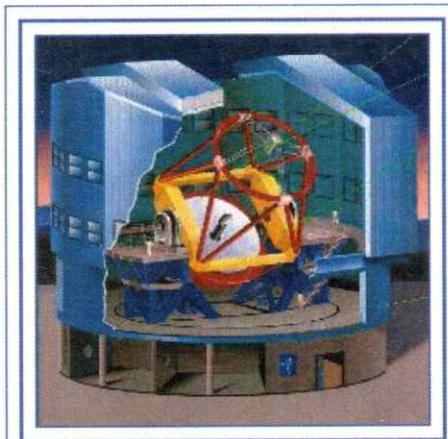
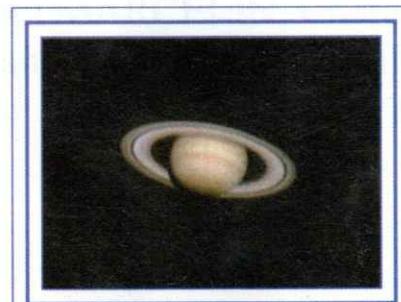
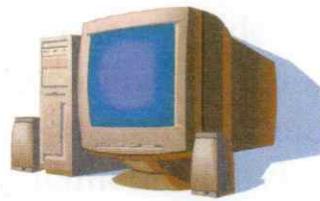


« Validation en langage C et VBA
d'un modèle donnant la relation
Courbure Pression pour les miroirs à
courbure variable. »

Stage de développeur informatique
AFPA d'Istres mai 2004 avril 2005
Jérôme FRASSON

Application effectuée au
Laboratoire d'optique de
l'observatoire de Marseille, unité du
CNRS

Du 1^{er} février au 8 avril 2005
sous la responsabilité de
Silvio MAZZANTI



Remerciements

Je remercie tout particulièrement mon maître de stage Silvio MAZZANTI pour son aide et sa patience exemplaire.

Merci également au personnel du laboratoire d'Optique pour toutes les explications techniques fournies.

Une attention particulière à Olivier BOISSIN, Technicien de la recherche à l'observatoire.

Merci enfin à Michel MARCELIN, Chargé de recherche, à Georges COMTE sans qui ce stage n'aurait pas été possible.

Merci à Christine pour le chaleureux accueil et la qualité de l'hébergement.

La structure

Le LAM résulte de la fusion en une Unité Mixte de Recherche (U.M.R. n°6110) CNRS-Université de Provence de deux laboratoires, anciennement Observatoire de Marseille et Laboratoire d'Astronomie Spatiale. Cette UMR est rattachée à l'observatoire des sciences de l'Univers Marseille Provence.

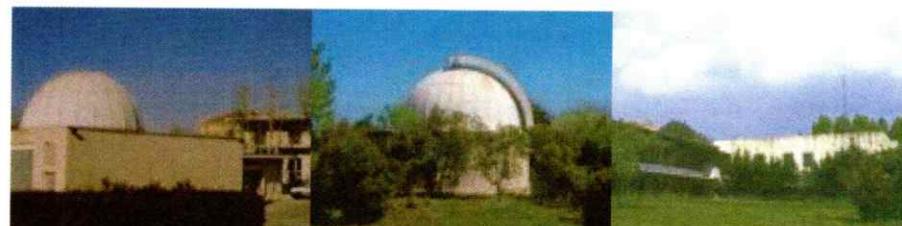
La nouvelle unité constitue l'un des plus gros institut de recherche publique de la région marseillaise et associe la recherche fondamentale en astronomie et la recherche technologique en instrumentation.

Le LAM héberge 45 chercheurs-100 ingénieurs,techniciens,administratifs-12 thésards et environ 20-25 stagiaires par an, qui en attendant un regroupement géographique sur le Technopole de Chateau-Gombert, sont répartis sur 2 sites:

le Site Peiresc-les Olives (ex: Laboratoire d'Astronomie Spatiale)



le Site Le Verrier-Longchamp (ex: Observatoire de Marseille)



L'institut travaille sur des projets, nationaux et internationaux dont il a souvent la responsabilité. Ces projets se font en collaboration avec des consortiums constitués par des laboratoires et des entreprises internationaux, pour le compte d'agences nationales ou internationales (CNES,NASA,ESO...) Pour les projets qui le demandent, les méthodes de travail s'approchent de celle de l'industrie spatiale (gestion de projet, suivi de documentation, assurance qualité ...).S'appuyant sur de solides bureaux d'études et ateliers, sur un service essais répondant aux spécifications des agences spatiales (salles blanches, hottes à flux laminaires, enceinte à vide, excitateur de vibration), et sur un secteur optique. important (R et D, polissage, intégration, contrôle).

Sommaire

Spécification

Présentation d'ensemble (problématique)	1
Système informatique embarqué	3
analyseur	5
Système informatique global	6
Calculs d'hystérèse	9
Rôle des applications	11

Conception

Usage Linux langage C	13
Usage Windows XP VBA	17
Méthodologie développement VBA	19

Programmation

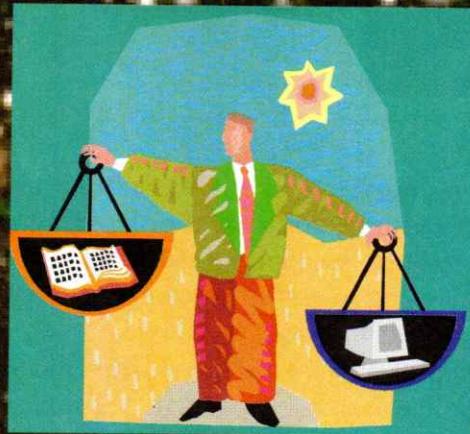
Environnement Linux	27
Environnement Windows et annexe	31





9

Spécification



L'observatoire astronomique VLTI (Very Large Telescope Interferometry), situé au Chili sur le mont Paranal à 2700m d'altitude, est géré par l'ESO (European Southern Observatory ou Observatoire Européen Austral). Il s'agit du plus grand observatoire au monde.

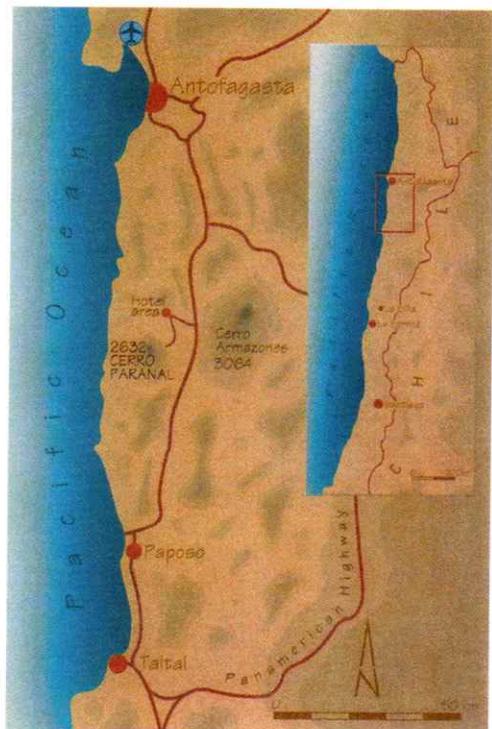
Il possède quatre principaux télescopes de 8,20m de diamètre (taille du miroir collecteur de la lumière des étoiles), chacun peut fonctionner indépendamment mais le but est de combiner les quatre faisceaux pour créer un télescope théorique équivalent de très grand diamètre, comme s'il possédait un miroir de plusieurs dizaines de mètres de diamètre, cette technologie s'appelle l'interférométrie.



The VLT Array on the Paranal Mountain

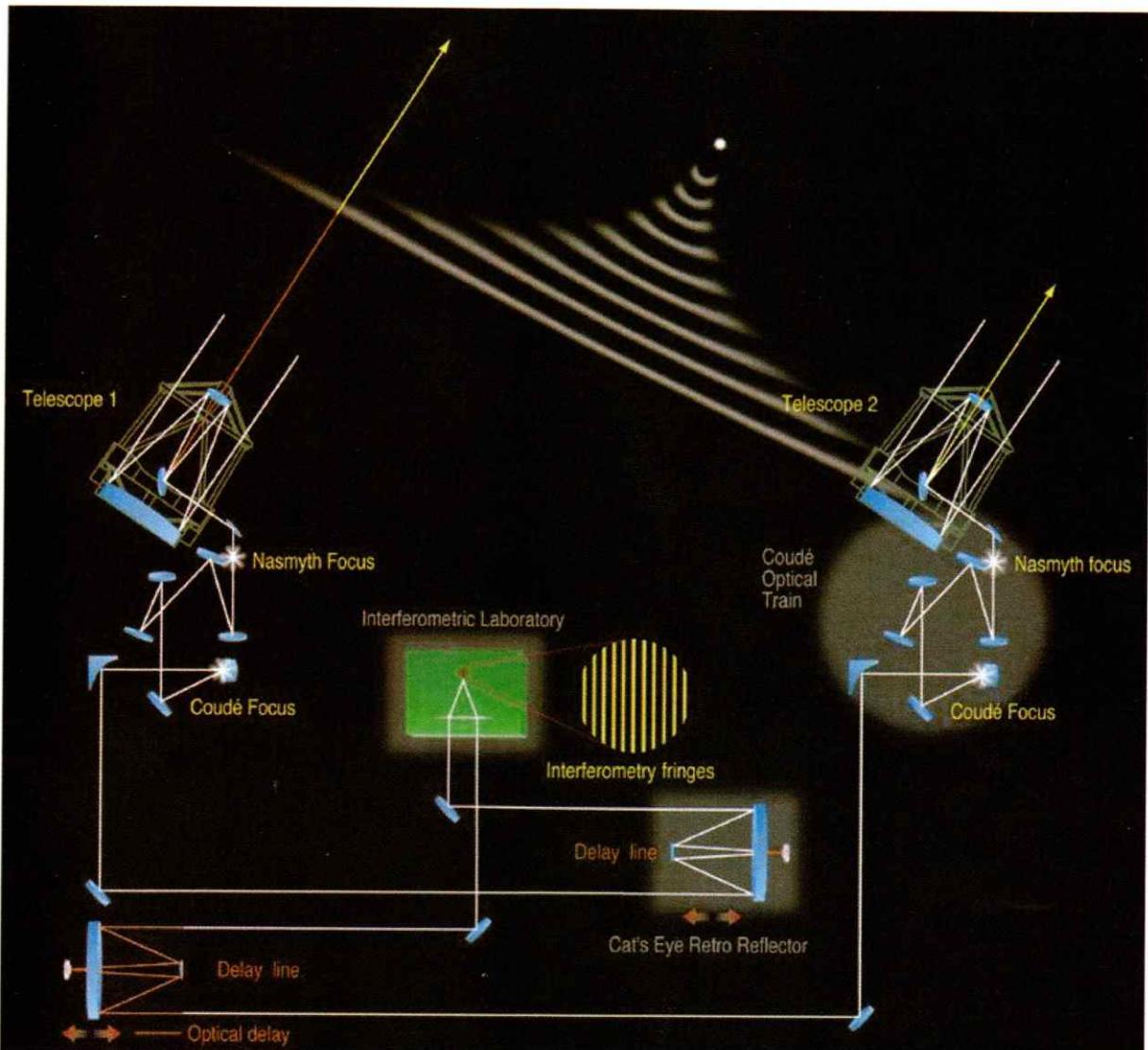
ESO PR Photo 11a/00 (24 May 2000)

© European Southern Observatory

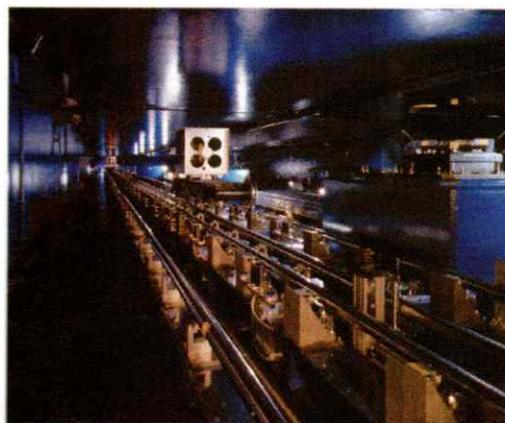


L'inconvénient technique est qu'il est nécessaire de synchroniser parfaitement les longueurs des faisceaux transmis en temps réel selon la direction pointée par les télescopes, en effet il apparaît une différence dans le train d'ondes lumineuses captées. Plus l'objet visé est bas sur l'horizon et plus cette différence est prononcée. Elle dépend également de la longueur d'onde à laquelle on observe.

Egalement il existe une variation à corriger au fur et à mesure que l'objet pointée varie au cours de la nuit (rotation de la Terre).

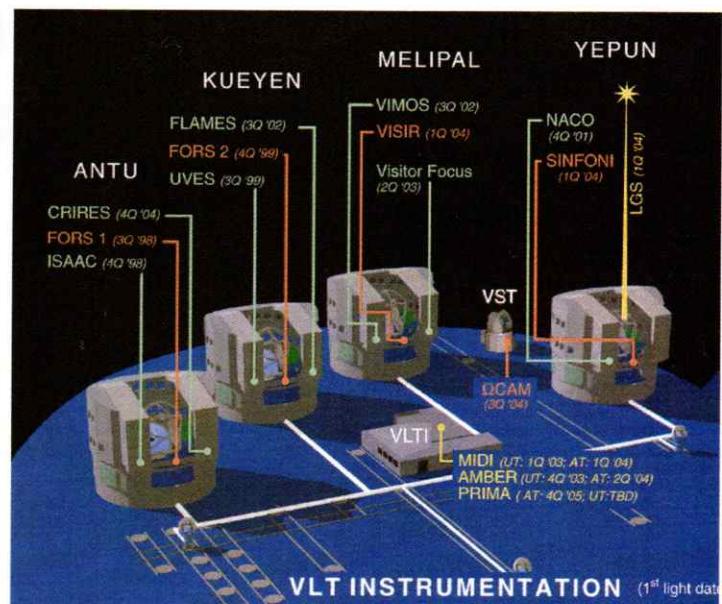


Cette synchronisation est faite de manière optique : les flux lumineux collectés par chaque télescope aboutissent après une série de réflexions successives dans des couloirs d'un longueur de 65 m.



