INTEGER I,Il,Ih
4870    REAL Dx,H
4880    Il=1
4890    Ih=N
4900    WHILE Ih>Il+1
4910      I=(Ih+Il) DIV 2
4920      IF X(I)>Vx THEN
4930        Ih=I
4940      ELSE
4950        Il=I
4960      END IF
4970    END WHILE
4980    IF Il=Ih THEN
4990      IF Il=N THEN
5000        Il=Il-1
5010      ELSE
5020        Ih=Ih+1
5030      END IF
5040    END IF
5050    H=X(Ih)-X(Il)
5060    Dx=Vx-X(Il)
5070    Vy=(A(Il)*Dx+B(Il))*Dx+C(Il)
5080    Vy1=2*A(Il)*Dx+B(Il)
5090  SUBEND


5100  SUB Smospl2a(REAL X(*),Y(*),Sigy,S,A(*),B(*),C(*))
5110  !_____________________________________________________________________
5120  !
5130  !  CALCUL LES COEFFICIENTS A,B,C DE LA FONCTION SPLINE PARABOLIQUE
5140  !  DE LISSAGE A L'AIDE DES POINTS DE DONNEES X(N) ET Y(N), N=1,...,Nn
5150  !
5160  !
5170  !  C(1)+C(2)+...+C(Nn)=Y(1)+Y(2)+...+Y(Nn)
5180  !  SQR(C(1)-Y(1))+SQR(C(2)-Y(2))+...+SQR(C(Nn)-Y(Nn))<S*SQR(Sigy)
5190  !
5200  !  ENTREE: X,Y  LES POINTS DE DONNEES
5210  !          Sigy DEVIATION STANDARD DES DONNEES DE Y
5220  !          S   PARAMETRE DE REGULARISATION  (S=Nn OU S=Nn-1)
5230  !          Nn  LE NOMBRE DE POINTS DE DONNEES
5240  !
5250  !  SORTIE: A,B,C  LES COEFFICIENTS DE LA FONCTION SPLINE PARABOLIQUE
5260  !                 DE LISSAGE.
5270  !_____________________________________________________________________
5280    INTEGER N,K
```

```
5290   REAL Dx_dx,Det,J,La,La0,La1,La2,Sn,Sd,S1,S2,S11,S12,S21,S22,V,W0,W1,W2
5291   N=SIZE(X,1)
5292   Nn=N
5300   ALLOCATE REAL Dc(-1:N),Dx(-1:N),P(-1:N),Q(-1:N),R(-1:N)
5310! VALEURS INITIALES DES PARAMETRES
5320   B(1)=DROUND(FNDer1(X(1),X(2),X(3),X(4),Y(1),Y(2),Y(3),Y(4)),14)
5330   C(1)=DROUND(Y(1),14)
5340   J=0
5341   Soman=0
5350   FOR N=1 TO Nn-1
5360     C(N+1)=DROUND(Y(N+1),14)
5370     Dx(N)=DROUND(X(N+1)-X(N),14)
5380     A(N)=DROUND((C(N+1)-C(N))/Dx(N)-B(N),14)
5390     V=DROUND(A(N),14)
5400     B(N+1)=DROUND(B(N)+2*A(N),14)
5410     A(N)=DROUND(A(N)/Dx(N),14)
5411     Soman=DROUND(Soman+A(N),14)
5420     J=DROUND(J+V*A(N),14)
5430   NEXT N
5431   IF ABS(Soman)>0 THEN
5440     J=J/2 ! LA NORME L2 DE LA SECONDE DERIVE DE LA SPLINE
5450! CALCUL DE P,Q,R ET S
5460     S11=0
5470     S12=0
5480     S21=0
5490     S22=0
5500     R(1)=0
5510     R(2)=0
5520     FOR N=1 TO Nn
5530       SELECT N
5540       CASE =1
5550         P(1)=1
5560         Q(1)=0
5570       CASE =2
5580         P(2)=0
5590         Q(2)=1
5600       CASE ELSE
5610         Dx_dx=Dx(N-1)/Dx(N-2)
5620         P(N)=P(N-1)+Dx_dx*(P(N-1)-P(N-2))
5630         Q(N)=Q(N-1)+Dx_dx*(Q(N-1)-Q(N-2))
5640       END SELECT
5650       S11=S11+P(N)
5660       S12=S12+Q(N)
5670       S21=S21+X(N)*P(N)
5680       S22=S22+X(N)*Q(N)
5690     NEXT N
5700     Det=S11*S22-S12*S21
5710     V=S11
5720     S11=S22/Det
5730     S12=-S12/Det
5740     S21=-S21/Det
5750     S22=V/Det
5760     W0=-S*(Sigy)^2
5770     W1=0
5780     W2=0
5790     La=0
5800     LOOP
5810! CALCUL LA0
5820       Sn=0
5830       Sd=0
5840       FOR N=1 TO Nn-1
5850         V=A(N)^2
5860         Sn=Sn+V
5870         Sd=Sd+V/Dx(N)
5880       NEXT N
5890       La0=-Sn/Sd
```

```
5900! CALCUL DC(N) POUR N=1,2,...,NN
5910       S1=0
5920       S2=0
5930       FOR N=3 TO Nn
5940         R(N)=R(N-1)+Dx(N-1)*((R(N-1)-R(N-2))/Dx(N-2)+A(N-1)+A(N-2))
5950         S1=S1-R(N)
5960         S2=S2-X(N)*R(N)
5970       NEXT N
5980! RESOLUTION DE L'EQUATION QUADRATIQUE AVEC LA1 ET LA2 RESPECTEES
5990       W0=W0+La*(La*W2+2*W1)
6000       W1=0
6010       W2=0
6020       FOR N=1 TO Nn
6030         SELECT N
6040         CASE =1
6050           Dc(1)=S11*S1+S12*S2
6060         CASE =2
6070           Dc(2)=S21*S1+S22*S2
6080         CASE ELSE
6090           Dc(N)=P(N)*Dc(1)+Q(N)*Dc(2)+R(N)
6100         END SELECT
6110         W1=W1+Dc(N)*(C(N)-Y(N))
6120         W2=W2+Dc(N)^2
6130       NEXT N
6140       La2=(-W1-FNSign(W1)*SQRT(W1^2-W0*W2))/W2
6150       La1=W0/W2/La2
6160! CHOIX DE La DANS La0,La1 ET La2
6170       IF La1>La2 THEN
6180         La=La1
6190         La1=La2
6200         La2=La
6210       END IF
6220       IF La0<0 THEN
6230         La=.9999*MAX(La1,La0)
6240       ELSE
6250         La=.9999*MIN(La0,La2)
6260       END IF
6270! CALCUL LES PARAMETRES DU SPLINE
6280       FOR N=1 TO Nn
6290         C(N)=C(N)+La*Dc(N)
6300       NEXT N
6310       FOR N=1 TO Nn-1
6320         A(N)=A(N)+La*A(N)/Dx(N)
6330       NEXT N
6340       K1=ABS(La*(Sn+La*Sd/2))
6350       K2=.001*J
6360     EXIT IF K1<K2
6370     END LOOP
6380     ! LE NOUVEAU J EST PLUS PETIT QUE 0.1% DE SA VALEUR INITIAL
6390     FOR N=1 TO Nn-1
6400       B(N)=(C(N+1)-C(N))/Dx(N)-A(N)*Dx(N)
6410     NEXT N
6411   END IF
6420   DEALLOCATE Dc(*),Dx(*),P(*),Q(*),R(*)
6430  SUBEND


6440  !
6450  !
6460  DEF FNDer1(REAL X0,X1,X2,X3,Y0,Y1,Y2,Y3)
6470  !_____________________________________________________________________
6480  !
6490  !  LA VALEUR DE dx/dy POUR x=X0 EST CALCULEE VIA L'INTERPOLATION DE
6500  !  LA DIFFERENTIATION DU POLYNOME DE LAGRANGE (X0,Y0),...,(X3,Y3)
6510  !
6520  !  ENTREE: X0,X1,X2,X3,Y0,Y1,Y2,Y3  LES POINTS DE DONNEES
```

```
6530 !
6540 ! SORTIE: Der1  LE PARAMETRE DE LA REPONSE
6550 !_____________________________________________________________
6560  REAL Der1,C0,C1,C2,C3,Dx1,Dx2,Dx3
6570  Dx1=X1-X0
6580  Dx2=X2-X0
6590  Dx3=X3-X0
6600  C0=-1/Dx1-1/Dx2-1/Dx3
6610  C1=Dx2*Dx3/Dx1/(X2-X1)/(X3-X1)
6620  C2=Dx1*Dx3/Dx2/(X1-X2)/(X3-X2)
6630  C3=Dx1*Dx2/Dx3/(X1-X3)/(X2-X3)
6640  Der1=C0*Y0+C1*Y1+C2*Y2+C3*Y3
6641  Der1=PROUND(Der1,-10)
6650  RETURN Der1
6660 FNEND


110  END
120  SUB Intspl3(X(*),Y(*),S(*))
130  !
140  !
150  !*************************************************************************
160  !
170  ! SOUS-PROGRAMME Intspl3
180  !
190  !  DESCRIPTION:  SOUS-PROGRAMME QUI PERMET DE DETERMINER S(1) A
200  !              S(N).  LA METHODE UTILISE CONSISTE A DETREMINER
210  !              LA DERIVE PREMIERE DES POINTS [X(1),Y(1)] ET
220  !              [X(N),Y(N)], D'ASSOCIER CES VALEURS A S(1) ET S(N)
230  !              ET A PARTIR DE CES VALEURS, ETABLIR UNE FONCTION
240  !              DE RECURANCE POUR DETERMINER S(2)...S(N-1).
250  !
260  !     ENTRER:  X(*),Y(*) OU X(*) REPRESENTE LES POINT DE MESURE
270  !              Y(*) LA VALEUR DE LA MESURE
280  !
290  !     SORTIE: S(*)
300  !
310  !*************************************************************************
320  !
330  !
340   N=SIZE(X,1)
350  !
360  !
370  !**********************************
380  ! RECHERCHE DE S1 ET SN
390  !**********************************
400  !
410  !
420   C1=-(1/(X(2)-X(1))+1/(X(3)-X(1))+1/(X(4)-X(1)))
430   C2=1/(X(2)-X(1))*(X(3)-X(1))/(X(3)-X(2))*(X(4)-X(1))/(X(4)-X(2))
440   C3=1/(X(3)-X(1))*(X(2)-X(1))/(X(2)-X(3))*(X(4)-X(1))/(X(4)-X(3))
450   C4=1/(X(4)-X(1))*(X(2)-X(1))/(X(2)-X(4))*(X(3)-X(1))/(X(3)-X(4))
460   S(1)=C1*Y(1)+C2*Y(2)+C3*Y(3)+C4*Y(4)
470   Cn=-(1/(X(N-1)-X(N))+1/(X(N-2)-X(N))+1/(X(N-3)-X(N)))
480   Cn1=1/(X(N-1)-X(N))*(X(N-2)-X(N))/(X(N-2)-X(N-1))*(X(N-3)-X(N))/(X(N-3)-X(N-1))
490   Cn2=1/(X(N-2)-X(N))*(X(N-1)-X(N))/(X(N-1)-X(N-2))*(X(N-3)-X(N))/(X(N-3)-X(N-2))
500   Cn3=1/(X(N-3)-X(N))*(X(N-1)-X(N))/(X(N-1)-X(N-3))*(X(N-2)-X(N))/(X(N-2)-X(N-3))
510   S(N)=Cn*Y(N)+Cn1*Y(N-1)+Cn2*Y(N-2)+Cn3*Y(N-3)
520  !
530  !
540  !*************************************************************
550  ! DETERMINATION DES COEFFICIENTS S(2) A S(N-1)
560  !*************************************************************
570  !
580  !
590   ALLOCATE At(1:N-2),Bt(1:N-2),Ct(1:N-2),R(1:N-2),U(1:N-2),Gam(1:N-2)
```

```
600  !
610  !***********************************************
620  ! DETERMINATION DES COEFF POUR N=2 JUSQU'A N=N-1
630  !***********************************************
640  !
650   R(1)=3*(X(3)-X(2))*(Y(1)-Y(2))/(X(1)-X(2))
660   R(1)=R(1)+3*(X(2)-X(1))*(Y(2)-Y(3))/(X(2)-X(3))
670   R(1)=R(1)-S(1)*(X(3)-X(2))
680   R(N-2)=3*(X(N)-X(N-1))*(Y(N-2)-Y(N-1))/(X(N-2)-X(N-1))
690   R(N-2)=R(N-2)+3*(X(N-1)-X(N-2))*(Y(N-1)-Y(N))/(X(N-1)-X(N))
700   R(N-2)=R(N-2)-S(N)*(X(N-1)-X(N-2))
710   FOR I=2 TO N-3 STEP 1
720    R(I)=3*(X(I+2)-X(I+1))*(Y(I)-Y(I+1))/(X(I)-X(I+1))
730    R(I)=R(I)+3*(X(I+1)-X(I))*(Y(I+1)-Y(I+2))/(X(I+1)-X(I+2))
740   NEXT I
750   At(1)=0
760   Bt(1)=2*(X(3)-X(1))
770   Ct(1)=X(2)-X(1)
780   At(N-2)=X(N)-X(N-1)
790   Bt(N-2)=2*(X(N)-X(N-2))
800   Ct(N-2)=0
810   FOR I=2 TO N-3
820    At(I)=X(I+2)-X(I+1)
830    Bt(I)=2*(X(I+2)-X(I))
840    Ct(I)=X(I+1)-X(I)
850   NEXT I
860   !
870   !
880   !*****************************************************
890   !   RESOLUTION DE LA MATRICE TRIDIAGONALE
900   !*****************************************************
910   !
920   !
930   IF Bt(1)=0 THEN
940    PRINT "ARRETE LE PROGRAMME, AJOUTE OPTION POUR EQUATION"
950    SUBEXIT
960   END IF
970   Bet=Bt(1)
980   U(1)=R(1)/Bet
990   FOR J=2 TO N-2
1000   Gam(J)=Ct(J-1)/Bet
1010   Bet=Bt(J)-At(J)*Gam(J)
1020   IF Bet=0 THEN
1030    PRINT "ALGORITHME NE FONCTIONNE PAS POUR CE CAS"
1040    SUBEXIT
1050   END IF
1060   U(J)=(R(J)-At(J)*U(J-1))/Bet
1070  NEXT J
1080  FOR J=N-3 TO 1 STEP -1
1090   U(J)=U(J)-Gam(J+1)*U(J+1)
1100  NEXT J
1110  !
1120  !
1130  !***********************************************
1140  ! U(J) CONTIENT LES VALEURS POUR S(2) A S(N-1)
1150  !***********************************************
1160  !
1170  !
1180  FOR J=2 TO N-1
1190   S(J)=U(J-1)
1200  NEXT J
1210  DEALLOCATE At(*),Bt(*),Ct(*),R(*),U(*),Gam(*)
1220 SUBEND
1230 !
1240 !
```

```
1250 !
1260 !
1270 SUB Optspl3b(X(*),Y(*),S(*))
1280 !
1290 !
1300 !*****************************************************************
1310 !
1320 !    SOUS-PROGRAMME Optspl3b
1330 !
1340 !       FONCTION:  TROUVE S(1)...S(N) A PARTIR DE LA MINIMISATION
1350 !              DE LA NORME 2
1360 !
1370 !       ENTRER:  X(*),Y(*) OU X(*) SONT LES POINTS DE MESURE
1380 !            ET Y(*) LA MESURE
1390 !
1400 !       SORTIE: S(*)  LES VALEURS DE S(1)...S(N)
1410 !
1420 !*****************************************************************
1430 !
1440 !
1450  N=SIZE(X,1)
1460  ALLOCATE Dt(1:N-1),Deltay(1:N-1)
1470  ALLOCATE Af(1:N-1),Bt(1:N-1),Gm(1:N-1),A(1:N-1)
1480 !
1490 !********* INITIALISATION *********************************
1500 !
1510  FOR I=1 TO N-1
1520   Dt(I)=X(I+1)-X(I)
1530   Deltay(I)=Y(I+1)-Y(I)
1540   S(I)=Deltay(I)/Dt(I)
1550  NEXT I
1560  Af(1)=0
1570  Bt(1)=1
1580  Gm(1)=0
1590  Af(2)=0
1600  Bt(2)=0
1610  Gm(2)=1
1620  Sum11=0
1630  Sum12=0
1640  Sum22=0
1650  Sum1=0
1660  Sum2=0
1670  FOR I=3 TO N-1
1680   Q=(Dt(I)+Dt(I-1))/(Dt(I-1)+Dt(I-2))
1690   P=Q-2-3*Dt(I)/Dt(I-1)
1700   R=S(I)-S(I-1)*(1+Q)+Q*S(I-2)
1710   Af(I)=P*Af(I-1)-Q*Af(I-2)+R
1720   Bt(I)=P*Bt(I-1)-Q*Bt(I-2)
1730   Gm(I)=P*Gm(I-1)-Q*Gm(I-2)
1740  NEXT I
1750  FOR I=1 TO N-1
1760   Dx3=Dt(I)^3
1770   Sum11=Sum11+Bt(I)^2/Dx3
1780   Sum22=Sum22+Gm(I)^2/Dx3
1790   Sum12=Sum12+Gm(I)*Bt(I)/Dx3
1800   Sum1=Sum1-Af(I)*Bt(I)/Dx3
1810   Sum2=Sum2-Af(I)*Gm(I)/Dx3
1820  NEXT I
1830  A(1)=(Sum1*Sum22-Sum2*Sum12)/(Sum22*Sum11-Sum12^2)
1840  A(2)=(Sum1-Sum11*A(1))/Sum12
1850  FOR I=1 TO N-1
1860   A(I)=Af(I)+Bt(I)*A(1)+Gm(I)*A(2)
1870  NEXT I
1880  K=Dt(2)/Dt(1)
1890  B1=(S(2)-S(1)-(2+3*K)*A(1)-A(2))/(1+K)
```

```
1900   C1=S(1)-B1-A(1)
1910   S(1)=C1
1920   FOR I=1 TO N-1
1930     S(I+1)=A(I)+2*(Y(I)-Y(I+1))/(X(I)-X(I+1))-S(I)
1940   NEXT I
1950 SUBEND
1960 !