The VLT Interferometric Tunnel with Delay-Lines

ESO PR Photo 22a/02 (29 September 2002)



VLT INSTRUMENTATION (1st light data)

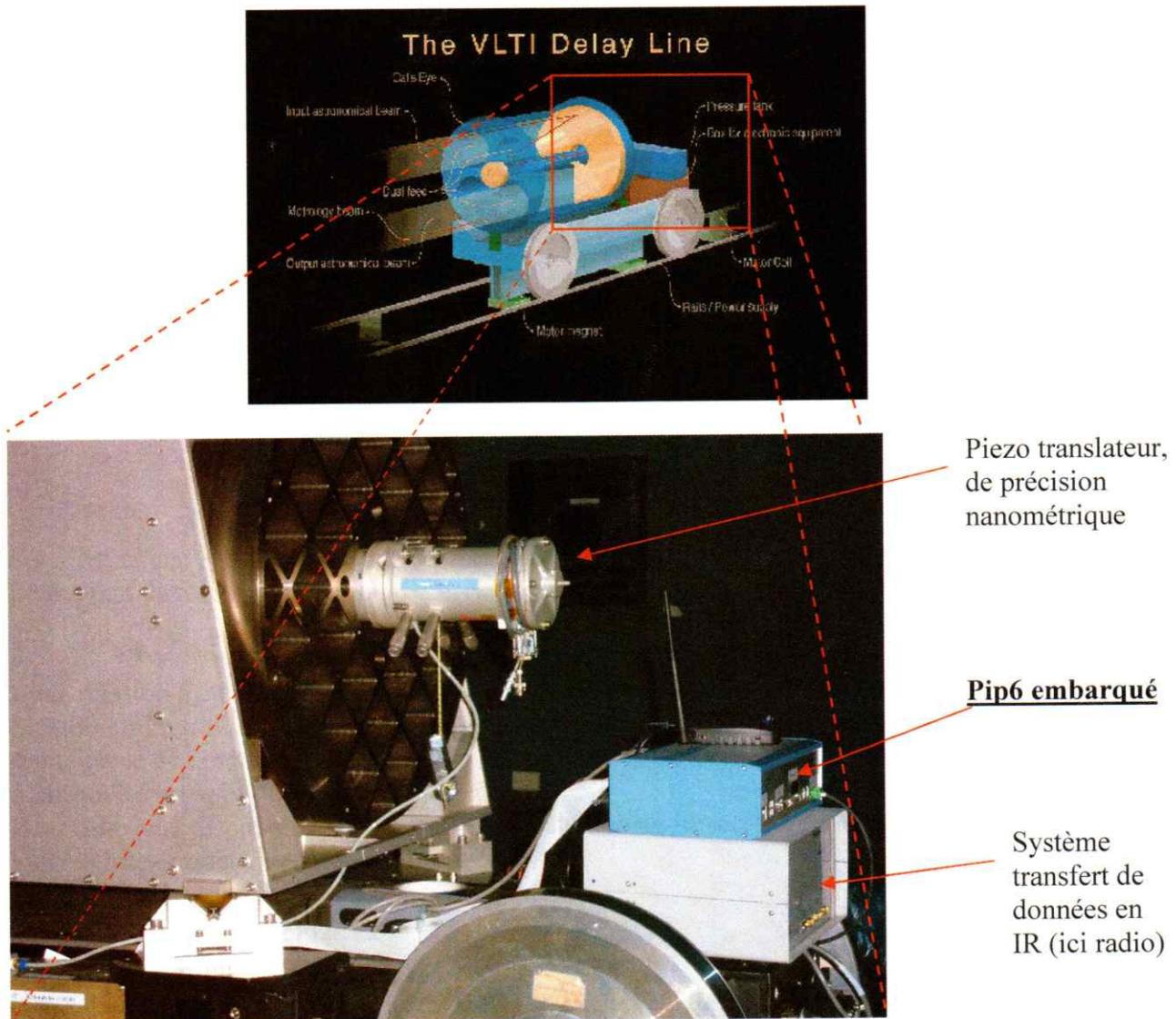
Dans chaque couloir se trouvent des chariots mobiles, ceux-ci sont porteurs de systèmes embarqués :

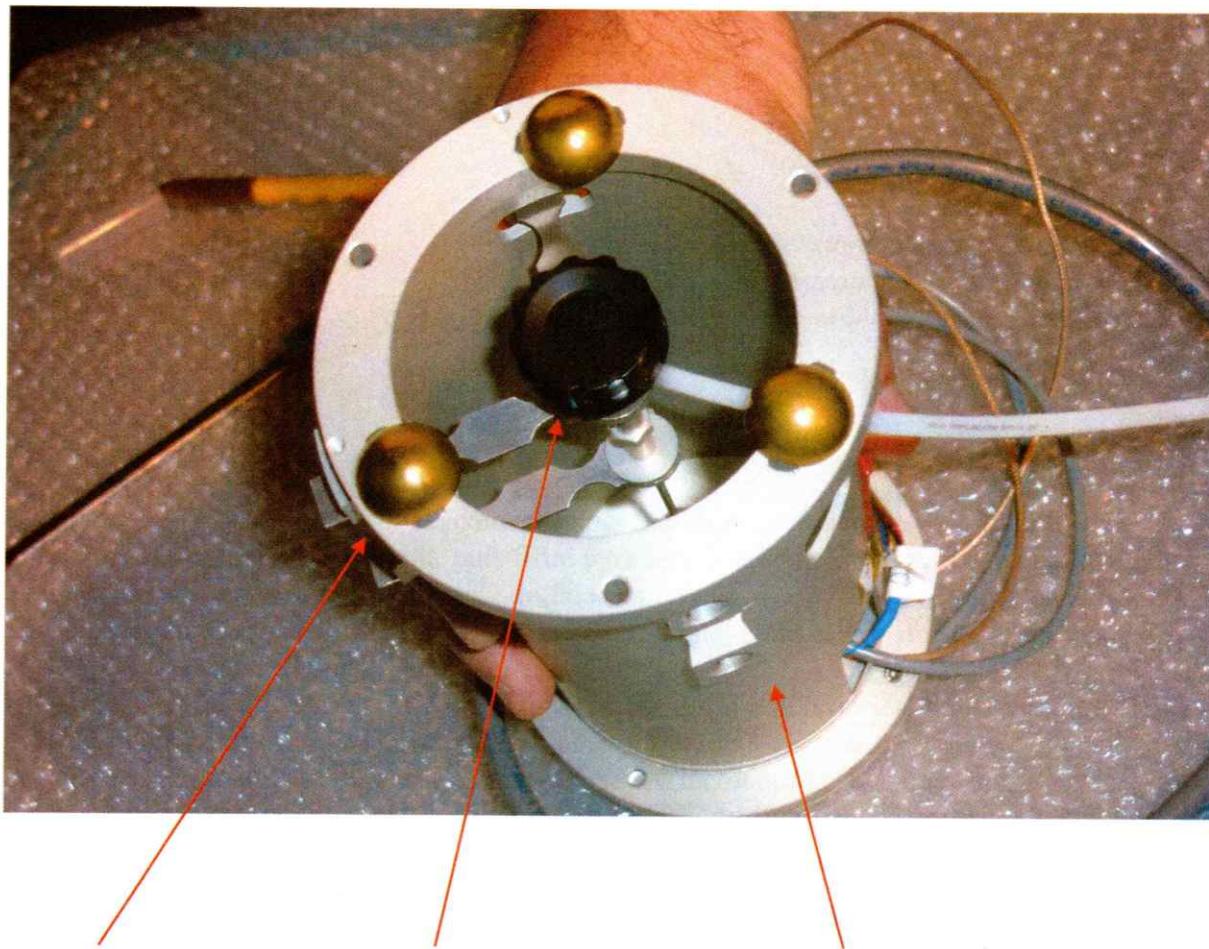
- Un miroir zoom à courbure variable, piloté par...
- ...un **système informatique temps réel** dialoguant avec...
- ...une station fixe grâce à un système de transfert d'information par infrarouge.

Le but de chaque ensemble chariot-miroir-calculateur embarqué, est de faire varier la courbure du miroir en fonction de la position relative du chariot porteur le long de la ligne à retard.

Le miroir varie en courbure pour maintenir une mise au point en un lieu fixe (capteurs fixes). Le déplacement du chariot permet de maintenir les longueurs des trajets lumineux.

Il était donc nécessaire de concevoir un miroir ayant les caractéristiques nécessaires, et de déterminer les équations faisant correspondre la pression exercée sur l'une des parois du miroir et la variation de courbure de celui-ci.





Dispositif de réglage manuel à trois branches

VCM ou miroir à Courbure variable

Carter de l'ensemble Miroir-Piezotranslateur

Par ailleurs une fois les équations établies il fallait trouver un programme informatique permettant de tester les équations, afin de vérifier les résultats obtenus expérimentalement.

Cette mission a été confiée au laboratoire d'optique de l'observatoire de Marseille. Le laboratoire travailla donc sur la propriété que possède une fine lame d'acier de la taille d'une pièce de monnaie jouant le rôle d'un miroir, de se déformer en fonction de la pression d'air qui lui est appliquée.

Pour cela deux dispositifs expérimentaux ont été utilisés, il serviront à vérifier la validité du modèle numérique. Ces dispositifs sont :

- L'Analyseur de Shack-Hartmann.
- L'Analyseur de Fizeau.

Notons que les mesures fournies par chaque procédé sont indépendantes.

Nous allons expliquer le principe de l'Analyseur de Shack-Hartmann :



*Banc d'essai du laboratoire
d'optique.*

*La calibration
pression/courbure de chaque
miroir y est effectuée de
manière expérimentale.*

Un faisceau laser Hélium-néon est concentré par une minuscule lentille (type lentille de microscope) qui le rend divergent, puis filtré à travers une petite ouverture dans une plaque, ne laissant passer que la partie centrale du faisceau.

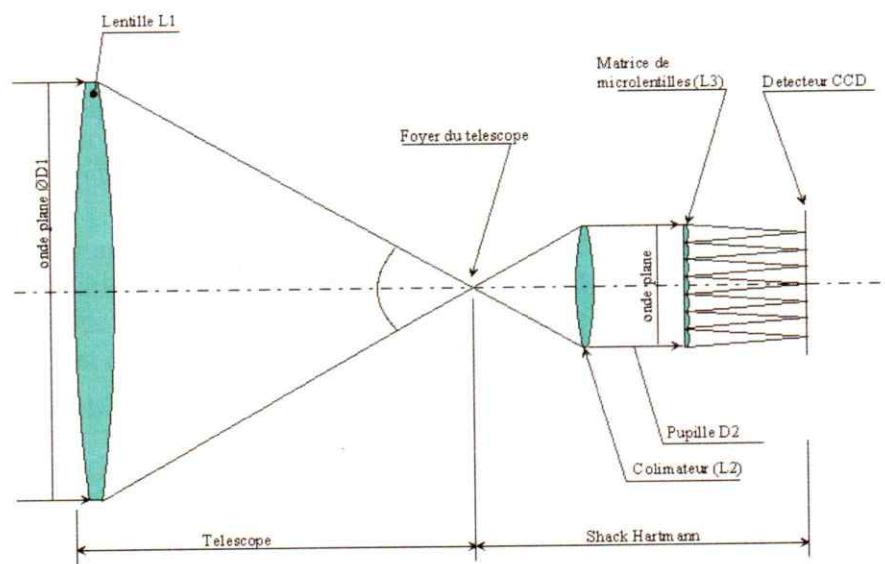
Sur le trajet du faisceau se situe une lentille convergente, celle-ci transforme le faisceau divergent en un faisceau parallèle.

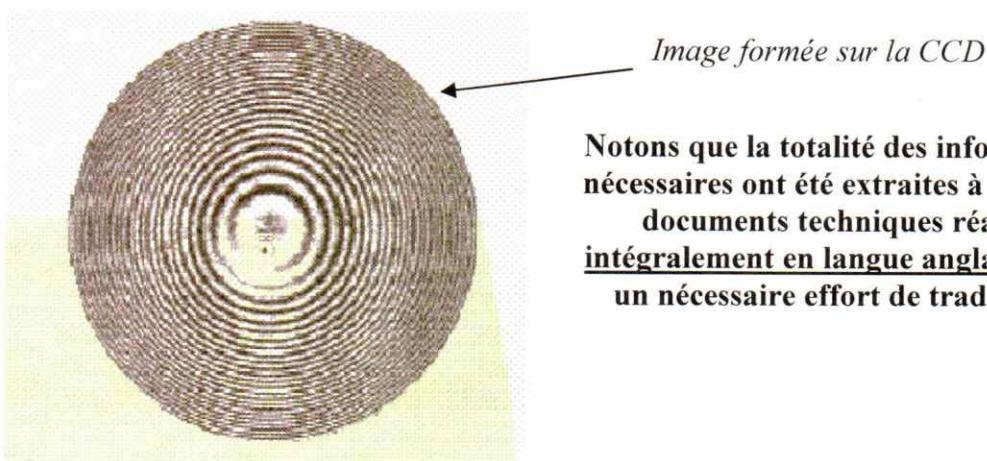
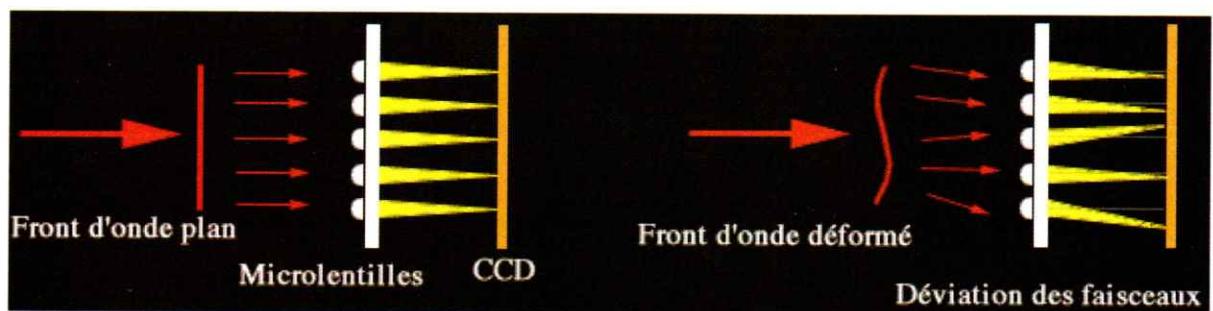
Le faisceau en question traverse ensuite un séparateur qui a la propriété de transmettre 50% de la lumière incidente sans réflexion, et 50% avec réflexion à angle droit.

Le faisceau lumineux atteint ensuite le miroir, est réfléchi sur la surface et repart en sens inverse vers le séparateur, qui réfléchi à nouveau 50% du flux vers un capteur CCD (charge couple device ou dispositif à couplage de charge), ce capteur est précédé d'un couche formée de microlentilles parallèles déviant plus ou moins chaque rayon en fonction de l'angle formé par le front d'onde (le front d'onde nous permet justement de déduire alors la courbure du miroir).

On applique différentes pression sur une face du miroir (50 mesures) et on regarde à chaque fois quelle est la courbure correspondante.

Voici un schéma expliquant le dispositif de l'analyseur :





Notons que la totalité des informations nécessaires ont été extraites à partir de documents techniques réalisés intégralement en langue anglaise, d'où un nécessaire effort de traduction.

Sur site, le principe de fonctionnement du système embarqué est le suivant : Sur chaque chariot se trouve un système possédant son propre microprocesseur FEP/PIP6 (Front End Processor/de type PIP6, un processeur de type CMOS), chaque processeur est contrôlé par une unité locale de contrôle

LCU (Local Control Unit) fixe via une liaison infrarouge RS232.

L'unité locale de contrôle est un système temps réel appelé VxWorks, elle reçoit elle-même des instructions venant d'autres stations UNIX.

L'unité locale de contrôle reçoit un ordre de courbure à afficher en provenance de la station de travail principale de la ligne à retard, cet ordre de courbure dépend de la position occupée par le chariot.