1970 !
1980 !
1990 SUB Valspl3(X(*),Y(*),S(*),Xm,Ym)
2000 !
2010 !
2020 !*************************************************************************
2030 !
2040 !  SOUS-PROGRAMME Valspl3
2050 !
2060 !       FONCTION : FAIT L'INTERPOLATION CUBIQUE DU POINT (Xm,Ym).
2070 !              POUR CE FAIRE LE S-P A BESOIN DES POINTS DE MESURE
2080 !              (X(*),Y(*)) ET DES VARIABLES S(*).
2090 !
2100 !       ENTREE:  X(*),Y(*),S(*),Xm
2110 !
2120 !       SORTIE: Ym
2130 !
2140 !*************************************************************************
2150 !
2160 !
2170   N=SIZE(X,1)
2180   Sortir=1
2190   I=0
2200   WHILE Sortir=1
2210     I=I+1
2220     SELECT I
2230     CASE <=N
2240       IF Xm<X(1) THEN
2250         PRINT "Xm N'EST PAS COMPRIS ENTRE ";X(1);" ET ";X(N)
2260         Sortir=0
2270       END IF
2280       IF Xm>X(N) THEN
2290         PRINT "Xm N'EST PAS COMPRIS ENTRE ";X(1);" ET ";X(N)
2300         Sortir=0
2310       END IF
2320       IF X(I)>Xm THEN
2330         Sortir=0
2340       END IF
2350     CASE >N
2360       Sortir=0
2370     END SELECT
2380   END WHILE
2390   IF Xm<X(N) AND Xm>=X(1) THEN
2400     I=I-1
2410     D=Y(I)
2420     C=S(I)
2430     B=(3*(Y(I)-Y(I+1))/(X(I)-X(I+1))-2*S(I)-S(I+1))/(X(I+1)-X(I))
2440     A=(S(I)+S(I+1)-2*(Y(I)-Y(I+1))/(X(I)-X(I+1)))/(X(I+1)-X(I))^2
2450     Ym=A*(Xm-X(I))^3+B*(Xm-X(I))^2+C*(Xm-X(I))+D
2460   END IF
2470   IF Xm=X(N) THEN
2480     Ym=Y(N)
2490   END IF
2500 SUBEND


2510 SUB Optspl3(X(*),Y(*),S(*))
```

```
2520 !
2530 !
2540 !*************************************************************
2550 !
2560 !    SOUS-PROGRAMME Optspl3
2570 !
2580 !        FONCTION:  TROUVE S(1)...S(N) A PARTIR DE LA MINIMISATION
2590 !                DE LA NORME 2
2600 !
2610 !        ENTRER:  X(*),Y(*) OU X(*) SONT LES POINTS DE MESURE
2620 !              ET Y(*) LA MESURE
2630 !
2640 !        SORTIE: S(*)  LES VALEURS DE S(1)...S(N)
2650 !
2660 !*************************************************************
2670 !
2680 !
2690  N=SIZE(X,1)
2700  ALLOCATE Dt(1:N-1),Deltay(1:N-1),A(1:N-1),B(1:N-1),C(1:N-1)
2710 !
2720 !********* INITIALISATION ******************************
2730 !
2740  FOR I=1 TO N-1
2750   Dt(I)=X(I+1)-X(I)
2760   Deltay(I)=Y(I+1)-Y(I)
2770   S(I)=Deltay(I)/Dt(I)
2780  NEXT I
2790  A(1)=1
2800  B(1)=0
2810  C(1)=0
2820  A(2)=0
2830  B(2)=1
2840  C(2)=0
2850  Sum11=0
2860  Sum12=0
2870  Sum22=0
2880  Sum1=0
2890  Sum2=0
2900  FOR I=3 TO N-1
2910   R=(Dt(I)+Dt(I-1))/(Dt(I-1)+Dt(I-2))
2920   P=R-2-3*Dt(I)/Dt(I-1)
2930   Q=S(I)-S(I-1)*(1+R)+R*S(I-2)
2940   A(I)=P*A(I-1)-R*A(I-2)
2950   B(I)=P*B(I-1)-R*B(I-2)
2960   C(I)=P*C(I-1)-R*C(I-2)+Q
2970  NEXT I
2980  FOR I=1 TO N-1
2990   Dx3=Dt(I)^3
3000   Sum11=Sum11+A(I)^2/Dx3
3010   Sum22=Sum22+B(I)^2/Dx3
3020   Sum12=Sum12+A(I)*B(I)/Dx3
3030   Sum1=Sum1-A(I)*C(I)/Dx3
3040   Sum2=Sum2-B(I)*C(I)/Dx3
3050  NEXT I
3060  A(1)=(Sum1*Sum22-Sum2*Sum12)/(Sum22*Sum11-Sum12^2)
3070  A(2)=(Sum1-Sum11*A(1))/Sum12
3080  FOR I=3 TO N-1
3090   A(I)=A(I)*A(1)+B(I)*A(2)+C(I)
3100  NEXT I
3110  K=Dt(2)/Dt(1)
3120  B(1)=(S(2)-S(1)-(2+3*K)*A(1)-A(2))/(1+K)
3130  C(1)=S(1)-B(1)-A(1)
3140  S(1)=C(1)
3150  FOR I=1 TO N-1
3160   S(I+1)=A(I)+2*(Y(I)-Y(I+1))/(X(I)-X(I+1))-S(I)
3170  NEXT I
```

```
3180  SUBEND


3190  SUB Der_valspl3(X(*),Y(*),S(*),Xm,Ym)
3200  !
3210  !
3220  !*************************************************************************
3230  !
3240  !   SOUS-PROGRAMME Valspl3
3250  !
3260  !        FONCTION : FAIT L'INTERPOLATION CUBIQUE DU POINT (Xm,Ym).
3270  !                POUR CE FAIRE LE S-P A BESOIN DES POINTS DE MESURE
3280  !                (X(*),Y(*)) ET DES VARIABLES S(*).
3290  !
3300  !        ENTREE:  X(*),Y(*),S(*),Xm
3310  !
3320  !        SORTIE: Ym
3330  !
3340  !*************************************************************************
3350  !
3360  !
3370   N=SIZE(X,1)
3380   Sortir=1
3390   I=0
3400   WHILE Sortir=1
3410    I=I+1
3420    SELECT I
3430    CASE <=N
3440     IF Xm<X(1) THEN
3450      PRINT "Xm N'EST PAS COMPRIS ENTRE ";X(1);" ET ";X(N)
3460      Sortir=0
3470     END IF
3480     IF Xm>X(N) THEN
3490      PRINT "Xm N'EST PAS COMPRIS ENTRE ";X(1);" ET ";X(N)
3500      Sortir=0
3510     END IF
3520     IF X(I)>Xm THEN
3530      Sortir=0
3540     END IF
3550    CASE >N
3560     Sortir=0
3570    END SELECT
3580   END WHILE
3590   IF Xm<X(N) AND Xm>=X(1) THEN
3600    I=I-1
3610    D=Y(I)
3620    C=S(I)
3630    B=(3*(Y(I)-Y(I+1))/(X(I)-X(I+1))-2*S(I)-S(I+1))/(X(I+1)-X(I))
3640    A=(S(I)+S(I+1)-2*(Y(I)-Y(I+1))/(X(I)-X(I+1)))/(X(I+1)-X(I))^2
3650    Ym=3*A*(Xm-X(I))^2+2*B*(Xm-X(I))+C
3660   END IF
3670   IF Xm=X(N) THEN
3680    Ym=S(N)
3690   END IF
3700   SUBEND


14   END
20   SUB Intspl2(X(*),Y(*),A(*),B(*),C(*))
30   !
40   !
50   !*************************************************************************
60   !
70   !  SOUS-PROGRAMME Intspl2
80   !
90   !   DESCRIPTION:  SOUS-PROGRAMME QUI PERMET DE DETERMINER S(1) A
```

```
100  !              S(N).  LA METHODE UTILISE CONSISTE A DETREMINER
110  !              LA DERIVE PREMIERE DES POINTS [X(1),Y(1)] ET
120  !              [X(N),Y(N)], D'ASSOCIER CES VALEURS A S(1) ET S(N)
130  !              ET A PARTIR DE CES VALEURS, ETABLIR UNE FONCTION
140  !              DE RECURANCE POUR DETERMINER S(2)...S(N-1).
150  !
160  !     ENTRER:  X(*),Y(*) OU X(*) REPRESENTE LES POINT DE MESURE
170  !              Y(*) LA VALEUR DE LA MESURE
180  !
190  !     SORTIE: S(*)
200  !
210  !*************************************************************************
220  !
230  !
231  OPTION BASE 1
240   N=SIZE(X,1)
260  !
270  !
280  !**********************************
290  ! RECHERCHE DE S1 ET SN
300  !**********************************
310  !
320  !
330   C1=-(1/(X(2)-X(1))+1/(X(3)-X(1))+1/(X(4)-X(1)))
340   C2=1/(X(2)-X(1))*(X(3)-X(1))/(X(3)-X(2))*(X(4)-X(1))/(X(4)-X(2))
350   C3=1/(X(3)-X(1))*(X(2)-X(1))/(X(2)-X(3))*(X(4)-X(1))/(X(4)-X(3))
360   C4=1/(X(4)-X(1))*(X(2)-X(1))/(X(2)-X(4))*(X(3)-X(1))/(X(3)-X(4))
370   B(1)=C1*Y(1)+C2*Y(2)+C3*Y(3)+C4*Y(4)
380  !
390  !
400  !
401   A(1)=(Y(2)-Y(1))/(X(2)-X(1))^2-B(1)/(X(2)-X(1))
402   C(1)=Y(1)
410   FOR I=1 TO N-2
420    B(I+1)=2*(Y(I+1)-Y(I))/(X(I+1)-X(I))-B(I)
430    C(I+1)=Y(I+1)
440    A(I+1)=(Y(I+2)-Y(I+1))/(X(I+2)-X(I+1))^2-B(I+1)/(X(I+2)-X(I+1))
450   NEXT I
1190 SUBEND
1200 !
1210 !


1220 !
1230 !
1240 SUB Optspl2(X(*),Y(*),A(*),B(*),C(*))
1241 OPTION BASE 1
1250 !
1260 !
1270 !*************************************************************************
1280 !
1290 !     SOUS-PROGRAMME Optspl2
1300 !
1310 !        FONCTION:  TROUVE S(1)...S(N) A PARTIR DE LA MINIMISATION
1320 !                   DE LA NORME 2
1330 !
1340 !        ENTRER:  X(*),Y(*) OU X(*) SONT LES POINTS DE MESURE
1350 !                 ET Y(*) LA MESURE
1360 !
1370 !        SORTIE: S(*)  LES VALEURS DE S(1)...S(N)
1380 !
1390 !*************************************************************************
1400 !
1410 !
1420   N=SIZE(X,1)
1430   ALLOCATE Dt(1:N-1)
```

```
1440   ALLOCATE S(1:N-1)
1450 !
1460 !********* INITIALISATION ********************************
1470 !
1480   FOR I=1 TO N-1
1490     Dt(I)=X(I+1)-X(I)
1510     S(I)=(Y(I+1)-Y(I))/Dt(I)
1520     C(I)=Y(I)
1530   NEXT I
1600   Sum11=0
1610   Sum12=0
1650   FOR I=1 TO N-1
1651     IF I=1 THEN
1652       Bt=0
1653     ELSE
1660       Bt=Bt+COS(PI*(K-1))*(S(I)-S(I-1))
1661     END IF
1670     Sum11=Sum11+1/Dt(I)
1680     Sum12=Sum12+Bt/Dt(I)
1720   NEXT I
1721   A(1)=-Sum12/Sum11
1722   B(1)=S(1)-A(1)
1730   FOR I=2 TO N-1
1740     B(I)=B(I-1)+2*A(I-1)
1750     A(I)=S(I)-B(I)
1760     A(I-1)=A(I-1)/Dt(I-1)
1800   NEXT I
1810   A(N-1)=A(N-1)/Dt(N-1)
1970 SUBEND


1980 !
1990 !
2000 !
2010 SUB Valspl2(X(*),Y(*),A(*),B(*),C(*),Xm,Ym)
2011 OPTION BASE 1
2020 !
2030 !
2040 !*************************************************************************
2050 !
2060 !   SOUS-PROGRAMME Valspl2
2070 !
2080 !        FONCTION : FAIT L'INTERPOLATION CUBIQUE DU POINT (Xm,Ym).
2090 !                   POUR CE FAIRE LE S-P A BESOIN DES POINTS DE MESURE
2100 !                   (X(*),Y(*)) ET DES VARIABLES S(*).
2110 !
2120 !        ENTREE:  X(*),Y(*),S(*),Xm
2130 !
2140 !        SORTIE: Ym
2150 !
2160 !*************************************************************************
2170 !
2180 !
2190   N=SIZE(X,1)
2200   Sortir=1
2210   I=0
2220   WHILE Sortir=1
2230     I=I+1
2240     IF Xm<X(1) THEN
2250       PRINT "Xm N'EST PAS COMPRIS ENTRE ";X(1);" ET ";X(N)
2260       Sortir=0
2270     END IF
2280     IF Xm>X(N) THEN
2290       PRINT "Xm N'EST PAS COMPRIS ENTRE ";X(1);" ET ";X(N)
2300       Sortir=0
2310     END IF
```

```
2311    IF Xm=X(N) THEN
2312      I=N
2313      Sortir=0
2314    END IF
2320    IF X(I)>Xm THEN
2330      Sortir=0
2340    END IF
2350   END WHILE
2360   IF Xm<=X(N) AND Xm>=X(1) THEN
2370    I=I-1
2380    Ym=A(I)*(Xm-X(I))^2+B(I)*(Xm-X(I))+C(I)
2390   END IF
2400  SUBEND


2401  !
2402  !
2410  SUB Der_vspl2(X(*),Y(*),A(*),B(*),C(*),Xm,Ym)
2411  OPTION BASE 1
2420   N=SIZE(X,1)
2430   Sortir=1
2440   I=0
2450   WHILE Sortir=1
2460    I=I+1
2470    IF Xm<X(1) THEN
2480      PRINT "Xm N'EST PAS COMPRIS ENTRE ";X(1);" ET ";X(N)
2490      Sortir=0
2500    END IF
2510    IF Xm>X(N) THEN
2520      PRINT "Xm N'EST PAS COMPRIS ENTRE ";X(1);" ET ";X(N)
2530      Sortir=0
2540    END IF
2541    IF Xm=X(N) THEN
2542      I=N
2543      Sortir=0
2544    END IF
2550    IF X(I)>Xm THEN
2560      Sortir=0
2570    END IF
2580   END WHILE
2590   IF Xm<=X(N) AND Xm>=X(1) THEN
2600    I=I-1
2610    Ym=2*A(I)*(Xm-X(I))+B(I)
2620   END IF
2630   SUBEND
```

Les programmes pour la procédure 3 sont pour les données réelles. L'algorithme de calcul est indiqué visiblement.

Les programmes pour la procédure 3 (PCR3) en MATLAB sont:

```
clear
%
%    cette simulation est pour tmin=0, tmax=30, pmin=3.75 pmax=30;
```

```
%    et un nombre de temperature egal a 7
%
%    pour les deux sinus
%
%        l'erreur sur la temperature est de .05
%        l'erreur sur la tension est de .002
%        l'erreur sur la pression est de .1
%
%    pour le guide
%
%        l'erreur sur la temperature est de .05
%        l'erreur sur la tension est de .002
%        l'erreur sur la pression est de .1
%

rand('normal');
load partir
if partir ==0
  debutp=1;
  debutt=1;
  clc
  niter_t=input('entrer le nombre d''iteration de temperature [3]: ');
  if isempty(niter_t)
    niter_t=3;
  end

  niter_p=input('entrer le nombre d''iteration de pression [5]: ');
  if isempty(niter_p)
    niter_p=5;
  end

  Memaxs=zeros(niter_p,niter_t);
  Memaxc=zeros(niter_p,niter_t);
  Memoys=zeros(niter_p,niter_t);
  Memoyc=zeros(niter_p,niter_t);
  nbt=7;
else
  debutp=i;
  debutt=j;
end

pmin=2.75;, pmax=31;, tmin=15;, tmax=35;
sdp=.01;,sdv=.002;,sdt=.01;
var(1)=sdv^2;
var(2)=sdt^2;
var(3)=sdp^2;


%*************** pour les deux sinus ********************


for j=debutt:niter_t

  if debutp==1
    interpoint=5;
  end
  [mat_ps, mat_pc, vec_t]=trv_pt(tmin,tmax,nbt);

  for i=debutp:niter_p
    partir=1;
    save partir partir j i Memaxs Memaxc Memoys Memoyc interpoint nbt ...
    niter_t niter_p

    [mat_psn, mat_pcn]=det_p(mat_ps,mat_pc,interpoint);
    [los1,los2]=size(mat_psn);
```

```
[loc1,loc2]=size(mat_pcn);
mat_psnp=mat_psn+sdp*rand(mat_psn);

I=find(mat_psnp<0);
if isempty(I)==0
  mat_psnp(I)=zeros(length(I),1);
end

I=find(mat_psnp>30);
if isempty(I)==0
  mat_psnp(I)=30*ones(length(I),1);
end

mat_pcnp=mat_pcn+sdp*rand(mat_pcn);

I=find(mat_pcnp<0);
if isempty(I)==0
   mat_pcnp(I)=zeros(length(I),1);
end

per=det_freq;
I=find(mat_pcnp>(30-per/4));
if isempty(I)==0
   mat_pcnp(I)=(30-per/4)*ones(length(I),1);
end
vec_tp=vec_t+sdt*rand(vec_t);
vec_tp(1)=tmin;
vec_tp(length(vec_tp))=tmax;
[mat_vs, mat_vc]=trv_v(mat_psnp, mat_pcnp, vec_tp);
[mat_cts,mat_cps,mat_ctc,mat_cpc]=trv_c(mat_ps,mat_pc,vec_t);

mat_vsp=mat_vs+sdv*rand(mat_vs);
mat_vcp=mat_vc+sdv*rand(mat_vc);