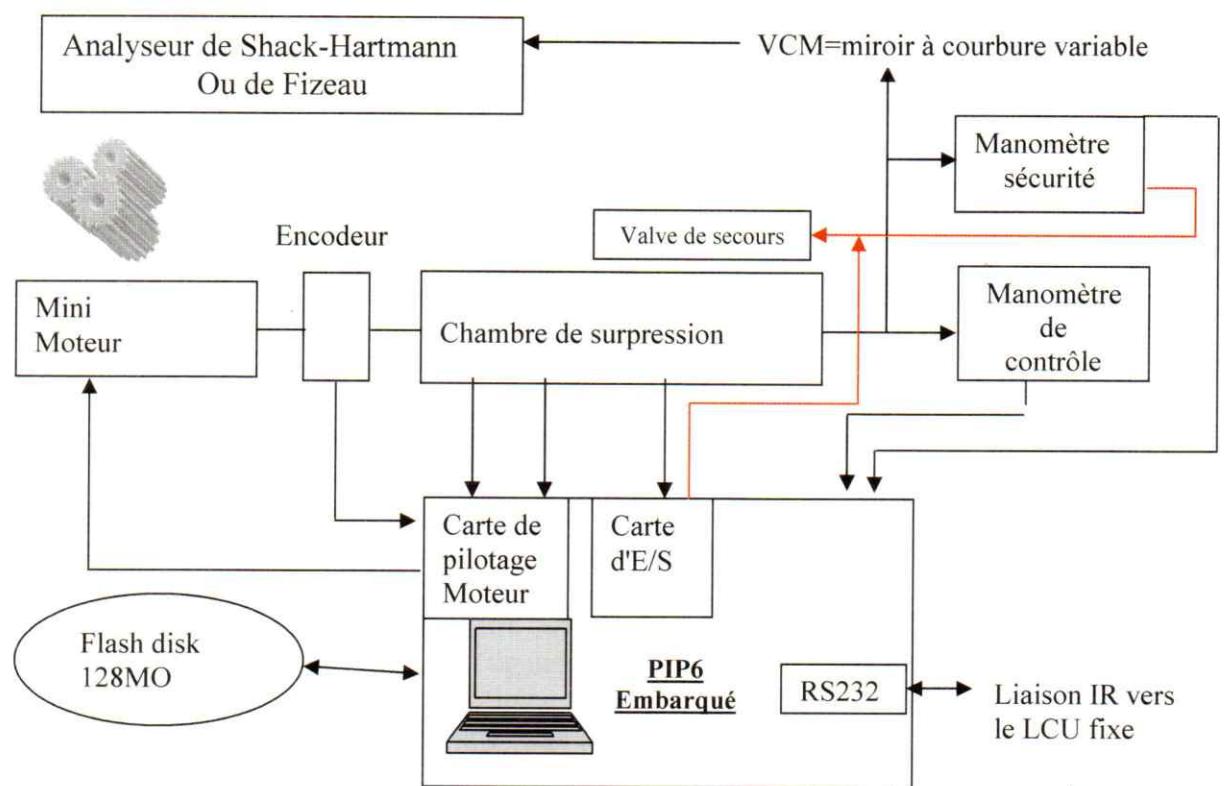
Elle transforme cette courbure en une valeur correspondante de la pression qui sera transmis à un système embarqué.

L'unité locale de contrôle commande 8 lignes à retard, et calcule pour chacune d'entre elles la pression correspondante.

Le système embarqué sur le chariot possède sa propre mémoire flash de 128mo, équipée d'un système d'exploitation Linux et de l'application servant au pilotage. Sa fonction est de recevoir un ordre de pression à afficher en provenance de l'unité locale de contrôle une à dix fois par seconde, et de maintenir la pression demander dans une marge de précision acceptable.

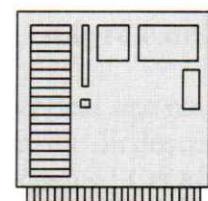
Les programmes embarqués sont développés sous la version Debian de LINUX, puis transférés sur la mémoire flash ou est préalablement installée une version allégée du système d'exploitation. Ceci permet une faible consommation d'énergie et l'absence de vibrations.

Du point de vue matériel, le système embarqué fonctionne de la manière suivante :



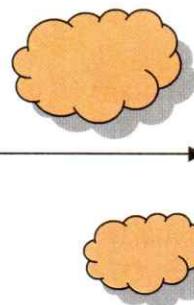
Système Embarqué

Système fixe

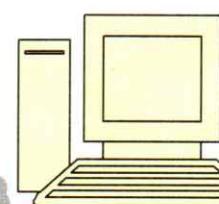


Station de travail principale

Ordre de courbure



Ordre de pression



Les calculs
courbure → pression
sont effectués ici...

Le système reçoit d'abord un ordre de pression à afficher en provenance de l'unité locale de contrôle notée LCU sur le schéma, par une liaison infrarouge.

Grâce au mini système d'exploitation UNIX et à l'application installés en mémoire flash, le processeur PIP6 utilise la carte de pilotage du moteur pour envoyer des instructions à celui-ci. Il s'agit d'un moteur à courant continu équipé d'un réducteur relié à un encodeur qui enregistre avec précision chaque mouvement et renvoie ces informations à la carte de pilotage. Ceci permet un contrôle en temps réel.

Le mini moteur actionne donc la course du cylindre d'une chambre de suppression d'air, qui communique la pression désirée au miroir déformant via un flexible, à ce stade un manomètre de contrôle renvoie vers le processeur la valeur de la pression appliquée au miroir.

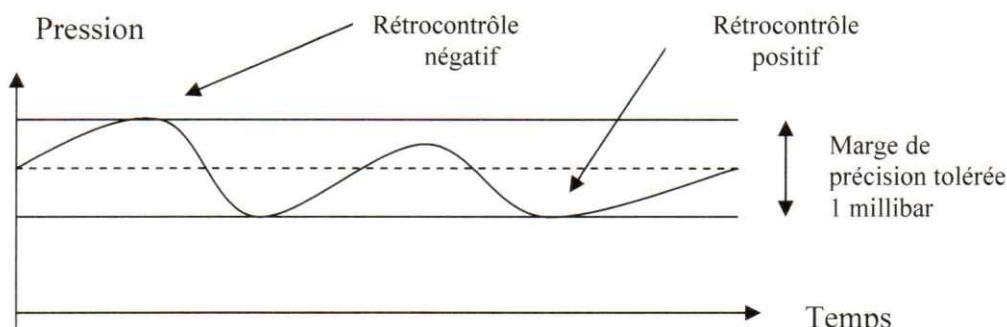
Il s'agit donc d'un système fonctionnant localement en boucle de régulation.

Si la pression dépasse accidentellement la limite de structure du miroir, le manomètre de sécurité déclenche l'ouverture de la valve de secours, ce qui purge la chambre de surpression et fait chuter immédiatement la pression appliquée au miroir.

La valve de secours peut également être ouverte directement par le processeur.

Des capteurs de position enregistrent également en temps réel le passage du piston en certains points.

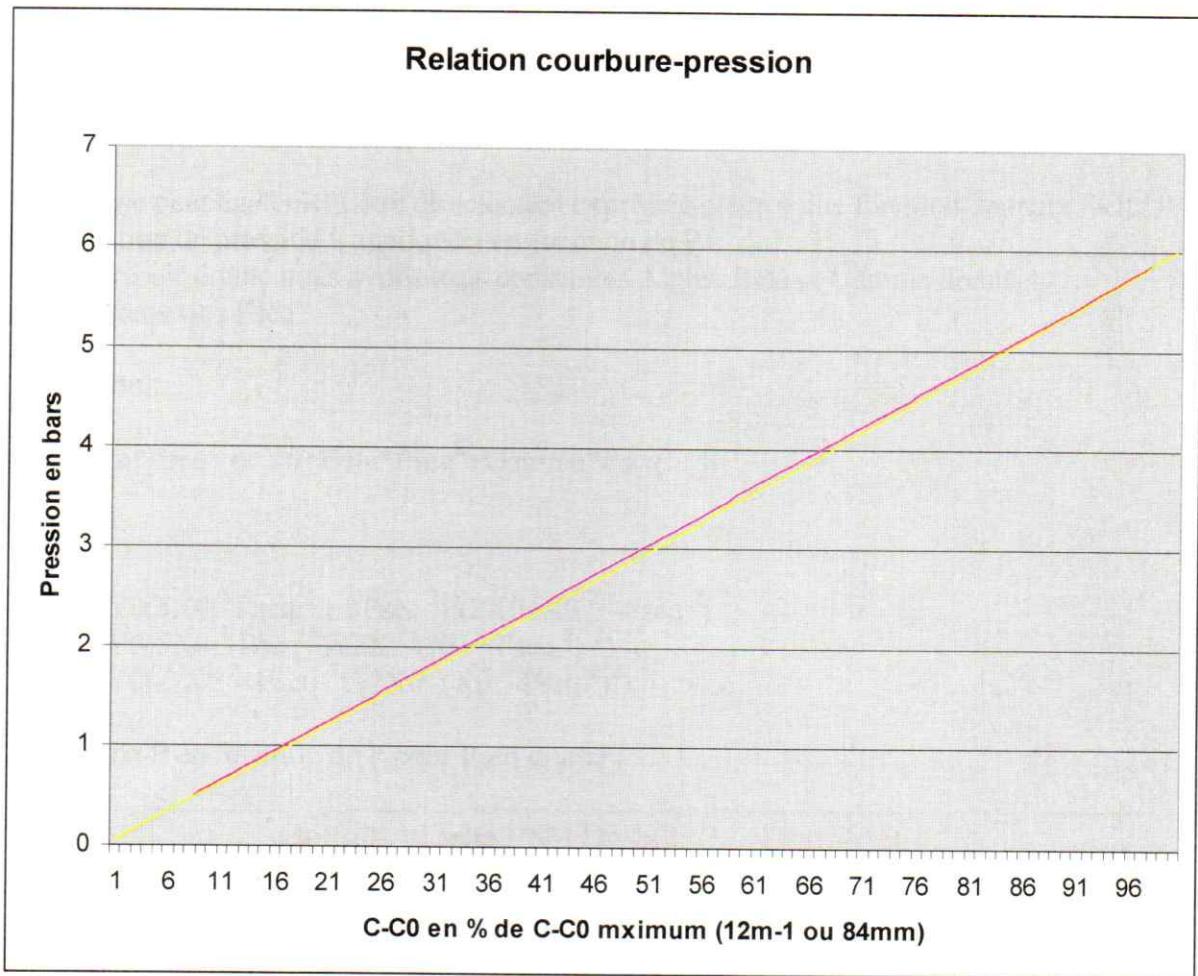
Les entrées/sorties sont gérées par une carte notée carte d'E/S.



Le schéma ci-dessus nous montre le principe de la boucle de régulation de la pression, ceci est le rôle de l'application embarquée.

Cependant une nouvelle difficulté devait se présenter : selon que l'on était en pression montante ou descendante, pour une valeur donnée de la pression les courbures n'étaient pas les mêmes ! Cet effet différait en fonction du type de miroir et de la pression maximale atteinte lors de la montée.

Ce phénomène est dénommé **Hystérose**.



Le but du programme qui va être présenté est justement de valider le modèle numérique permettant la détermination de l'hystérèse qui existe entre les courbes de pression montantes et descendantes dans le graphique pression/courbure des miroirs à courbure variable.

Le calcul de l'hystérèse s'effectue uniquement dans le cas d'une pression descendante, il est utilisé pour le pilotage en temps réel de la courbure des miroirs.

En pression montante, l'équation de la courbe est la suivante :

$$P = A_1(C - C_0) + A_3(C - C_0)^3 + A_5(C - C_0)^5$$

Où C est la courbure du miroir, C₀ la courbure au repos, P la pression appliquée et A₁, A₃ et A₅ des constantes caractéristiques d'un miroir donné.

Appelons P_w la pression maximale à ne pas dépasser pour un miroir donné, il existe une pression limite notée P_a entraînant le déclenchement d'une alarme.

Soit P_{sec} la pression maximale atteinte dans la séquence de montée en pression.
Nous devons toujours avoir :

$$0 < P_{\text{seq}} < P_a < P_w$$

L'hystérèse sera maximale à la diminution de la pression si $P_{seq}=P_w$, l'équation correspondante est également un polynôme de degré 5.
En revanche si $P_{seq} < 2$ bars l'hystérèse est négligeable.

L'hystérèse peut également être directement exprimée grâce à une fonction donnant Delta P (la correction de pression à appliquer) en fonction de P.
Pour un miroir donné nous avons trois constantes Alpha, Beta et Gamma données.
Nous connaissons P_{seq} .

On en déduit :

$$X_0 = \text{Alpha} * P_{seq} \quad \text{et} \quad Y_0 = \text{Beta} * P_{seq}^2 + \text{Gamma} * P_{seq}^4$$

Puis :

$$\begin{aligned} \Delta 1 &= Y_0(5X_0^2 P_{seq}^2 + 3P_{sec}^2)/(2X_0(X_0^2 - P_{seq}^2)^2) \\ \Delta 3 &= Y_0(5X_0^4 - P_{seq}^4)/(2X_0^3(X_0^2 - P_{seq}^2)^2) \\ \Delta 5 &= Y_0(-3X_0^2 + P_{seq}^2)/(2X_0^3(X_0^2 - P_{seq}^2)^2) \end{aligned}$$

Enfin Delta P en fonction de P pour P_{seq} donné :

$$\Delta P = \Delta 1 * P + \Delta 3 * P^3 + \Delta 5 * P^5$$

En résumé, chaque miroir possède les paramètres suivants :

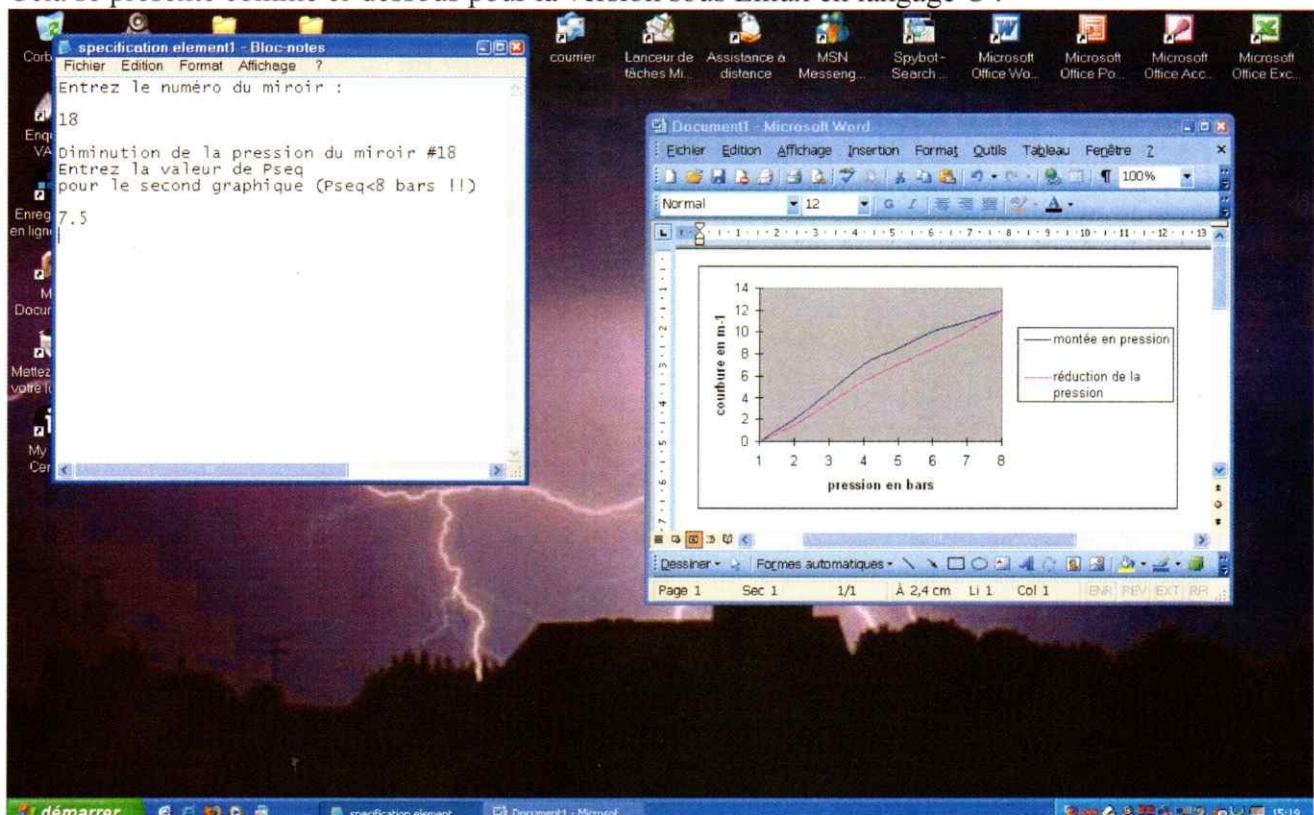
1	#	Numéro du miroir
2	P_w	Pression maximale supportable par le miroir
3	P_a	Pression de déclenchement de l'alarme
4	A_1	Premier coefficient de l'équation de montée
5	A_3	Second coefficient de l'équation de montée
6	A_5	Troisième coefficient de l'équation de montée
7	C_0	Courbure du miroir au repos
8	Alpha	Coefficient de calcul de X_0
9	Beta	Premier coefficient de calcul de Y_0
10	Gamma	Second coefficient de calcul de Y_0
11	...	Réservé pour un futur paramètre si nécessaire

Dans un premier temps, l'utilisateur souhaite entrer le numéro du miroir, puis la valeur de la pression maximale atteinte, dénommée P_{seq} .