[mas,mbs,mcs,mds]=eta_abcd(mat_psn,mat_vsp,mat_ps,mat_cts,mat_cps,var);
[mac,mbc,mcc,mdc]=eta_abcd(mat_pcn,mat_vcp,mat_pc,mat_ctc,mat_cpc,var);

% ici nous avons pour les deux sinus ma mb mc md

%********************* pour le guide ************************

[vec_pg,vec_tg]=detg_pt(tmin,tmax,pmin,pmax,15,15);

vec_pgp=vec_pg+sdp*rand(vec_pg);
vec_tgp=vec_tg+sdt*rand(vec_tg);

mat_vg=det_vg(vec_pgp,vec_tgp);
mat_vgp=mat_vg+sdv*rand(mat_vg);
[vec_ctg,vec_cpg]=detg_c(vec_pg,vec_tg);


[mag,mbg,mcg,mdg]=etag_abc(vec_tg,mat_vgp,vec_ctg,vec_cpg,var);

save trans mat_ps mat_psn mat_pc mat_pcn vec_t vec_pg vec_tg mat_vgp ...
mat_vsp mat_vcp mas mbs mcs mds mac mbc mcc mdc mag mbg mcg mdg ...
pmin pmax tmin tmax var

[emax,emoy]=evalu2(var);

Memax(i,j)=emax;
Memoy(i,j)=emoy;

%
%   calcul des nouvelles matrices mat_pcn mat_psn
%
%    [mat_psn,mat_pcn]=det_mat(mat_retps,mat_retpc,mat_psn,mat_pcn);
```

```
    interpoint=interpoint+2;

    save edit;
    clear
    load edit;
    rand('normal');

  end
%  load val_init;
%  [mat_ps,mat_psn,mat_pc,mat_pcn,vec_t]=det_tsc(v_tret,pmin,pmax,mat_ps,mat_psn,mat_pc,mat_pcn,vec_t);
  nbt=nbt+2;
end
partir=0;
save partir partir;
save r_0514 j i Memaxs Memaxc Memoys Memoyc interpoint nbt ...
   niter_t niter_p pmin pmax tmin tmax var□




function [emax, emoy ]=evalu2(var)
%
%  programme d'evaluation
%  retourne: 4 valeurs d'erreurs (2 erreurs moyennes, 2 erreurs max)
%          2 matrices de pression
%          une valeur de temperature
%
%  pour fonctionner besoins de:
%      mat_ps, mat_psn, mat_pc, mat_pcn, vec_t, vec_pg, vec_tg,mat_vgp
%      mat_vsp, mat_vcp, mas, mbs, mcs, mds, mac, mbc, mcc, mdc, mag,
%      mbg, mcg, mdg, pmin, pmax, tmin, tmax.
%
%



load trans;
rand('normal');
sdv=sqrt(var(1));
sdt=sqrt(var(2));


CS=1;
CC=0;

nbinters=length(mat_ps(:,1));
nbinterc=length(mat_pc(:,1));

load donnee1

v_teva=temp;
v_pev=p(2:length(p)-3);



%
%  evaluation de mat_psev
%


[mat_vs,mat_vc]=trv_v(mat_ps,mat_pc,vec_t);

%
%  calcul des matrices de tension
%  m_vs, m_vc, m_vg
```

```
%

   [m_vs, m_vc, m_vg]=trv_vscg(v_teva,v_pev);

%
%  ajout des perturbations
%

   m_vs=m_vs+sdv*rand(m_vs);
   m_vc=m_vc+sdv*rand(m_vc);
   m_vg=m_vg+sdv*rand(m_vg);
   v_tevap=v_teva+sdt*rand(v_teva);
   if v_tevap(1)<15
     v_tevap(1)=15;
   end
   if v_tevap(length(v_tevap))>35
     v_tevap(length(v_tevap))=35;
   end

%
%  calcul de la pression reconstitue
%

   nt=length(v_teva);
   n=length(v_pev);
   for k=1:nt
            temp=v_tevap(k);
            for l=1:n
               entree=0;
               if l==1
                 entree=1;
               else
                 if v_pev(l-1)~=v_pev(l)
                          entree=1;
                 end
               end

            if entree==1

%*********************************************************
%*********************************************************
%***************  DEBUT DE L'ALGORITHME  ***************
%*********************************************************
%*********************************************************

%
%      CALCUL A PARTIR DU GUIDE
%

                  for kk=1:length(vec_pg)
                          v_int(kk)=valspl3(vec_tg,mag(kk,:),mbg(kk,:),mcg(kk,:),mdg(kk,:),temp);
                  end
                  [ag,bg,cg,dg]=intspl3(v_int,vec_pg);
                  if m_vg(l,k)<min(v_int)
                          tm_vg=min(v_int);
                  elseif m_vg(l,k)>max(v_int)
                          tm_vg=max(v_int);
                  else
                          tm_vg=m_vg(l,k);
                  end
                  pg=valspl3(v_int,ag,bg,cg,dg,tm_vg);


%
%   choix du capteur
%
```

```
        if abs(m_vc(l,k)-2.5)>abs(m_vs(l,k)-2.5)
                tenm=m_vs(l,k);
                mat_p=mat_ps;
                mat_v=mat_vs;
                mat_pn=mat_psn;
                mat_vn=mat_vsp;
                ma=mas;
                mb=mbs;
                mc=mcs;
                md=mds;
                choix=CS;
                nbinter=nbinters;
        else
                tenm=m_vc(l,k);
                mat_p=mat_pc;
                mat_v=mat_vc;
                mat_pn=mat_pcn;
                mat_vn=mat_vcp;
                ma=mac;
                mb=mbc;
                mc=mcc;
                md=mdc;
                nbinter=nbinterc;
                choix=CC;
        end
        choix

%
%  calcul de la pression a partir du sinus choisit
%
        boucle=1;
        bou_count=1;
        while(boucle)

         l=find(temp<=vec_t);
         vec_p=mat_p(:,l(1));
         vec_pn=mat_pn(:,l(1));
         vec_v=mat_v(:,l(1));
         pos_temp=l(1);
         count=1;
         for j=1:2:length(vec_p-1)
                 if vec_p(j)~=vec_p(j+1)
                   vec_ptrs(count)=vec_p(j);
                   vec_vtrs(count)=vec_v(j);
                   count=count+1;
                   vec_ptrs(count)=vec_p(j+1);
                   vec_vtrs(count)=vec_v(j+1);
                   count=count+1;
                 end
         end
         [ec_min,ll]=sort(abs(vec_ptrs-pg));
         pos=ll(bou_count);
         if (floor(pos/2)*2)==pos
                 posmax=pos;
                 posmin=pos-1;
         else
                 posmax=pos+1;
                 posmin=pos;
         end
         psup=vec_ptrs(posmax);
         penteref=vec_ptrs(posmax)-vec_ptrs(posmin);
         penteref=(vec_vtrs(posmax)-vec_vtrs(posmin))/penteref;
         if penteref>0
                 penteref=1;
         else
                 penteref=-1;
```

```
        end
        psup_ant=psup;
        if pos_temp~=1
            while pos_temp~=1
                pos_temp=pos_temp-1;
                [Srt,Isrt]=sort(abs(mat_p(2:2:nbinter,pos_temp)-psup));
                Isrt=Isrt*2;
                for kk=1:length(Isrt)
                    if sign(mat_v(Isrt(kk),pos_temp) ...
                    -mat_v(Isrt(kk)-1,pos_temp))==penteref
                      break;
                    end
                end
                psup=mat_p(Isrt(kk),pos_temp);
                if abs(psup_ant-psup)>3
                    pos_temp=pos_temp+1;
                    psup=psup_ant;
                    break;
                else
                    psup_ant=psup;
                end
            end
        end

%
%  recherche des intervalles pour interpolation
%
        inc=1;
        continue=0;
        for kk=pos_temp:length(vec_t)
            for jj=2:2:nbinter
              if mat_p(jj,kk)==psup
                if mat_p(jj,kk)~=mat_p((jj-1),kk)
                    pente=mat_p(jj,kk)-mat_p((jj-1),kk);
                    pente=(mat_v(jj,kk)-mat_v((jj-1),kk))/pente;
                    pente=sign(pente);
                    if pente==penteref
                      if (pente*tenm)>=(pente*(mat_v(jj-1,kk)-.1)) ...
                          &(pente*tenm)<=((mat_v(jj,kk)+.1)*pente)
                          mat_pintval1=mat_p((jj-1),kk);
                          mat_pintval2=mat_p(jj,kk);
                          imin=find(mat_pn(:,kk)==mat_pintval1);
                          imax=find(mat_pn(:,kk)==mat_pintval2);
                          if length(imin)>1 & length(imax)==1
                             ecc_min=99e99;
                             for lk=1:length(imin)
                                 if (imax-imin(lk))<ecc_min & (imax-imin(lk))>0
                                     ipos=lk;
                                     ecc_min=imax-imin(lk);
                                 end
                             end
                             imin=imin(ipos);
                          end
                          ma_iv(:,inc)=ma(imin:imax,kk);
                          mb_iv(:,inc)=mb(imin:imax,kk);
                          mc_iv(:,inc)=mc(imin:imax,kk);
                          md_iv(:,inc)=md(imin:imax,kk);
                          mv_iv(:,inc)=mat_vn(imin:imax,kk);
                          mp_iv(:,inc)=mat_pn(imin:imax,kk);
                          vt_iv(inc)=vec_t(kk);
                          inc=inc+1;
                          if inc>4
                            continue=1;
                          end
                      end
                    end
```