Le programme affiche alors le graphe Pression/courbure à la montée en pression et à la descente en pression, la courbure est en abscisse et la pression en ordonnée.

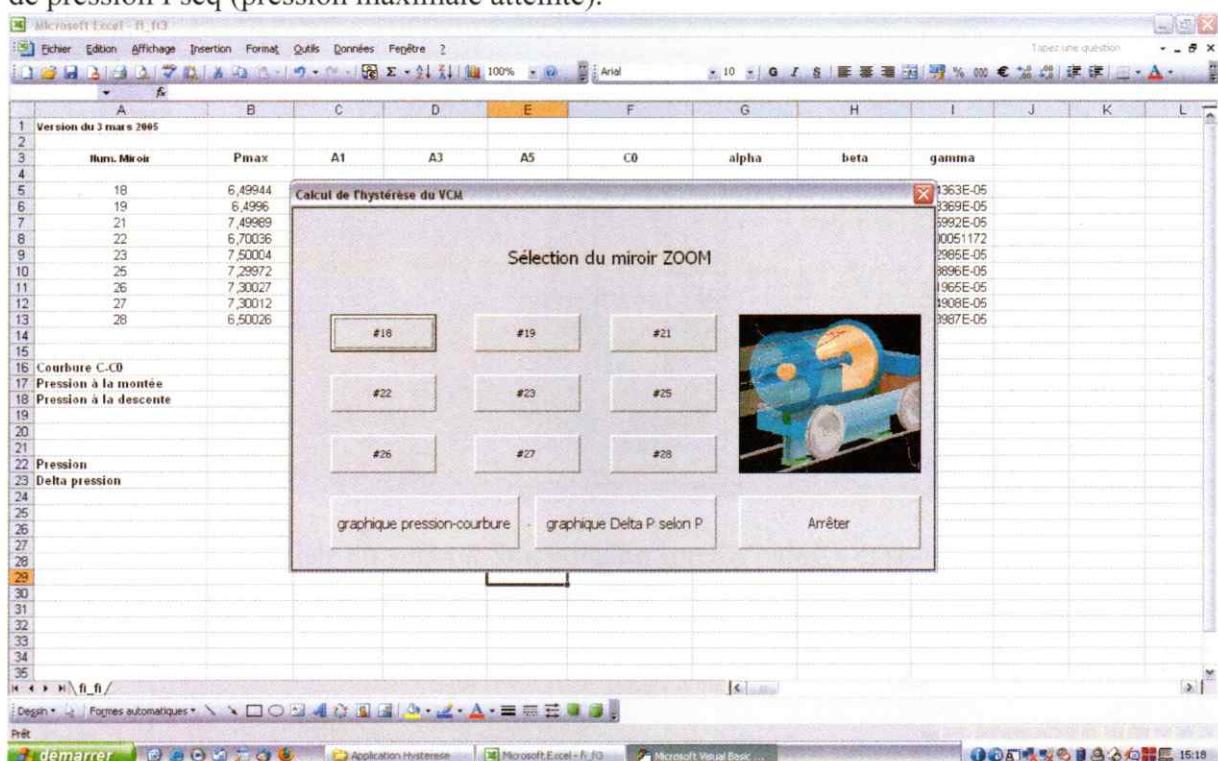
A chaque étape des calculs le programme affiche chaque valeur de l'ensemble des variables : Pression, courbure, valeurs calculées pour les coefficients des équations.

Cela se présente comme ci-dessous pour la version sous Linux en langage C :

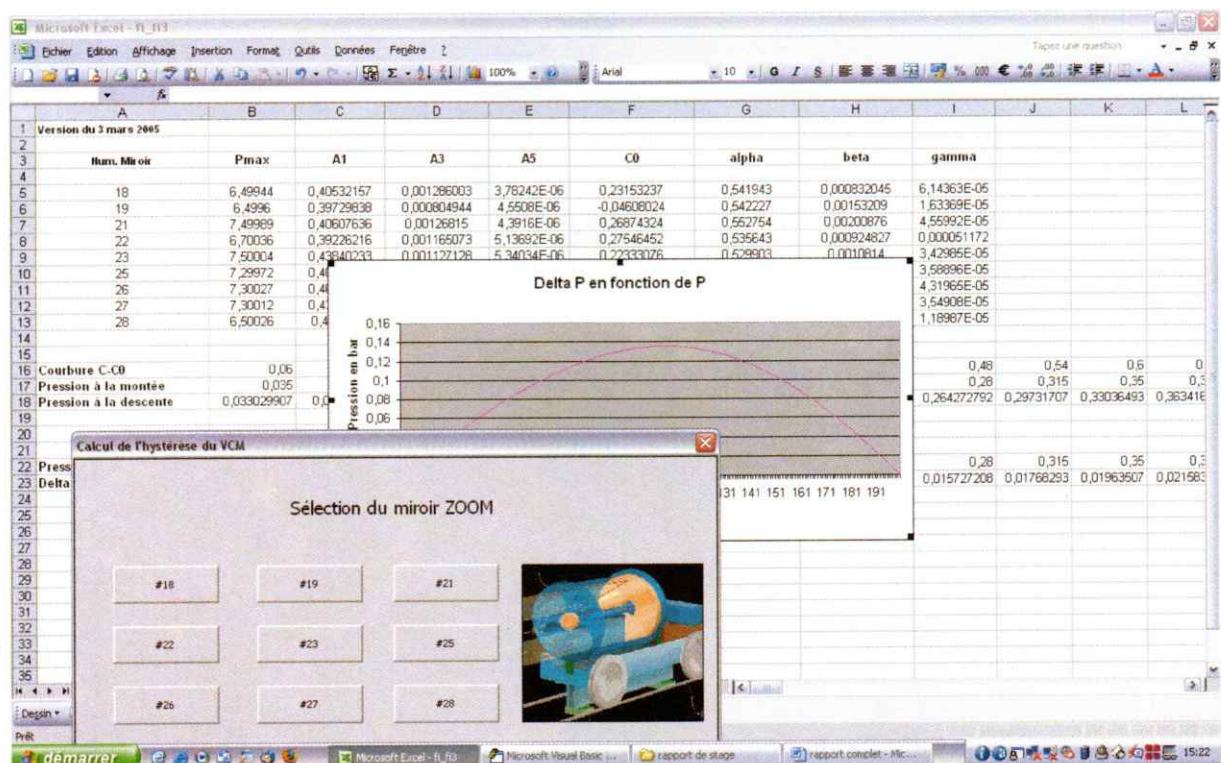
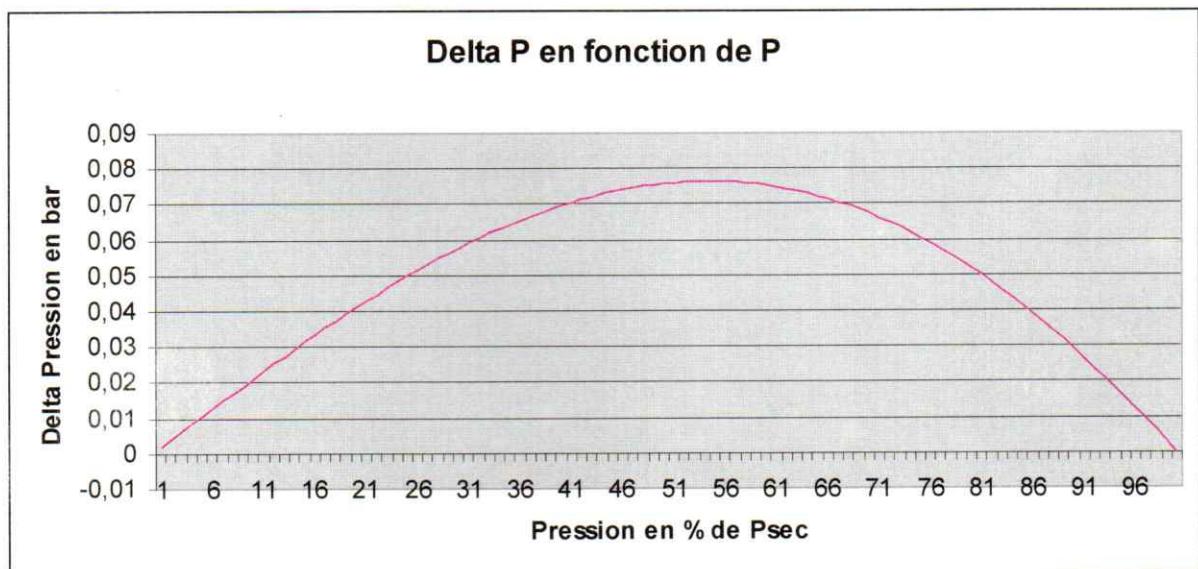


Mais il est nécessaire de mettre au point une version équivalente du programme fonctionnant sous environnement Windows, permettant également d'afficher des graphiques, cela se présenterait le cas échéant comme ci après :

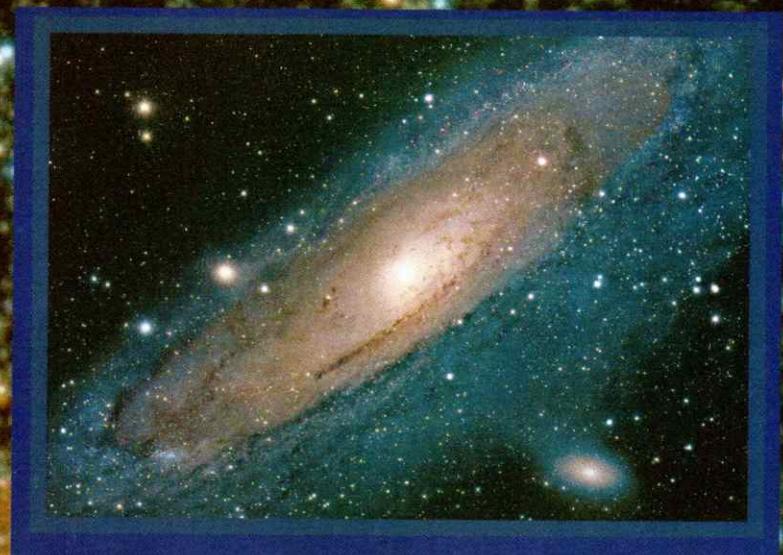
D'abord l'opérateur clique sur le numéro du miroir correspondant avant d'entrer le paramètre de pression Pseq (pression maximale atteinte).



L'application affiche ensuite les valeurs de pressions et de courbures correspondantes, selon des cellules et selon un graphique.



Ainsi l'application existe en version fonctionnant sous Windows et sous Linux, elle est donc utilisable sur le plus grand nombre de stations PC.



Conception



Sous Linux nous allons utiliser la version SUSE 9.2, nous développerons en langage C ANSI sur l'éditeur GVIM, qui nous appellerons grâce à la commande :

- Gvim &

Nous allons organiser l'arborescence de fichiers page suivante.

L'éditeur graphique sera la librairie pgplot, appelée par l'instruction en tête de programme :
`#include cpgplot.h`

Nous utiliserons également la librairie d'entrées sorties standard :

`#include stdio.h`

Ainsi que la librairie mathématique :

`#include math.h`

Le fichier correspondant au programme lui-même s'appelle `essai_pgplot01.c`, l'extension « .c » désignant le langage dans lequel est écrit le fichier.

Les fichiers `fifi.dat` et `fizeau` contiennent respectivement :

- Pour le premier les caractéristiques des différents miroirs :

num. Miroir	Pmax	A1	A3	A5	C0	alpha	beta	gamma
18	6,49944	0,40532157	0,001286003	3,78242E-06	0,23153237	0,541943	0,000832045	6,14363E-05
19	6,4996	0,39729838	0,000804944	4,5508E-06	-0,04608024	0,542227	0,00153209	1,63369E-05
21	7,49989	0,40607636	0,00126815	4,3916E-06	0,26874324	0,552754	0,00200876	4,55992E-05
22	6,70036	0,39226216	0,001165073	5,13692E-06	0,27546452	0,535643	0,000924827	0,000051172
23	7,50004	0,43840233	0,001127128	5,34034E-06	0,22333076	0,529903	0,0010814	3,42985E-05
25	7,29972	0,40438133	0,000811113	4,37047E-06	0,25118364	0,539063	0,00179396	3,58896E-05
26	7,30027	0,46001738	0,000562812	8,08951E-06	0,28523802	0,536214	0,0012016	4,31965E-05
27	7,30012	0,43717096	0,000912421	4,00211E-06	0,10589429	0,56045	0,00194258	3,54908E-05
28	6,50026	0,4114861	0,000581454	4,42791E-06	0,13007918	0,565176	0,00200381	1,18987E-05

- Pour le second les coordonnées en abscisse et en ordonnée des mesures effectuées par la méthode de Fizeau. Notons qu'à chaque mesure correspond un fichier Fizeau, il y a donc plusieurs fichiers « Fizeau », ce qui nous obligera à les ouvrir à tour de rôle.

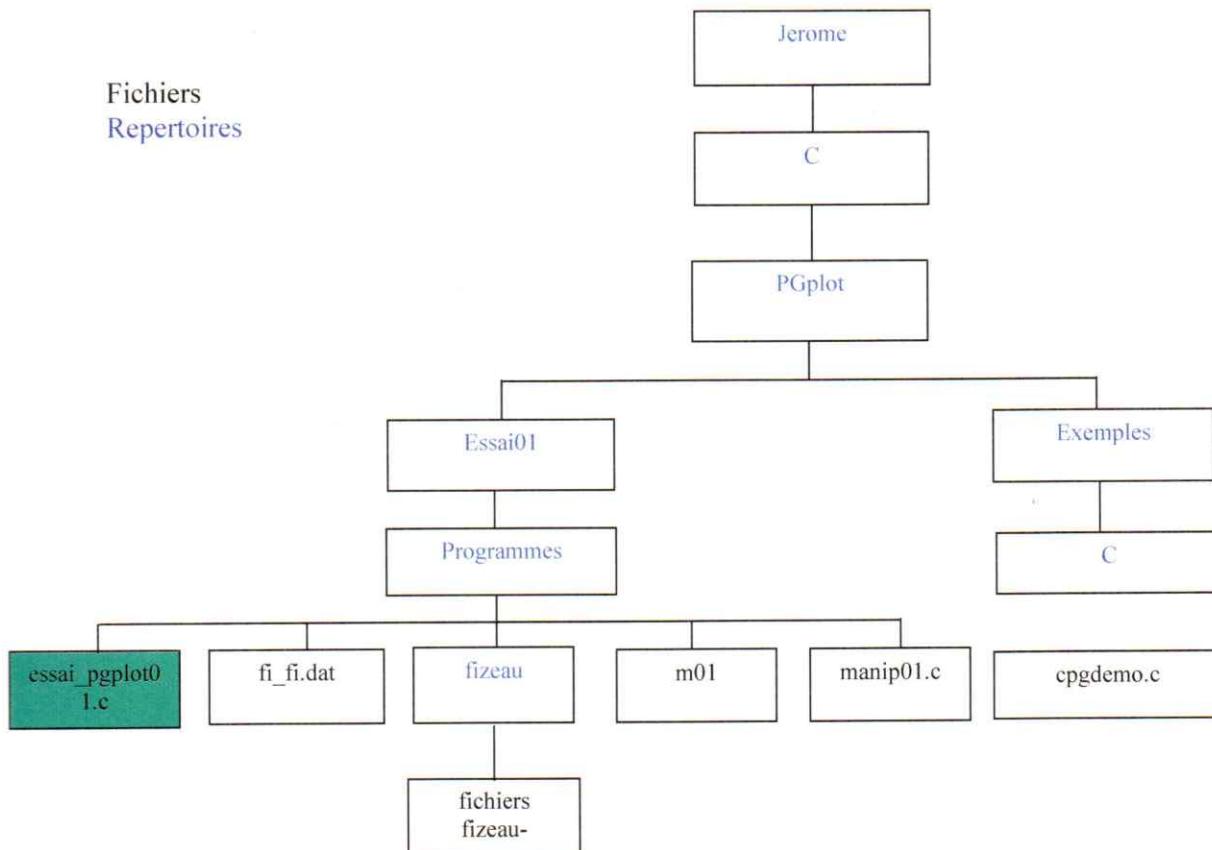
Le fichier « M01 » rassemble des instructions nécessaires à la compilation et à l'édition de liens, il est appelé par l'instruction :

- Sh M01

Pour exécuter ensuite le programme on utilisera l'expression :

- `./essai_pgplot01`

Arborescence des fichiers sous Linux



Secondairement, le fichier manip01.c contient des programmes d’entraînement à la mise en œuvre de certains procédés : lecture et écriture sur le périphérique disque en langage C, conversion de données etc...

Dans le répertoire « exemples » nous trouvons le répertoire « C » (hébergeant des exemples en langage C) puis le fichier « cpgdemo.c » contenant diverses application de la librairie cpgplot permettant de réaliser des graphes.

Tous ces fichiers et répertoires sont regroupés dans un répertoire principal « pgplot », lui-même inclus dans un répertoire « C » puis dans « Jérôme », du nom du développeur.

Afin de créer un répertoire on utilisera la commande :

- /mkdir « nom du répertoire »

Pour lister le contenu d'un répertoire on utilisera la commande :

- LL ou LS « nom du répertoire »

Pour se déplacer en aval dans l'arborescence on utilisera la commande :

- CD « nom du répertoire de destination »

Pour remonter d'un niveau dans l'arborescence :

- CD ..

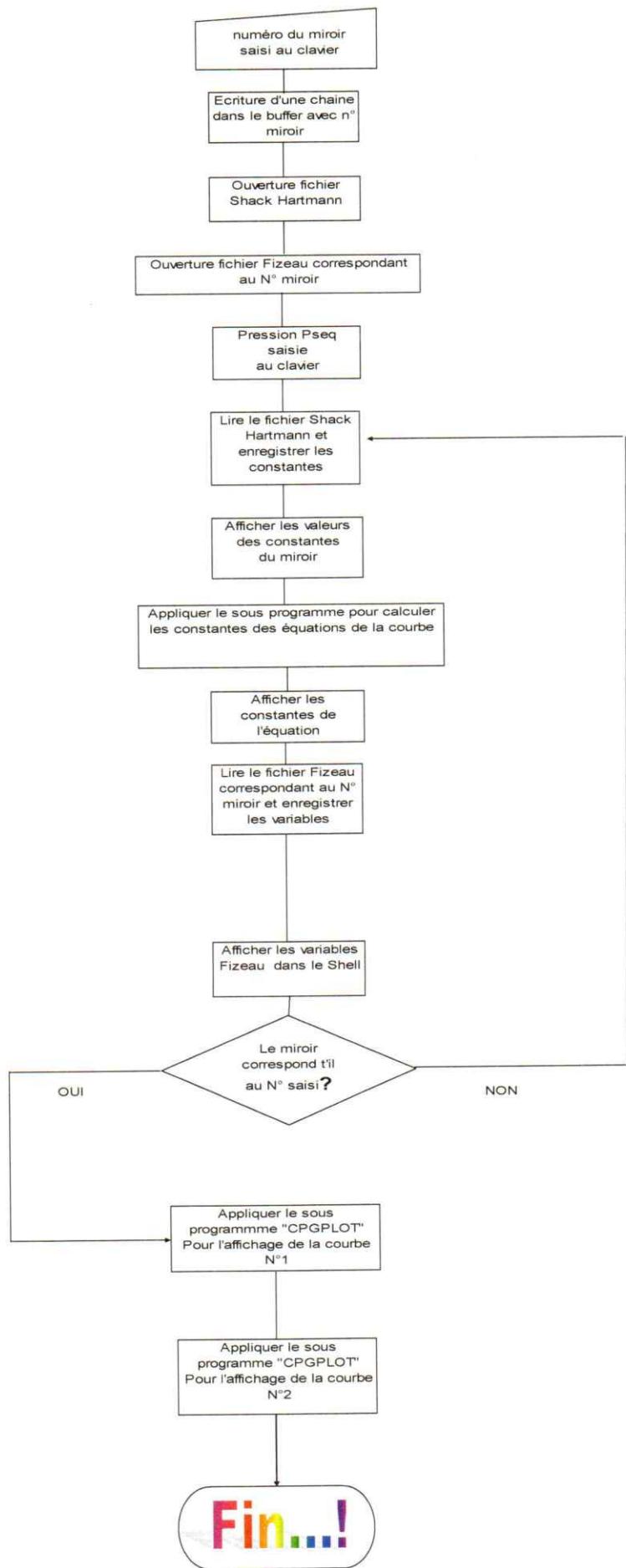
Pour copier un fichier dans un répertoire :

- CP « nom du fichier » « nom du répertoire de destination »

L'environnement Linux présente l'avantage de la gratuité, pour le système d'exploitation comme pour les applications, en revanche sa mise en œuvre dans le cas de la version SUSE est plus délicate que Windows et peut dérouter au premier abord.

Linux est très utilisé dans le monde de la recherche scientifique, c'est un système réputé stable par ceux qui l'utilise régulièrement.

En ce qui concerne l'organigramme du programme en C regardons à la page suivante :



Sous Microsoft Excel2003 et environnement Windows XP, nous partirons de la feuille contenant les données correspondant à chaque miroir, il n'y aura pas de référence à des fichiers externes. L'application n'est contenu que dans un seul fichier Excel.

Nous utiliserons l'environnement de développement intégré (EDI) d'Excel2003, le langage correspondant est le VBA (pour Visual Basic Application).

A la différence de la programmation en C qui est dite « procédurale », il s'agit là d'une programmation « événementielle », autrement dit des « événements » (exécutions d'instruction) se déclanchant suite à une action, par exemple le clic avec la souris sur le bouton « OK » d'une boîte de dialogue. Ce type d'application rend l'utilisation plus « conviviale »

Le VBA est aussi un langage orienté objet : on peut par exemple sélectionner la *propriété* « couleur » de la *classe* « police d'écriture » dans *l'objet* « cellule » et lui attribuer la *valeur* « bleu » ...

Pour faire fonctionner cette application il sera néanmoins nécessaire d'installer préalablement Office Excel sur l'ordinateur utilisateur : en effet à la différence de VB 6.0, VBA ne fonctionne qu'à partir de l'application correspondante dans Office : Word, Excel, Access ou PowerPoint.

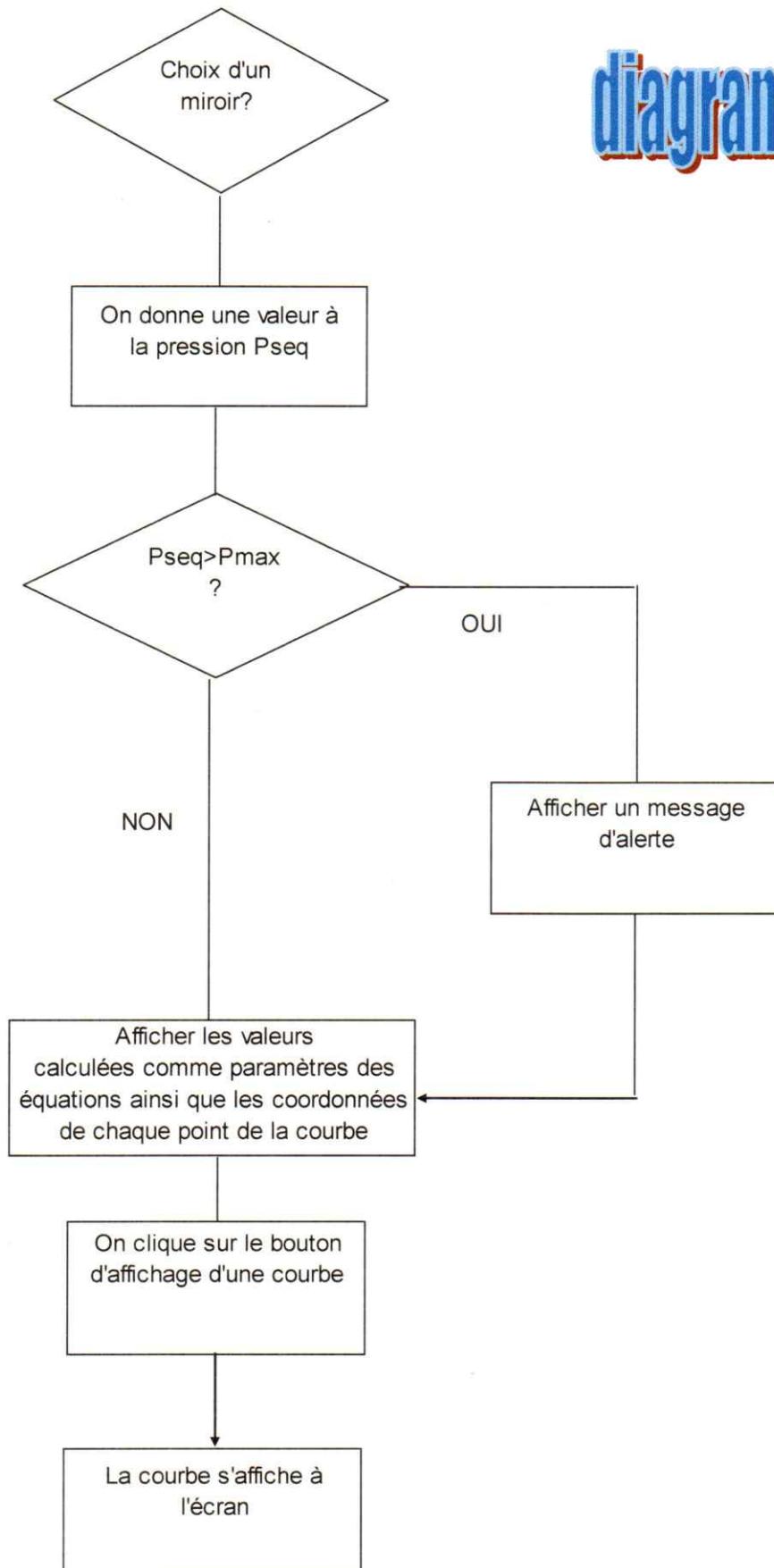
Cette application ne nécessite pas une puissance de calcul particulière : pour information l'ordinateur utilisé pour le développement est un portable avec un processeur Intel Centrino 1,5Ghz, un disque dur de 60Go, 512Mo de RAM et une carte graphique ATI 9200 avec 64Mo de mémoire vidéo.

Le système d'exploitation est Windows XP avec le Service Pack 2, l'application est Office2003 avec Excel.

L'organigramme utilisé précédemment s'avère moins bien adapté à la programmation événementielle que procédurale (mais il servira quand même largement à guider le développement...), ici nous présenterons plutôt un « diagramme d'utilisation », énumérant les différentes actions effectuées par l'utilisateur.

Regardons le à la page suivante...

diagramme utilisateur



Pour lancer l'application sous Excel2003, la procédure est la suivante :

- double-cliquer sur le fichier « fi_fi3 », qui se présente comme ci-dessous :



fi_fi3
Feuille de calcul Microsoft Excel
129 Ko

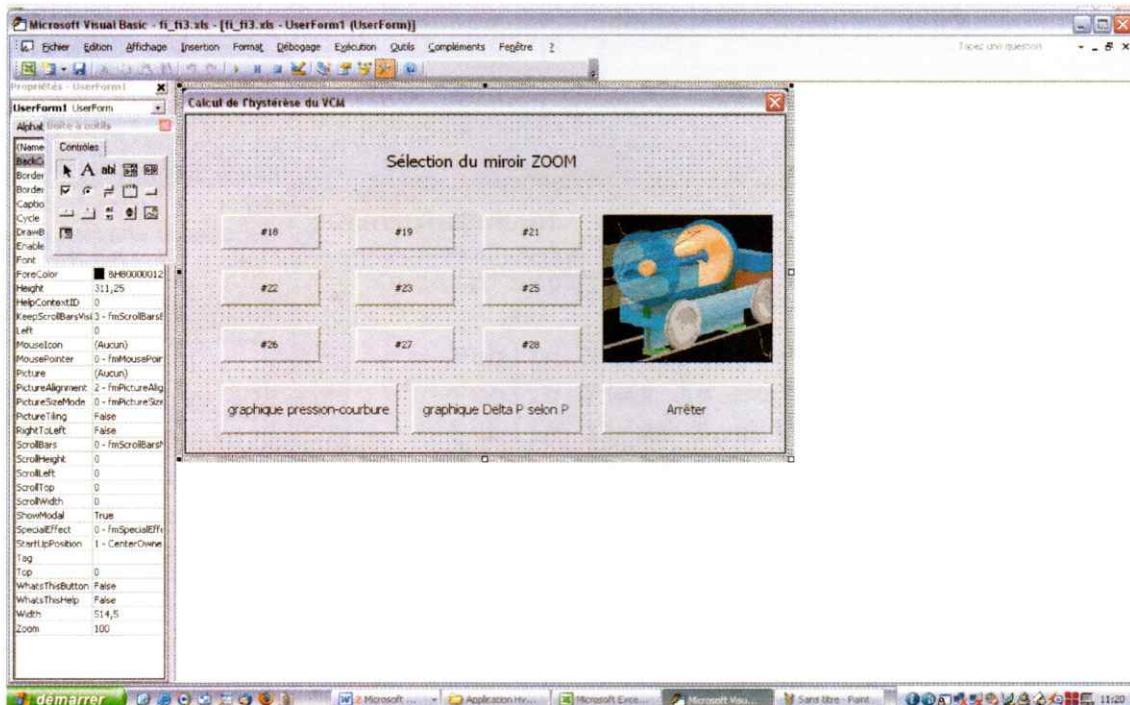
- Une fois Excel2003 ouvert l'écran se présente comme suit :

	A	B	C	D	E	F	G	H	I	J	K	L
3	Beta, Micro	Pmax	A1	A3	A5	C0	alpha	beta	gamma			
4												
5	18	6,49944	0,40532157	0,001268003	3,78342E-06	0,23153237	0,541943	0,000832045	6,14363E-06			
6	19	6,4996	0,39729838	0,001268044	4,3916E-06	-0,04900024	0,542227	0,00152099	1,63392E-06			
7	7,49989	0,40607636	0,00126815	4,3916E-06	0,26874324	0,562754	0,00200875	4,55932E-06				
8	22	6,70036	0,39226216	0,001165073	5,13692E-06	0,27546452	0,536454	0,000924627	0,00051172			
9	23	7,50004	0,43840233	0,001127128	5,34034E-06	0,22333076	0,550063	0,00179614	3,58989E-06			
10	25	7,29972	0,40438133	0,000811113	4,37047E-06	0,25118364	0,539063	0,00179606	3,58989E-06			
11	26	7,30027	0,46001738	0,0006562812	8,08951E-06	0,28523802	0,586214	0,00120168	4,31985E-06			
12	27	7,30012	0,43717096	0,000912421	4,00211E-06	0,10589429	0,66046	0,00194268	3,54600E-06			
13	28	6,50026	0,4114861	0,000581454	4,42791E-06	0,13007918	0,566176	0,0200381	1,18982E-06			
14												
15												
16	Combure C-C0											
17	Pression à la montée											
18	Pression à la descente											
19												
20												
21	Pression											
22	Delta pression											
23												
24												
25												
26												
27												
28												
29												
30												
31												
32												
33												
34												
35												
36												
37												