```
                    end
                  end
                end

                if kk<length(vec_t)
                  [Srt,Isrt]=sort(abs(mat_p(2:2:nbinter,kk+1)-psup));
                  Isrt=Isrt*2;
                  for lk=1:length(Isrt)
                      if sign(mat_v(Isrt(lk),kk+1)-mat_v(Isrt(lk)-1,kk+1))==penteref
                        break;
                      end
                  end
                  psup=mat_p(Isrt(lk),kk+1);
                end
            end

%
%  interpolation de p a partir de ma_iv, mb_iv, mc_iv, md_iv, mp_iv
%  mv_iv, vt_iv
%
            if continue==1
                inc=inc-1;
                for kk=1:inc
                  if tenm<min(mv_iv(:,kk))
                      ttenm=min(mv_iv(:,kk));
                  elseif tenm>max(mv_iv(:,kk));
                      ttenm=max(mv_iv(:,kk));
                  else
                      ttenm=tenm;
                  end
                   vp_int(kk)=newspl3(mp_iv(:,kk),mv_iv(:,kk),ma_iv(:,kk),mb_iv(:,kk),mc_iv(:,kk),md_iv(:,kk),ttenm);
                end
                [v_a,v_b,v_c,v_d]=intspl3(vt_iv,vp_int);
                p_int(l,k)=valspl3(vt_iv,v_a,v_b,v_c,v_d,temp);
                if abs(p_int(l,k)-pg)<1 | bou_count==2
                  boucle=0;
                  if bou_count==2
                    if abs(p_int_pre-pg)<abs(p_int(l,k)-pg)
                      p_int(l,k)=p_int_pre;
                    end
                  end
                else
                  bou_count=2;
                  p_int_pre=p_int(l,k);
                end
                if abs(p_int(l,k)-pg)>.1
                  p_int(l,k)=pg;
                end
            else
                p_int(l,k)=pg;
                boucle=0;
            end
            clear ma_iv mb_iv mc_iv md_iv mp_iv mv_iv vt_iv v_int vp_int
            clear vec_ptrs vec_vtrs
          end

%*************************************************************************
%*************************************************************************
%************************   FIN DE L'ALGORITHME   ************************
%*************************************************************************
%*************************************************************************
        else
          p_int(l,k)=p_int(l-1,k);
        end
      end
  end
```

```
%
%   calcul des erreurs
%


mat_pev=v_pev'*ones(1,nt);
err=abs(p_int-mat_pev);
emax=max(max(err));
emoy=sum(sum(err.^2));
[n,m]=size(err);
nbelem=n*m;
emoy=sqrt(emoy/nbelem);



function [mat_cts, mat_cps, mat_ctc, mat_cpc]=trv_c(mat_ps,mat_pc,vec_t)
%
%   fonction qui determine les constantes pour variances
%
%   [mat_cts, mat_cps, mat_ctc, mat_cpc]=det_c(mat_ps,mat_pc,vec_t)
%


[ns,ms]=size(mat_ps);
v1=carte(vec_t,mat_ps(:,1));
[nc,mc]=size(mat_pc);
v2=carte(vec_t,mat_pc(:,1));
A=2*pi/25;
B=pi/50;

mat_cps1=(2.5*cos(A*mat_ps+B*v1)*A).^2;
mat_cts1=(2.5*cos(A*mat_ps+B*v1)*B).^2;
mat_cps=(mat_cps1(2:2:ns,:)+mat_cps1(1:2:ns-1,:))/2;
mat_cts=(mat_cts1(2:2:ns,:)+mat_cts1(1:2:ns-1,:))/2;


mat_cpc1=(2.5*sin(A*mat_ps+B*v1)*A).^2;
mat_ctc1=(2.5*sin(A*mat_ps+B*v1)*B).^2;
mat_cpc=(mat_cpc1(2:2:nc,:)+mat_cpc1(1:2:nc-1,:))/2;
mat_ctc=(mat_ctc1(2:2:nc,:)+mat_ctc1(1:2:nc-1,:))/2;


function [mat_vs, mat_vc]=det_v(mat_psn, mat_pcn, vec_t)
%
%   fonction qui tourne deux matrice de tension.
%   [mat_vs, mat_vc]=det_v(mat_psn, mat_pcn, vec_t)
%


n=length(mat_psn(:,1));
m=length(vec_t);
mat_vs=zeros(n,m);
mat_vc=zeros(n,m);
mat_t=ones(n,m)*diag(vec_t);


A=2*pi/25;
B=pi/50;


mat_vs=2.5+2.5*sin(A.*mat_psn+B.*mat_t);
mat_vc=2.5+2.5*sin(A.*mat_pcn+B.*mat_t-pi/2);
```

```
function [ma,mb,mc,md]=eta_abcd(mat_x,mat_y,mat_pos,mat_ct,mat_cp,variance)
%
% fonction qui trouve des matrices de coefficients.
% Ces coefficients sont trouves a partir de mat_x, mat_y, mat_pos
%
% [ma,mb,mc,md] = eta_abcd(mat_x,mat_y,mat_pos,mat_ct,mat_cp,variance)
%
% variance = vecteur de trois elements
%          variance(1) = variance de la tension
%          variance(2) = variance de la temperature
%          variance(3) = variance de la pression
%
%          mat_ct = matrice de constante pour variance
%          mat_cv = matrice de constante pour variance
%
%


[n m]=size(mat_pos);
[dim1, dim2]=size(mat_x);

ma=zeros(dim1,dim2);
mb=zeros(dim1,dim2);
mc=zeros(dim1,dim2);
md=zeros(dim1,dim2);

for i=1:m
 for j=1:2:n-1
    pmin=mat_pos(j,i);
    pmax=mat_pos(j+1,i);
    pos_min=find(mat_x(:,i)==pmin);
    pos_max=find(mat_x(:,i)==pmax);
    if length(pos_min)>1 & length(pos_max)==1
      ec_min=99e99;
      for kk=1:length(pos_min)
        if (pos_max-pos_min(kk))<ec_min & (pos_max-pos_min(kk))>0
          posi_pos=kk;
          ec_min=pos_max-pos_min(kk);
        end
      end
      pos_min=pos_min(posi_pos);
    end
    if length(pos_min)==1;
      v_x=mat_x(pos_min:pos_max,i);
      v_y=mat_y(pos_min:pos_max,i);
      S=pos_max-pos_min+1;
      S=(S-sqrt(S*3));
      kk=(j+1)/2;
      dy=mat_cp(kk,i)*variance(3)+mat_ct(kk,i)*variance(2)+variance(1);
      dy=sqrt(dy);
      [va,vb,vc,vd]=smospl3c(v_x,v_y,dy,S);
      ma(pos_min:pos_max,i)=va;
      mb(pos_min:pos_max-1,i)=vb;
      mc(pos_min:pos_max-1,i)=vc;
      md(pos_min:pos_max-1,i)=vd;
    end
 end
end


function [vec_ctg, vec_cpg]=detg_c(vec_pg,vec_tg)
%
% fonction qui determine les constantes pour variances
%
% [vec_ctg, vec_cpg]=detg_c(vec_pg,vec_tg)
%
```

```
ng=length(vec_pg);
[temp,pres]=carte(vec_tg,vec_pg);
A=2*pi/300;
B=pi/200;
C=pi/24;

mat_cpg=(2.5*cos(A*pres+B*temp+C)*A).^2;
mat_ctg=(2.5*cos(A*pres+B*temp+C)*B).^2;
mat_cpg=mat_cpg';
mat_ctg=mat_ctg';

vec_cpg=sum(mat_cpg)/length(mat_cpg(:,1));
vec_ctg=sum(mat_ctg)/length(mat_ctg(:,1));


function [vec_pg,vec_tg]=detg_pt(tmin,tmax,pmin,pmax,nbt,nbp)
%
%  fonction qui trouve un vecteur de pression, un vecteur de
%  temperature et une matrice de tension.
%
%  [vec_pg,vec_tg,mat_vg]=detg_ptv(tmin,tmax,pmin,pmax,nbt,nbp)
%

past=(tmax-tmin)/(nbt-1);
pasp=(pmax-pmin)/(nbp-1);

vec_pg=pmin:pasp:pmax;
vec_tg=tmin:past:tmax;


function [mag,mbg,mcg,mdg]=etag_abc(vec_tg,mat_vg,vec_ctg,vec_cpg,variance)
%
% fonction qui trouve des matrices de coefficients.
% Ces coefficients sont trouves a partir de mat_x, mat_y, mat_pos
%
% [mag,mbg,mcg,mdg] = etag_abc(vec_t,mat_vg,vec_ctg,vec_cpg,variance)
%
% variance = vecteur de trois elements
%         variance(1) = variance de la tension
%         variance(2) = variance de la temperature
%         variance(3) = variance de la pression
%
%         vec_ctg = matrice de constante pour variance
%         vec_cvg = matrice de constante pour variance
%
%


m=length(mat_vg(:,1));
N=length(mat_vg(1,:));
S=floor(N-sqrt(3*N));
[dim1, dim2]=size(mat_vg);

mag=zeros(dim1,dim2);
mbg=zeros(dim1,dim2);
mcg=zeros(dim1,dim2);
mdg=zeros(dim1,dim2);

for i=1:m
      v_x=vec_tg;
      v_y=mat_vg(i,:);
      dy=vec_cpg(i)*variance(3)+vec_ctg(i)*variance(2)+variance(1);
      dy=sqrt(dy);
      [va,vb,vc,vd]=smospl3c(v_x,v_y,dy,S);
```