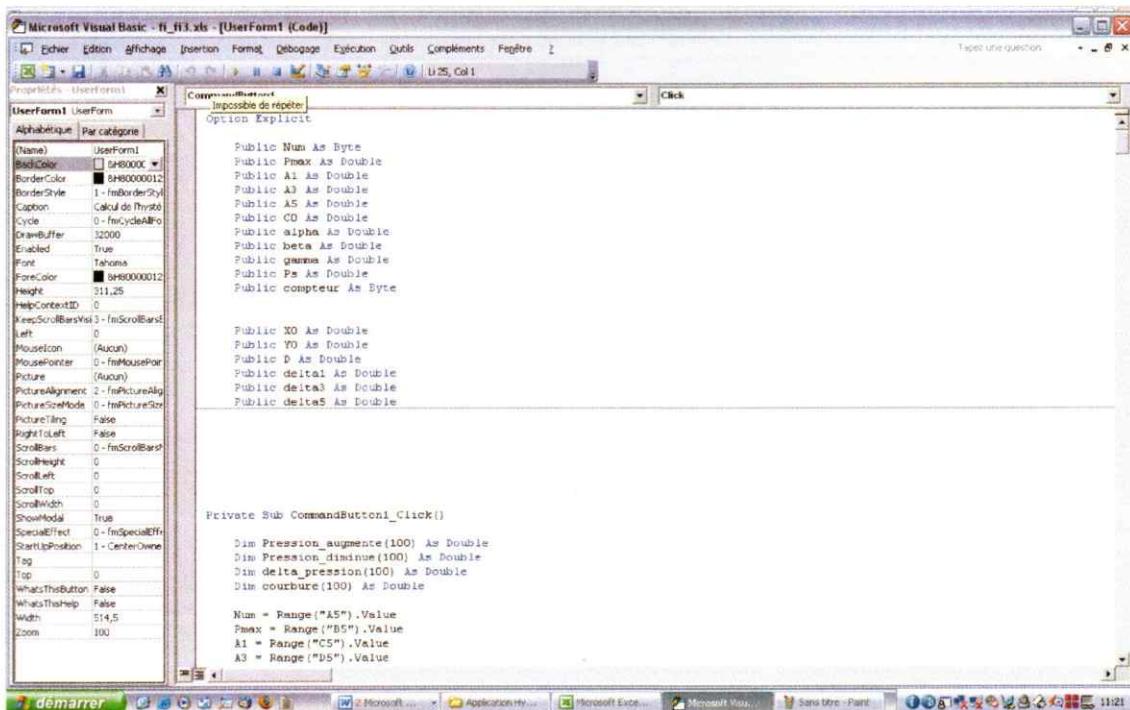
- Aller dans le menu « Outils » en haut de l'écran puis sélectionner « macro » puis « Visual Basic Editor » ou bien utiliser les touches Alt+F11

	B	A1
6,49944	0,40532157	
6,4996	0,39729838	
7,49989	0,40607636	
6,70036	0,39226216	
7,50004	0,43840233	
7,29972	0,40438133	
7,30027	0,46001738	
7,30012	0,43717096	
6,50026	0,4114861	

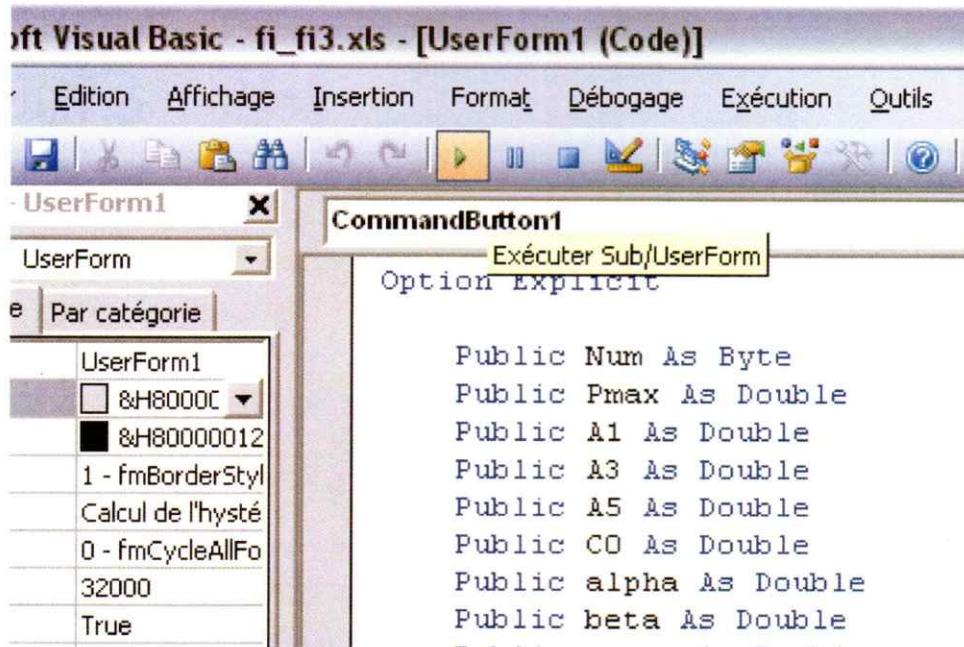
- L'écran se présente alors comme suit, on peut dès lors lancer l'application ou intervenir sur les objets de l'interface homme-machine (IHM) :



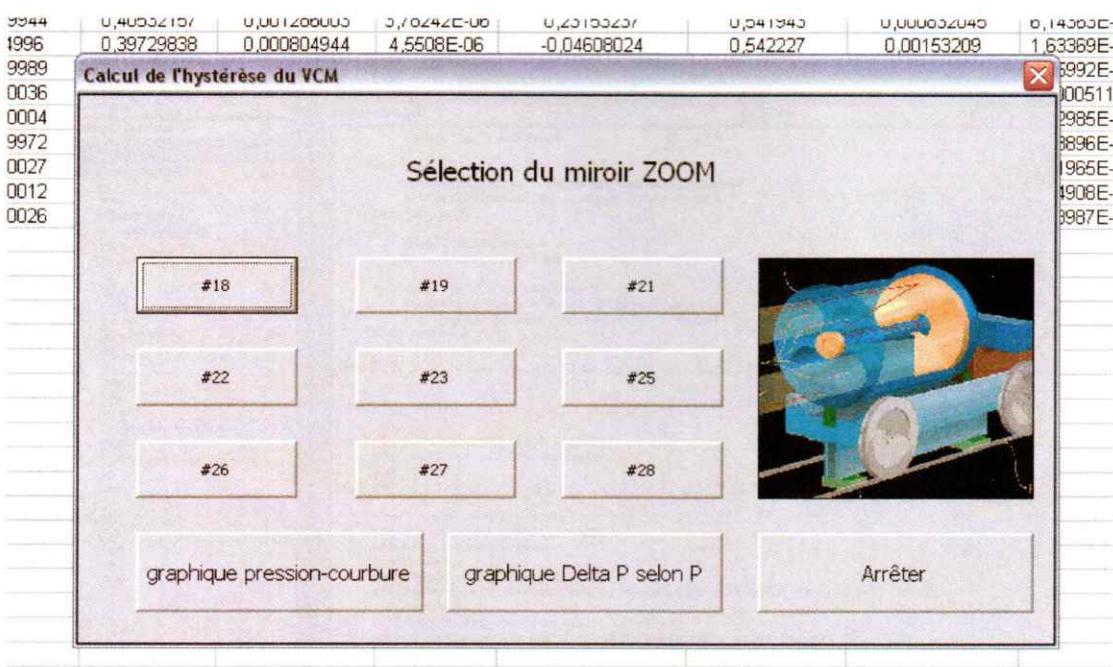
- Ou sur le code comme suit selon les cas, cela n'a pas d'importance pour lancer l'application :



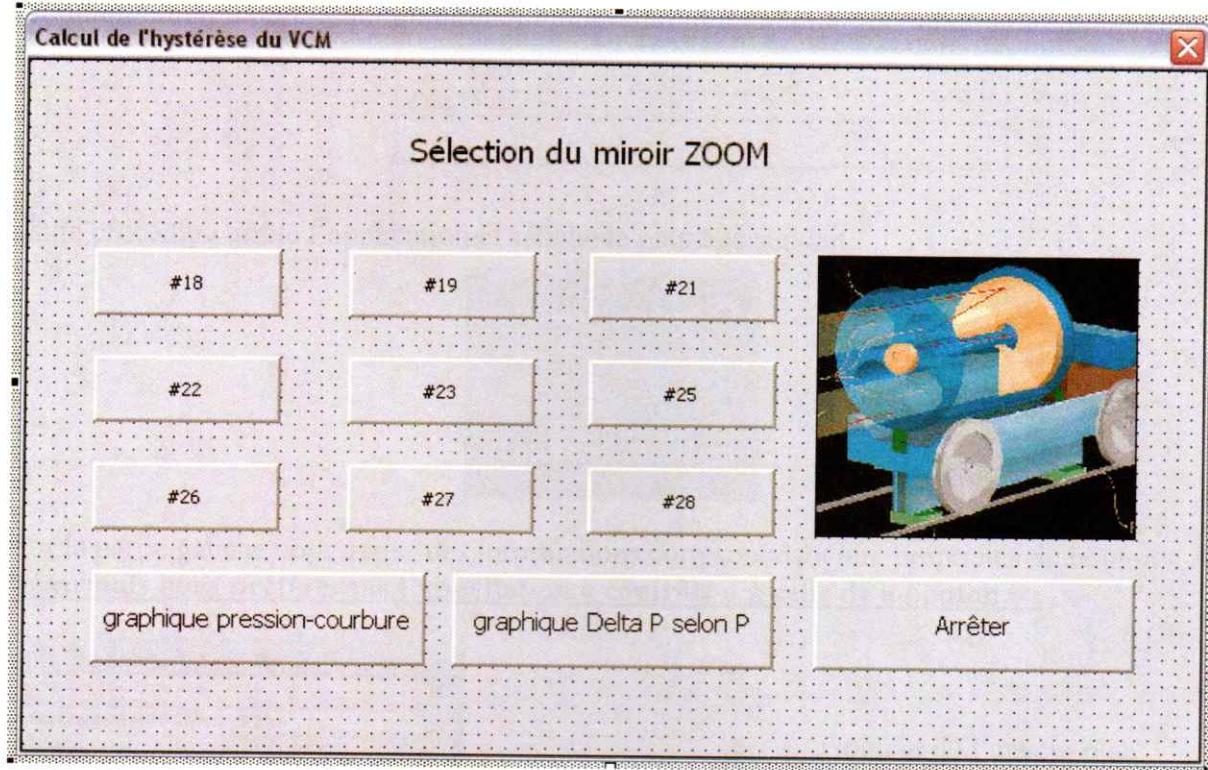
- Cliquer ensuite sur l'icône en haut de l'écran :



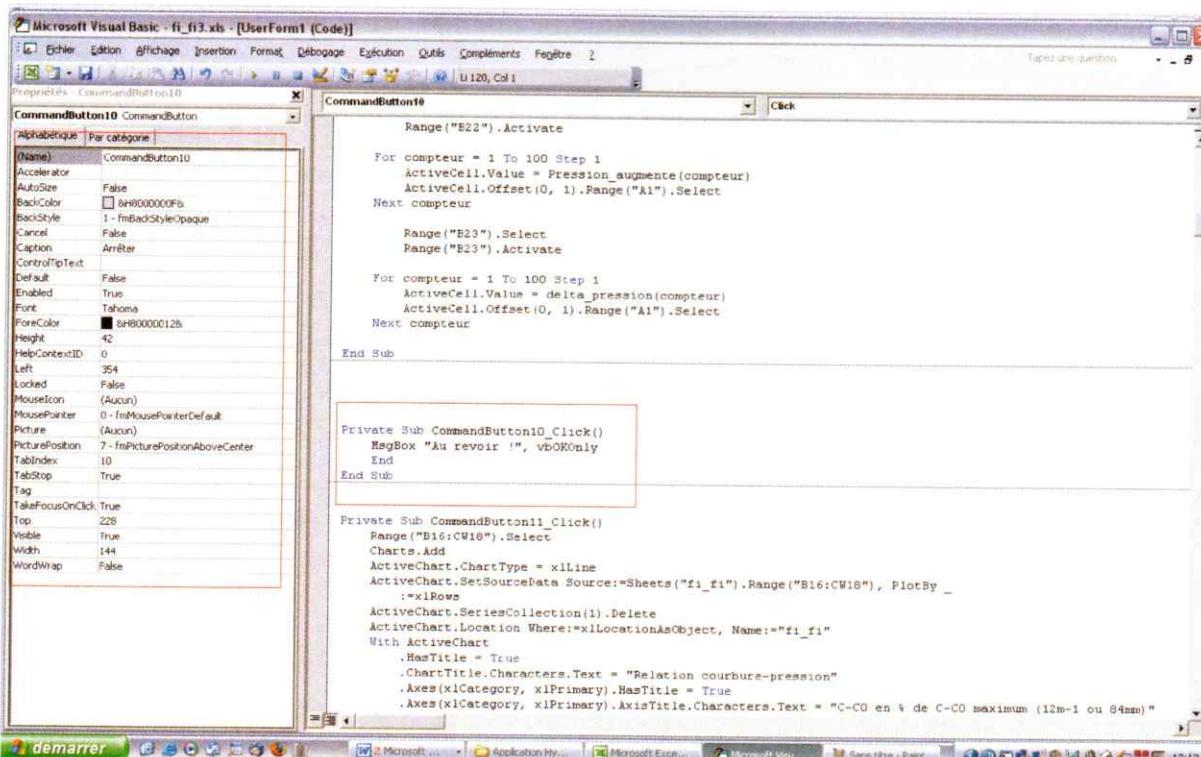
- L'application démarre et il suffit alors de cliquer sur les boutons :



Attardons nous un instant avec un exemple sur le mécanisme mis en œuvre pour développer l'application :



- Double-cliquons sur le bouton « Arrêter » en bas à droite de l'interface, l'écran suivant s'affiche :