```
      mag(i,:)=va';
      mbg(i,1:N-1)=vb';
      mcg(i,1:N-1)=vc';
      mdg(i,1:N-1)=vd';
end


function [mat_vs, mat_vc, mat_vg]=trv_vscg(vec_t,vec_p);
%
%   fonction qui calcule les vecteurs de tension
%   vec_vs, vec_vc, vec_vg a partir de vec_t et vec_p
%
%   [vec_vs, vec_vc, vec_vg]=det_vscg(vec_t,vec_p)
%

A=1/25;,B=1/100;,C=-1/4;

[x,y]=carte(vec_t,vec_p);

mat_vs=5*(cos((A*y+B*x+C)*pi)).^2;

C=1/2;

mat_vc=5*(cos((A*y+B*x+C)*pi)).^2;

A=1/300;,B=1/400;,C=25/48;

mat_vg=5*(cos((A*y+B*x+C)*pi)).^2;


function ym=valspl3(x,a,b,c,d,xm)
%
%   fonction qui evalue la valeur ym a partir du polynome
%   f[x(i)]=a(i) + b(i)*(x-x(i)) + c(i)*(x-x(i))^2 + d(i)*(x-x(i))^3
%
%
%   ym=valspl3(x,a,b,c,d,xm)
%
%   entree:
%         x = vecteur de base ou x(1)<x(2)<...<x(n)
%         a, b, c, d = vecteur de coeff.
%         xm = vecteur de valeurs a interpoler
%
%  sortie:
%         ym = vecteur de valeurs trouve
%



[n m]=size(xm);

if m>1, dim=m;, else dim=n;, end

[n m]=size(x);

if m>1, n=m;, end

for j=1:dim

   suivant=0;
   vec_p=find(x>xm(j));
   if isempty(vec_p);
     if xm(j)==x(n)
       pos=n-1;
```

```
      else
        disp(sprintf('xm n"est pas compris entre %g et %g',x(1),x(n)));
        suivant=1;
      end
    elseif isempty(vec_p)==0
      if vec_p(1)==1
        disp(sprintf('xm n"est pas compris entre %g et %g',x(1),x(n)));
        suivant=1;
      else
        pos=vec_p(1)-1;
      end
    end
    if suivant==0;
      h=xm(j)-x(pos);
      ym(j)=a(pos)+b(pos)*h+c(pos)*h^2+d(pos)*h^3;
    else
      ym(j)=99e99;
    end
end



function [a,b,c,d]=intspl3(x,y)
%
%  fonction d"interpolation
%     [a,b,c,d]=intspl3(x,y)
%     ou x et y sont des vecteurs.  Avec x(1)<x(2)<...<x(n);
%     a, b, c, d, sont des vecteurs de coefficients d"un polynome
%     f[x(i)] = a(i) + b(i)*(x-x(i)) + c(i)*(x-x(i))^2 + d(i)*(x-x(i))^3
%


[n,m]=size(x);, if m>1, x=x';, end

[n,m]=size(y);, if m>1, y=y';, end

[n,m]=size(x);

a=zeros(n-1,1);
b=zeros(n-1,1);
d=zeros(n-1,1);
c=zeros(n,1);
ct=zeros(n-2,1);

H=zeros(n-1,1);
T=zeros(n-2,n-2);
V=zeros(n-2,1);
T1=zeros(n-2,n-2);

%******** definition de la matrice T **********************

for j=1:n-1, H(j,1)=x(j+1)-x(j);, end
for j=1:n-2, V(j,1)=2/3*(H(j,1)+H(j+1,1));, end

T1=diag(V);
T=T+T1;
T1=diag(H(2:n-2)/3,-1);
T=T+T1;
T1=diag(H(2:n-2)/3,1);
T=T+T1;
clear T1;

%********* definition de la matrice QT **********************

QT=zeros(n-2,n);
```

```
Q=zeros(n,n-2);
Q1=zeros(n+2,n+2);

for j=1:n-2, V(j)=-(1/H(j)+1/H(j+1));, end

Q1=diag(H(1:n-2).^(-1),-2);
QT=QT+Q1(3:n,:);
Q1=diag(H(2:n-1).^(-1),2);
QT=QT+Q1(1:n-2,:);
QT(:,2:n-1)=QT(:,2:n-1)+diag(V);
clear Q1;

%********* definition des condions frontieres **************

con1=-(1/(x(2)-x(1))+1/(x(3)-x(1))+1/(x(4)-x(1)));
con2=1/(x(2)-x(1))*(x(3)-x(1))/(x(3)-x(2))*(x(4)-x(1))/(x(4)-x(2));
con3=1/(x(3)-x(1))*(x(2)-x(1))/(x(2)-x(3))*(x(4)-x(1))/(x(4)-x(3));
con4=1/(x(4)-x(1))*(x(2)-x(1))/(x(2)-x(4))*(x(3)-x(1))/(x(3)-x(4));
d0=con1*y(1)+con2*y(2)+con3*y(3)+con4*y(4);

con1=-(1/(x(n-1)-x(n))+1/(x(n-2)-x(n))+1/(x(n-3)-x(n)));
con2=1/(x(n-1)-x(n))*(x(n-2)-x(n))/(x(n-2)-x(n-1))*(x(n-3)-x(n))/(x(n-3)-x(n-1));
con3=1/(x(n-2)-x(n))*(x(n-1)-x(n))/(x(n-1)-x(n-2))*(x(n-3)-x(n))/(x(n-3)-x(n-2));
con4=1/(x(n-3)-x(n))*(x(n-1)-x(n))/(x(n-1)-x(n-3))*(x(n-2)-x(n))/(x(n-2)-x(n-3));
dn=con1*y(n)+con2*y(n-1)+con3*y(n-2)+con4*y(n-3);

V=zeros(n-2,1);
V(1)=d0/2;
V(n-2)=-dn/2;

%*************** definition de la matrice T *********

T(1,1)=H(1)/2+2*H(2)/3;
T(n-2,n-2)=2/3*H(n-2)+H(n-1)/2;

%*************** definition de la matrice Q **********

QT(1,1)=3/(2*H(1));
QT(1,2)=-(3/2/H(1)+1/H(2));
QT(n-2,n-1)=-(1/H(n-2)+3/2/H(n-1));
QT(n-2,n)=3/2/H(n-1);

V=V+QT*y;
ct=T\V;

a=y(1:n-1,1);
c(2:n-1,1)=ct;
c(1)=3/2*(y(2)-y(1))/H(1)^2-3*d0/(2*H(1))-c(2)/2;
c(n)=-3/2*(y(n)-y(n-1))/H(n-1)^2+3*dn/(2*H(n-1))-c(n-1)/2;

for j=1:n-1
   d(j)=(c(j+1)-c(j))/(3*H(j));
   b(j)=(y(j+1)-y(j))/H(j)-c(j)*H(j)-d(j)*H(j)^2;
end



function xm=newspl3(x,y,a,b,c,d,ym)
%
%fonction qui cherche xm lorsque ym est connue.
%Soit y(i) = a(i) + b(i)*( x-x(i) ) + c(i)*( x-x(i) )^2 + d(i)*( x-x(i) )^3
%Si ym est connue, xm est trouve par la methode de Newton-Raphson.
%
%  xm=newspl3(x,y,a,b,c,d,ym)
%
```

```
[n m]=size(ym);

if m>1, dim=m;, else dim=n;, end

[n m]=size(x);

if m>1, n=m;, end

for j=1:dim

  suivant=0;
  vec_p=find(y>=ym(j));
  if isempty(vec_p);
    disp(sprintf('ym n''est pas compris entre %g et %g',y(1),y(n)));
    suivant=1;
  else
    vec_p=find(y<=ym(j));
    if isempty(vec_p)
      disp(sprintf('ym n''est pas compris entre %g et %g',y(1),y(n)));
      suivant=1;
    else
      pente=y(2)-y(1);
      if pente>=0
        vec_p=find(y>ym(j));
        if isempty(vec_p)==0
          pos=vec_p(1)-1;
        else
          if y(length(y))==ym(j)
            pos=length(y)-1;
          end
        end
      else
        vec_p=find(y<ym(j));
        if isempty(vec_p)==0
          pos=vec_p(1)-1;
        else
          if y(length(y))==ym(j)
            pos=length(y)-1;
          end
        end
      end
    end
  end
  if suivant==0;