Dans le premier encadré rouge nous voyons la procédure déclenchée par l'événement « cliquer sur le bouton Quitter »

```

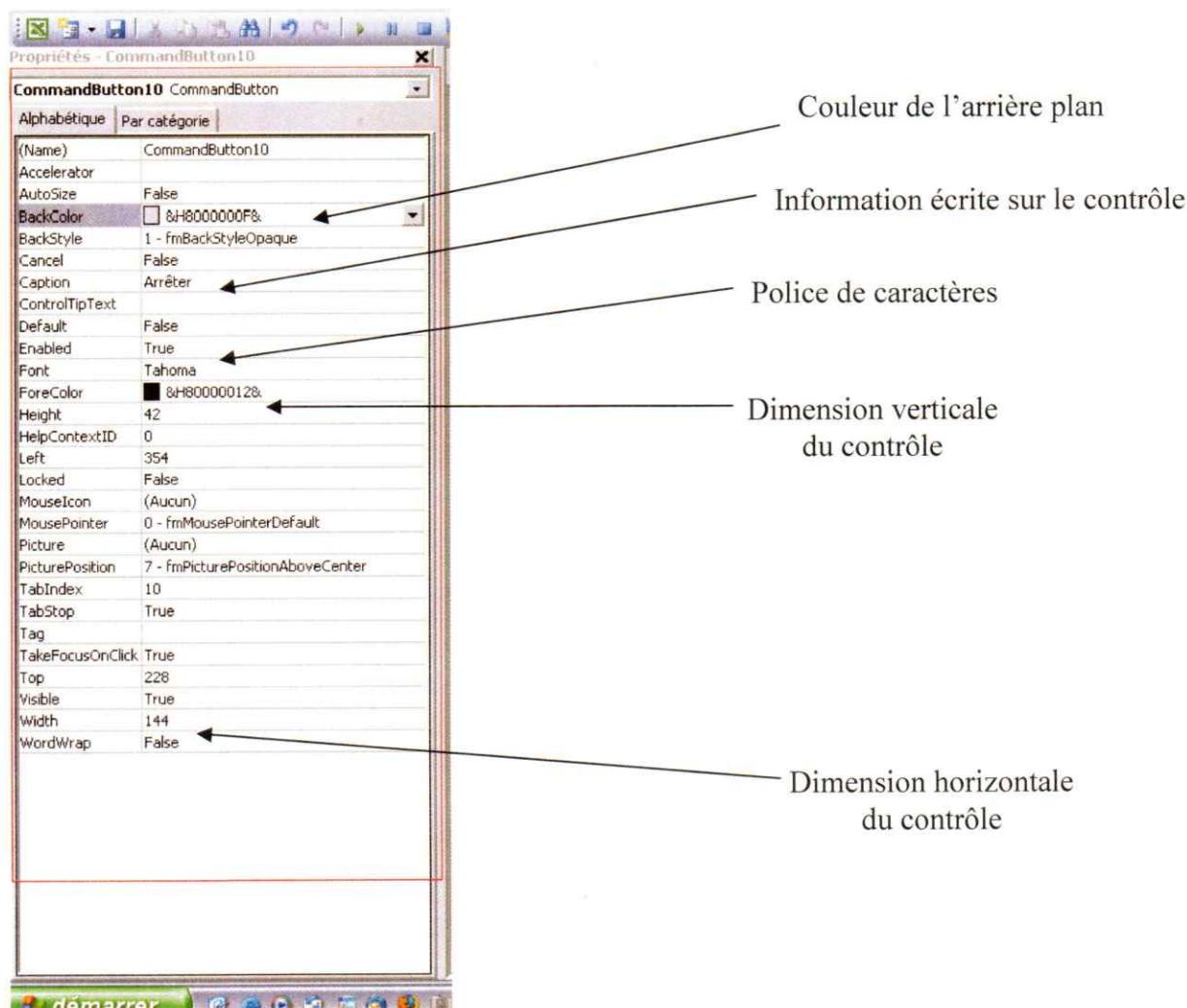
Private Sub CommandButton10_Click()
    MsgBox "Au revoir !", vbOKOnly
End
End Sub

```

Cette procédure arrête l'application et affiche le message suivant :



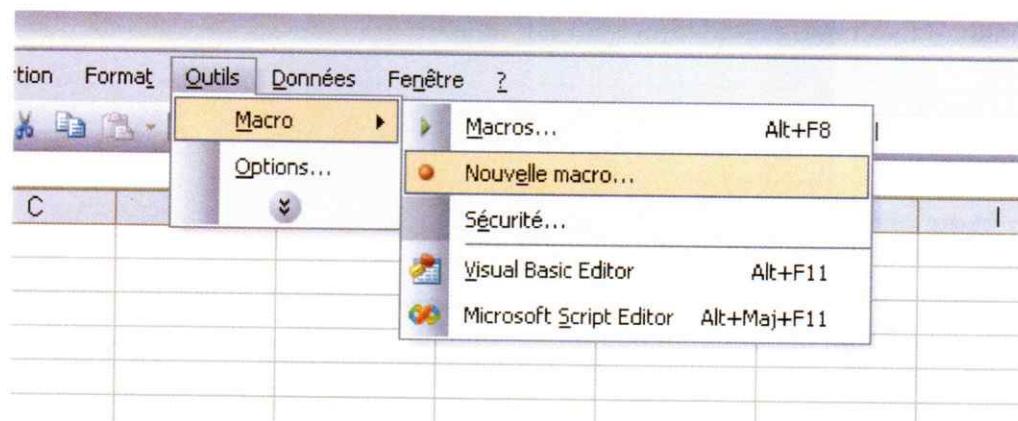
Sur la partie gauche nous avons des informations correspondant aux propriétés du bouton « Arrêter », **désormais nous préférerons l'appellation « contrôle » à celle de « bouton » :**



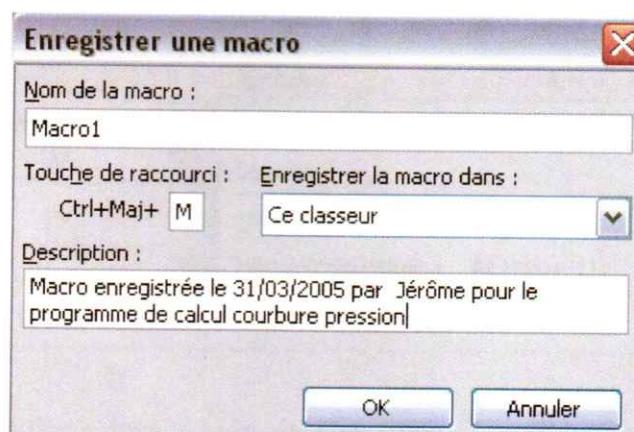
Il est également possible de créer une macro en enregistrant une série d'opérations manuelles, comme celle consistant à créer un graphique à partir de sélection d'une plage de cellules contenant des valeurs. Cette macro peut ensuite être récupérée sous forme de code et insérée dans le programme.

Cela se fait de la manière suivante :

- Créer la macro désirée, par exemple la réalisations d'un graphique, pour cela on sélectionne :



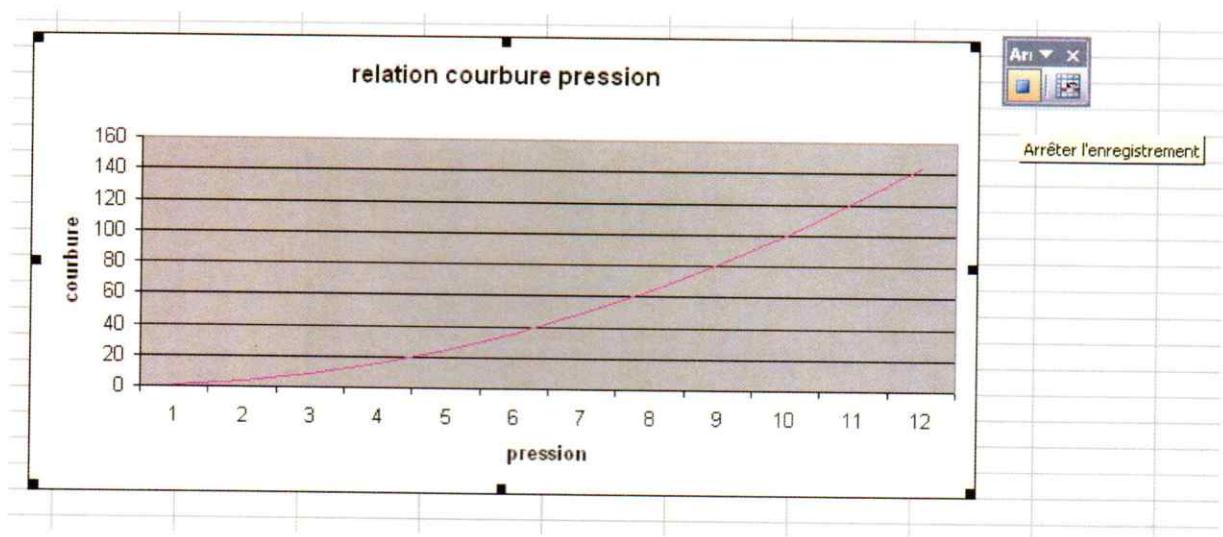
- La boîte de dialogue suivante apparaît sur l'écran, on la complète à la manière suivante :



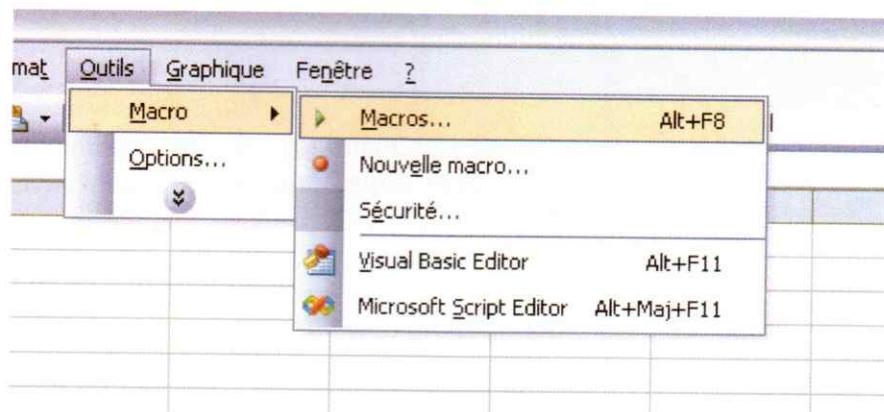
- En cliquant sur OK l'icône indique que la macro est en cours d'enregistrement.



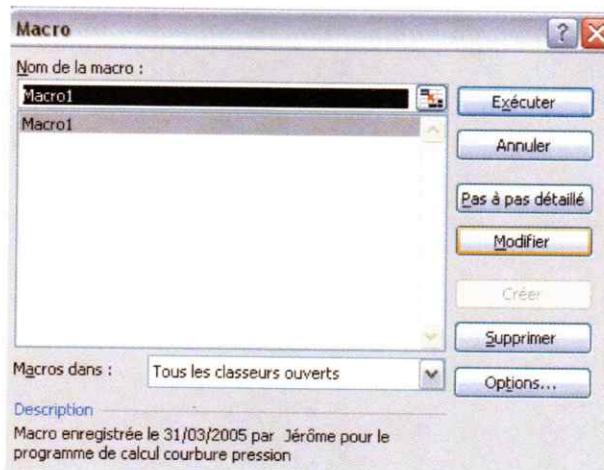
- On réalise le graphique puis on stoppe l'enregistrement en cliquant sur :



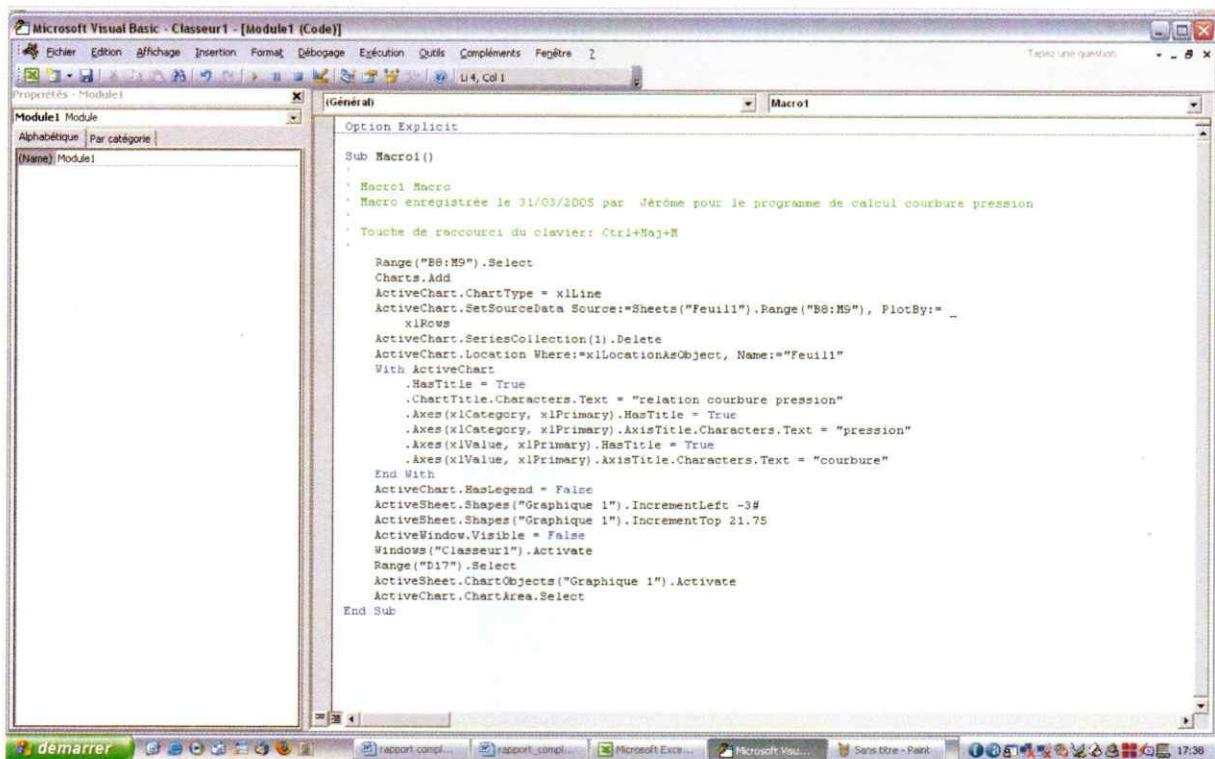
- On sélectionne ensuite :



- Puis finalement « Modifier » dans la boîte de dialogue :

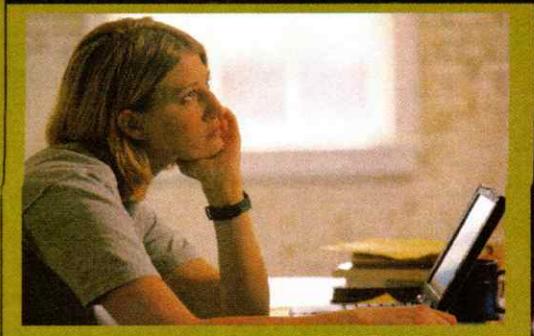


- Voici finalement le code Visual Basic que l'on pourra extraire pour ajouter dans le programme par un « copier coller » :



Nous veillerons cependant à ajouter des commentaires dans le code afin de rendre celui plus compréhensible et faciliter les éventuelles interventions ultérieures. Pour cela on écrira :