%  methode de Newton-Raphson

    xini=(x(pos+1)+x(pos))/2;
    sortir=1;
    conteur=0;
    while sortir
      h=xini-x(pos);
      df=b(pos)+2*c(pos)*h+3*d(pos)*h^2;
      f=a(pos)-ym(j)+b(pos)*h+c(pos)*h^2+d(pos)*h^3;
      delx=-f/df;
      xini=xini+delx;

      if conteur<5

        if xini>x(pos+1)
          xini=x(pos+1);
        end

        if xini<x(pos)
          xini=x(pos);
        end
```

```
      end
      conteur=conteur+1;
      if abs(delx)<1e-6
        sortir=0;
      end
      if conteur>10
        sortir=0;
      end
    end
    xm(j)=xini;

  else
    xm(j)=99e99;
  end
end


function [a,b,c,d]=smospl3(x,y,dy,S)
%
%  fonction spline de lissage utilisant les splines
%  naturelles pour effectuer le lissage.
%     entree:  x = vecteurs de base
%                 il faut que x(1)<x(2)<..<x(n)
%              y = vecteurs de coordonnes
%              dy = estime de la deviation standard
%              S = parametre de lissage
%
%      [a,b,c,d]=smospl3(x,y,dy,S)
%
%     sortie:  a,b,c,d   vecteurs de coefficients du polynome
%
%     u(i) = a + b*(x-x(i)) + c*(x-x(i))^2 + d*(x-x(i))^3
%     ou u est un estimateur.
%


%******  initialisation et transposition des vecteurs ******

[n,m]=size(x);, if m>1, x=x';, end

[n,m]=size(y);, if m>1, y=y';, end

[n,m]=size(x);

a=zeros(n,1);
b=zeros(n-1,1);
d=zeros(n-1,1);
c=zeros(n-1,1);
ct=zeros(n,1);

D=diag(dy*ones(n,1));
H=zeros(n-1,1);
T=zeros(n-2,n-2);
V=zeros(n-2,1);
T1=zeros(n-2,n-2);

%******** definition de la matrice T **********************

for j=1:n-1, H(j,1)=x(j+1)-x(j);, end
for j=1:n-2, V(j,1)=2/3*(H(j,1)+H(j+1,1));, end

T1=diag(V);
T=T+T1;
T1=diag(H(2:n-2)/3,-1);
T=T+T1;
```

```
T1=diag(H(2:n-2)/3,1);
T=T+T1;
clear T1;

%********* definition de la matrice QT **********************

QT=zeros(n-2,n);
Q=zeros(n,n-2);
Q1=zeros(n+2,n+2);

for j=1:n-2, V(j)=-(1/H(j)+1/H(j+1));, end

Q1=diag(H(1:n-2).^(-1),-2);
QT=QT+Q1(3:n,:);
Q1=diag(H(2:n-1).^(-1),2);
QT=QT+Q1(1:n-2,:);
QT(:,2:n-1)=QT(:,2:n-1)+diag(V);
clear Q1;
Q=QT';

M1=zeros(n-2,n-2);
M2=zeros(n-2,1);
M3=zeros(n-2,n-2);
M4=zeros(n,n);

%********* debut de l'algo **********************


p=0;

M1=QT*D*D*Q;
M2=QT*y;
M4=D*Q;

R=zeros(n-2,n-2);
u=zeros(n-2,1);
v=zeros(n,1);

sortir=1;

while  sortir==1
     M3=M1+p*T;
     R=chol(M3);
     u=M3\M2;
     v=M4*u;
     e=v'*v;
     if e>(S+.05*n)
       f=u'*T*u;
       w=R'\(T*u);
       g=w'*w;
       p=p+(e-sqrt(S*e))/(f-p*g);
      else
       sortir=0;
      end
end

ay=y-D*v;
[a,b,c,d]=intspl3(x,ay);


clear v u R M1 Q QT V T H D;
clear e f g w sortir n m j p;
clear x y ct M2 M3 M4;
```

```
function [a,b,c,d]=natspl3(x,y)
%
%  fonction spline naturelles
%     entree:  x = vecteurs de base
%                 il faut que x(1)<x(2)<..<x(n)
%              y = vecteurs de coordonnes
%
%      [a,b,c,d]=natspl3(x,y)
%
%     sortie:  a,b,c,d   vecteurs de coefficients du polynome
%
%     u(i) = a + b*(x-x(i)) + c*(x-x(i))^2 + d*(x-x(i))^3
%


%******  initialisation et transposition des vecteurs ******

[n,m]=size(x);, if m>1, x=x';, end

[n,m]=size(y);, if m>1, y=y';, end

[n,m]=size(x);

a=zeros(n,1);
b=zeros(n-1,1);
d=zeros(n-1,1);
c=zeros(n,1);
ct=zeros(n,1);

H=zeros(n-1,1);
T=zeros(n-2,n-2);
V=zeros(n-2,1);
T1=zeros(n-2,n-2);

%******** definition de la matrice T **********************

for j=1:n-1, H(j,1)=x(j+1)-x(j);, end
for j=1:n-2, V(j,1)=2/3*(H(j,1)+H(j+1,1));, end

T1=diag(V);
T=T+T1;
T1=diag(H(2:n-2)/3,-1);
T=T+T1;
T1=diag(H(2:n-2)/3,1);
T=T+T1;
clear T1;

%********* definition de la matrice QT ***********************

QT=zeros(n-2,n);
Q=zeros(n,n-2);
Q1=zeros(n+2,n+2);

for j=1:n-2, V(j)=-(1/H(j)+1/H(j+1));, end

Q1=diag(H(1:n-2).^(-1),-2);
QT=QT+Q1(3:n,:);
Q1=diag(H(2:n-1).^(-1),2);
QT=QT+Q1(1:n-2,:);
QT(:,2:n-1)=QT(:,2:n-1)+diag(V);
clear Q1;
Q=QT';

a=y;
c(2:n-1,1)=T\(QT*a);
```

```
for j=1:n-1

   d(j)=(ct(j+1)-ct(j))/(3*H(j));
   b(j)=(a(j+1)-a(j))/H(j)-ct(j)*H(j)-d(j)*H(j)^2;
end
```

ANNEXE B

DONNÉES RÉELLES

Les valeurs présentées sont pour les données réelles d'un capteur de précision.

	15	16	17	18	19	20	21	← température
0	5.0702	4.8966	3.8663	2.9241	1.9947	1.3295	0.8344	
3	8.6306	8.4475	8.0534	7.4807	6.4826	5.6455	4.8003	
6	7.6202	7.8726	8.3256	8.8574	8.8034	9.1207	8.6472	
9	3.4218	4.2774	5.1807	5.9548	6.8969	7.6851	8.3684	
12	0.0674	0.1929	0.4929	1.0225	1.7883	2.7690	3.6811	
15	1.8591	1.2881	0.7331	0.3277	0.0809	0.1011	0.1700	
18	6.7455	6.0951	5.3434	4.2396	3.0571	2.1245	1.6008	
21	9.4052	9.3077	8.8907	8.3673	7.6148	6.8392	6.0059	
24	6.8488	7.2748	7.8867	8.4047	8.8159	8.7866	8.4066	
27	2.3054	2.8617	3.3810	4.2801	5.4863	6.2897	6.8187	
30	0.0512	0.0926	0.2659	0.5182	1.1511	1.8379	2.5222	

↑
tension

	22	23	24	25	26	27	28	← température
0	0.3725	0.1282	0.1116	0.1523	0.4142	0.6837	1.4237	
3	3.9948	3.1180	2.5303	1.7870	1.2417	0.7360	0.2560	
6	8.4296	7.6835	7.2843	6.7663	5.8292	5.0250	4.1363	
9	8.7829	9.3082	9.4634	9.6138	9.2832	8.8864	8.3849	
12	4.5449	5.4465	6.3470	7.0912	7.6962	8.3041	8.9941	
15	0.4656	1.0345	1.7205	2.5464	3.2214	4.0288	4.9655	
18	0.9053	0.4093	0.1526	0.0356	0.0960	0.2604	0.7484	
21	5.2502	4.3654	3.3562	2.4632	1.6558	1.0917	0.5437	
24	8.1622	7.6370	6.9773	6.1588	5.5269	4.9327	3.8572	
27	7.4899	7.9392	8.1062	8.3806	8.2708	8.0760	7.6680	
30	3.3627	4.6130	5.6074	6.0828	7.0944	8.2400	8.7527	

↑
tension

	29	30	31	32	33	34	35	← température
0	1.9278	2.6540	3.7148	4.6014	5.7344	6.3963	7.3108	
3	0.0900	0.0918	0.2342	0.4584	0.9491	1.6743	2.3472	
6	3.3239	2.4952	1.7726	1.1557	0.6305	0.1966	0.0514	
9	7.6412	6.9252	6.1494	5.4269	4.3537	3.4691	2.7128	
12	9.1777	9.3221	9.3224	9.1805	8.3887	8.1547	7.3415	
15	6.0906	6.8690	7.6100	7.9632	8.7149	9.0465	9.3885	
18	1.3412	1.8004	2.5298	3.6191	4.5548	5.3703	6.0219	
21	0.2109	0.0471	0.0850	0.2841	0.5291	0.9501	1.5751	
24	3.0750	2.3013	1.6267	1.0228	0.5471	0.2494	0.0542	
27	7.2555	6.6814	5.8987	5.2462	4.6564	3.6892	2.7386	
30	9.2441	9.5198	9.4151	9.5298	9.1242	8.6161	7.9535	

↑
tension