- ' texte du commentaire



Programmation



Venons en maintenant à la programmation à proprement parler, en langage C cela nous donne :

```
#include « cpgplot.h » /*librairie servant au traçage des graphiques en langage C*/
#include <math.h> /*librairie des fonctions mathématiques*/
#include <stdio.h> /*librairie des entrées sorties standard*/

Int main()
{
    Int i,c,d,choix ; /*variables pour le choix du miroir*/
    Int num ;
    Float xr[201],yr[201] ; /*tableau contenant les valeurs pour la courbe de montée*/
    Float xs[201],ys[201] ; /*tableau contenant les valeurs pour la courbe de descente*/
    Float nsup=0.0 ; /*variable initialisée pour l'ajustement de la courbe*/
    Double Ps ; /*variable correspondant à Pseq*/
    Double Pw,A1,A3,A5,C0,Alpha,Beta,Gamma ; /* constantes pour chaque VCM*/
    Double Cf1,Pf1,Cf2,Pf2,Cf3,Pf3,Cf4,Pf4,Cf5,Pf5,Cf6,Pf6 ; /*courbure et pression mesurées au Fizeau*/
    Double Delta1,Delta3,Delta5; /*variable utilisées dans l'équation de l'hystérèse*/
    Double A,B,C,coeff /*variables de contrôle des calculs*/
    FILE *fptra,*fptrb ; /*pointeurs des fichiers Shack-Hartmann et Fizeau*/
    Double X0,Y0 ; /*variables pour déterminer l'équation de l'hystérèse*/
    Double D ; /*variables pour déterminer l'équation de l'hystérèse*/
    Char buffer[20] ; /*zone d'écriture d'une chaîne représentant le fichier à ouvrir*/

    /*On demande le numéro du miroir*/

    printf("entrez le numéro du miroir (entre 18 et 28): \n");
    scanf("%d",&choix) ;
    sprintf(buffer,"fizeau/fizeau-m%d.dat"); /*on écrit le chemin et le nom du fichier dans le buffer*/

    /*on ouvre le fichier contenant les valeurs mesurées au Shack-Hartmann*/
    fptra=fopen("fi_fi.dat","r") ;

    /*on ouvre les fichiers de manière selective pour les valeurs obtenues au Fizeau*/
    fptrb=fopen(buffer,"r") ;
    /*on demande la valeur de Pseq*/

    {

        printf("diminution de la pression miroir # %d\n",choix) ;
        printf("entrez la valeur de Pseq pour le second graphique (Pseq<8bars) !!): \n") ;
        scanf("%lf",&Ps) ;

        /*on scanne chaque ligne sur les fichiers*/

        do{
            /*mesures à l'aide du dispositif de Shack-Hartmann*/

            fscanf(fptra,"%d %lf %lf %lf %lf %lf %lf %lf %lf", &num, &Pw,&A1,&A3,&A5,
&C0,&Alpha,&Beta,&Gamma) ;
```

```

    printf("%d %lf %lf %lf %lf %lf %lf %lf %lf %lf", &num, &Pw,&A1,&A3,&A5,
&C0,&Alpha,&Beta,&Gamma) ;

/*mesures à l'aide du dispositif de Fizeau, les 2 mesures ont été réalisées indépendamment*/

fscanf(fptr2," %lf %lf",
Cf1,Pf1,Cf2,Pf2,Cf3,Pf3,Cf4,Pf4,Cf5,Pf5,Cf6,Pf6) ;

printf("%lf %lf %lf",
Cf1,Pf1,Cf2,Pf2,Cf3,Pf3,Cf4,Pf4,Cf5,Pf5,Cf6,Pf6) ;

/*on determine les valeurs de X0 et Y0*/

X0=Alpha*Ps ;
Y0=Beta*pow(Ps,2)+Gamma*pow(Ps,4) ;
printf("valeur de X0 : %f, valeur de Y0 : %f\n",X0,Y0) ;

/*on determine enfin les valeurs Delta*/

D=pow(pow(X0,2)-pow(Ps,2),2) ; /*D est un element constant du denominateur*/

Delta1=Y0*(-5*(pow(X0,2)*pow(Ps,2))+3*pow(Ps,4))/(2*X0*D)
Delta3=Y0*(5*pow(X0,4)-pow(Ps,4))/(2*pow(X0,3)*D) ;
Delta5=Y0*(-3*pow(X0,2)+pow(Ps,2))/(2*pow(X0,3)*D) ;
printf("valeur de Delta1 : %lf Delta3 : %lf Delta5 : %lf\n", Delta1, Delta3, Delta5) ;

printf("valeur de Psec : %lf\n",Ps) ;
printf("valeur de D : %lf\n",D) ;

c=1 ;

/*tracage du graphique numero 1*/

if(num==choix)
{
    for(i=0 ;i<201 ;i++)
    {
        xr[i]=(12.0/200)*i ;
/*courbure à la montée en pression*/
        yr[i]=A1*((xr[i]-C0)+A3*pow(((xr[i]-C0),3)+A5*pow(((xr[i])-C0),5) ;
/*pression affichée à la montée*/
        xs[i]=(Ps/200)*i ;
/*courbure à la descente en pression*/
        coeff=(Delta*xs[i]+Delta3*pow(xs[i],3)+Delta5*pow(xs[i],5)) ;
/*ceci est la difference de pression due à l'effet d'hysteresis*/
        A=yr[i] ;
        B=coeff ;
        C=A-B ;
        ys[i]=C ;
}

```

```

/*pression affichée à la descente*/
/*on affiche dans le shell des valeurs de contrôle*/
printf("A=%lf B=%lf C=%lf xr[%d]=%lf yr[%d]=%lf coeff=%lf **ys[%d]=%lf**\n",
A,B,C,i,xr[i],yr[i],coeff,i,ys[i]);
}

if (cpgopen("//XWINDOW")<1)
return 1;
cpgev(0.,12.,0.,Ps,0,0); /*echelle en abscisse et en ordonnée*/
cpgp1(6,0.5,8); /*legende du graphe*/
cpgtex(6.2,0.5,"mesures au Fizeau");
cpgtex(6,1.5,"montée en pression");
cpgsci(2); /*couleur de la police en rouge*/
cpgtex(6,1,"réduction de la pression");
cpgsci(1); /*retour à la couleur standard, c'est-à-dire la couleur blanche sur fond noir*/
/*annotations sur les axes ligne ci-dessous*/
cpglab("courbure C-C0 (m-1)","Pression (bar)","calibration du miroir");

printf("Fizeau courbure : %lf pression : %f\n",Cf1,Pf1); /*valeurs de contrôle*/
printf("Fizeau courbure : %lf pression : %f\n",Cf2,Pf2); /*valeurs de contrôle*/
printf("Fizeau courbure : %lf pression : %f\n",Cf3,Pf3); /*valeurs de contrôle*/
printf("Fizeau courbure : %lf pression : %f\n",Cf4,Pf4); /*valeurs de contrôle*/
printf("Fizeau courbure : %lf pression : %f\n",Cf5,Pf5); /*valeurs de contrôle*/
printf("Fizeau courbure : %lf pression : %f\n",Cf6,Pf6); /*valeurs de contrôle*/

cpgslw(3); /*largeur de la courbe*/
cpgle(200,xr,yr); /*on trace la courbe de montée en pression*/
/*on marque sur le graphique numero 1 les valeurs mesurées au Fizeau*/

cpgp1(Cf1,Pf1,8); /*le 8 correspond à un symbole donné*/
cpgp1(Cf2,Pf2,8);
cpgp1(Cf3,Pf3,8);
cpgp1(Cf4,Pf4,8);
cpgp1(Cf5,Pf5,8);
cpgp1(Cf6,Pf6,8);

cpgslw(3); /*largeur de la courbe*/
cpgsls(2); /*choix d'une courbe en pointillés*/
cpgsci(2); /*choix de la couleur rouge pour la courbe*/

cpgle(200,xr,ys); /*on trace la courbe de réduction de la pression*/

getchar(); /*permet d'afficher les graphes l'un après l'autre*/
cpgclos();
c=0;

```

```

/*tracage du graphique numero 2*/

for(i=0 ;i<201 ;i++)
{
    xr[i]=(Ps/200)*i ;
    yr[i]=(Delta1*xr[i]+Delta3*pow(xr[i],3)+Delta5*pow(xr[i],5) ;
/*ceci equivaut à ecrire yr[i]=coeff*/
printf("xr[%d]=%lf  yr[%d]=%lf\n",i,xr[i],i,yr[i]) ; /*valeurs de contrôle*/
/*selon les miroirs pour Pseq<4.2 l'ajustement suivant fonctionne mal...*/
if (yr[i]>nsp)
nsp=yr[i] ;
/*nsp recoit la plus grande valeur atteinte par yr[i] et fixe une hauteur minimale à l'axe des y*/
}

if(cpgopen("XWINDOW")<1)
return 1 ;

/*...on fait donc en sorte que l'échelle du graphique reste adaptée à la courbe*/
if ((Ps>3.5)&&(Ps<=4.2))
cpgenv(0.,Ps,0.,0.04,0,0) ;
else if ((Ps>3.0)&&(Ps<=3.5))
cpgenv(0.,Ps,0.,0.03,0,0) ;
else if (Ps<=3.0)
cpgenv(0.,Ps,0.,0.02,0,0) ;
else cpgenv(0.,Ps,0.,nsp+0.015,0,0) ; /*dans cette situation on utilise la valeur de nsp*/
cplab("Pression (bar)","Delta pression (bar)","représentation graphique de l'effet d'hystérèse");
/*on trace la courbe représentant Delta P en fonction de P (courbe d'hystérèse)*/
cpline(200,xr,yr) ;
getchar() ;
cpclos ;
c=0 ;
} ;

}while(c!=0) ;
fclose(fptr1) ;
fclose(fptr2) ;
}
}

```

En Visual Basic sous Excel 2003, la déclaration de variable est :

Option Explicit

'on déclare les variables

```
Public Num As Byte  
Public Pmax As Double  
Public A1 As Double  
Public A3 As Double  
Public A5 As Double  
Public C0 As Double  
Public alpha As Double  
Public beta As Double  
Public gamma As Double  
Public Ps As Double  
Public compteur As Byte
```

```
Public X0 As Double  
Public Y0 As Double  
Public D As Double  
Public delta1 As Double  
Public delta3 As Double  
Public delta5 As Double
```

Notons que chaque variable est déclarée « Public » afin d'être utilisable dans chaque segment de code. L'événement « cliquer sur un numéro de miroir » déclanche la procédure suivante :

```
Private Sub CommandButton1_Click()  
'on déclare les tableaux recevant des valeurs  
Dim Pression_augmente(100) As Double  
Dim Pression_diminue(100) As Double  
Dim delta_pression(100) As Double  
Dim courbure(100) As Double
```

'Les variables recoivent les valeurs contenues dans les cellules

```
Num = Range("A5").Value  
Pmax = Range("B5").Value  
A1 = Range("C5").Value  
A3 = Range("D5").Value  
A5 = Range("E5").Value  
C0 = Range("F5").Value  
alpha = Range("G5").Value  
beta = Range("H5").Value  
gamma = Range("I5").Value
```

'On demande la pression Pseq

```
Ps = InputBox("Pression Pseq : ", "saisie des paramètres")
```

```
If Ps > Pmax Then
```

```
MsgBox "ATTENTION ! Pression trop élevée ! Détérioration du Miroir !", vbOKOnly + vbInformation
```

```
End If
```

'l'application confirme les coefficients lus dans le tableau correspondant

```
MsgBox "Miroir numéro " & Num & ", Valeur de Pseq " & Ps & " bars", vbOKOnly + vbInformation
```

```
MsgBox "A1=" & A1 & " A3=" & A3 & " A5=" & A5 & " C0=" & C0 & " alpha=" & alpha & "
```

```
beta=" & beta & _
```

```
" gamma=" & gamma
```

'On calcule les coefficients pour les équations

```
X0 = alpha * Ps
```

```
Y0 = beta * Ps ^ 2 + gamma * Ps ^ 4
```

MsgBox "X0=" & X0 & " Y0=" & Y0 'l'application affiche les coefficients calculés

```
D = (X0 ^ 2 - Ps ^ 2) ^ 2
```

```
delta1 = Y0 * (((-5 * X0 ^ 2) * Ps ^ 2) + 3 * Ps ^ 4) / (2 * X0 * D)
```

```
delta3 = Y0 * ((5 * X0 ^ 4) - Ps ^ 4) / (2 * X0 ^ 3 * D)
```

```
delta5 = Y0 * ((-3 * X0 ^ 2) + Ps ^ 2) / (2 * X0 ^ 3 * D)
```

MsgBox "delta1=" & delta1 & " delta3=" & delta3 & " delta5=" & delta5 'l'application affiche les coefficients calculés

'courbure C-C0

```
Range("B16").Select
```

```
Range("B16").Activate
```

For compteur = 1 To 100 Step 1

```
courbure(compteur) = (12 / 100) * compteur
```

```
Pression_augmente(compteur) = A1 * (courbure(compteur) - C0) + A3 * (courbure(compteur) - C0) ^ 3 + A5 * (courbure(compteur) - C0) ^ 5
```

```
ActiveCell.Value = courbure(compteur)
```

```
ActiveCell.Offset(0, 1).Range("A1").Select
```

```
Next compteur
```

For compteur = 1 To 100 Step 1

```
Pression_augmente(compteur) = (Ps / 100) * compteur
```

```
delta_pression(compteur) = delta1 * (Pression_augmente(compteur)) + delta3 * (Pression_augmente(compteur)) ^ 3 + delta5 * (Pression_augmente(compteur)) ^ 5
```

```
Pression_diminue(compteur) = Pression_augmente(compteur) - delta_pression(compteur)
```

```
Next compteur
```

'Pression à la montée

```
Range("B17").Select
```

```
Range("B17").Activate  
  
For compteur = 1 To 100 Step 1  
    ActiveCell.Value = Pression_augmente(compteur)  
    ActiveCell.Offset(0, 1).Range("A1").Select  
Next compteur  
  
'pression à la descente  
Range("B18").Select  
Range("B18").Activate  
  
For compteur = 1 To 100 Step 1  
    ActiveCell.Value = Pression_diminue(compteur)  
    ActiveCell.Offset(0, 1).Range("A1").Select  
Next compteur  
  
'pression courbe 2  
Range("B22").Select  
Range("B22").Activate  
  
For compteur = 1 To 100 Step 1  
    ActiveCell.Value = Pression_augmente(compteur)  
    ActiveCell.Offset(0, 1).Range("A1").Select  
Next compteur  
  
'delta pression courbe 2  
Range("B23").Select  
Range("B23").Activate  
  
For compteur = 1 To 100 Step 1  
    ActiveCell.Value = delta_pression(compteur)  
    ActiveCell.Offset(0, 1).Range("A1").Select  
Next compteur  
  
End Sub
```

L'évènement « cliquer sur le contrôle pour afficher la première courbe » déclanche la procédure page suivante :

```

Private Sub CommandButton11_Click() 'code obtenu à partir d'une macro pour l'affichage de la première courbe
    Range("B16:CW18").Select 'on sélectionne une plage de cellules actives
    Charts.Add 'on lance l'assistant édition de graphique
    ActiveChart.ChartType = xlLine 'on sélectionne le type de graphique, ici une courbe
    ActiveChart.SetSourceData Source:=Sheets("fi_fi").Range("B16:CW18"), PlotBy _ :=xlRows 'les données proviennent de la feuille fi_fi et de la plage de cellules spécifiée
    ActiveChart.SeriesCollection(1).Delete 'on efface la légende marquée "série 1 série 2"
    ActiveChart.Location Where:=xlLocationAsObject, Name:="fi_fi"

With ActiveChart 'ensemble des propriétés de "ActiveChart"
    .HasTitle = True 'propriété du graphique "posséder un titre" à "vraie" (booléen)
    .ChartTitle.Characters.Text = "Relation courbure-pression" 'intitulé du graphique
    .Axes(xlCategory, xlPrimary).HasTitle = True 'propriété axe des abscisses "posséder un titre" à "vraie" (booléen)
        .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = "C-C0 en % de C-C0 maximum (12m-1 ou 84mm)" 'titre axe des abscisses
        .Axes(xlValue, xlPrimary).HasTitle = True 'propriété axe des ordonnées "posséder un titre" à "vraie" (booléen)
            .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "Pression en bars" 'titre axe des ordonnées
End With
    ActiveChart.HasLegend = False 'propriété "posséder une légende" du graphique à "faux" (booléen)
End Sub

```

L'évènement « cliquer sur le contrôle pour afficher la première courbe » déclanche la procédure suivante :

```

Private Sub CommandButton12_Click() 'code obtenu à partir d'une macro pour l'affichage de la seconde courbe
    Range("B22:CW23").Select 'on sélectionne une plage de cellules actives
    Charts.Add 'on lance l'assistant édition de graphique
    ActiveChart.ChartType = xlLine 'on sélectionne le type de graphique, ici une courbe
    ActiveChart.SetSourceData Source:=Sheets("fi_fi").Range("B22:CW23"), PlotBy _ :=xlRows 'les données proviennent de la feuille fi_fi et de la plage de cellules spécifiée
    ActiveChart.SeriesCollection(1).Delete 'on efface la légende marquée "série 1 série 2"
    ActiveChart.Location Where:=xlLocationAsObject, Name:="fi_fi"

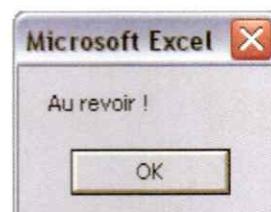
With ActiveChart 'ensemble des propriétés de "ActiveChart"
    .HasTitle = True 'propriété du graphique "posséder un titre" à "vraie" (booléen)
    .ChartTitle.Characters.Text = "Delta P en fonction de P" 'intitulé du graphique
    .Axes(xlCategory, xlPrimary).HasTitle = True 'propriété axe des abscisses "posséder un titre" à "vraie" (booléen)
        .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = "Pression en % de Psec" 'titre axe des abscisses
        .Axes(xlValue, xlPrimary).HasTitle = True 'propriété axe des ordonnées "posséder un titre" à "vraie" (booléen)
            .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "Delta Pression en bar" 'titre axe des ordonnées
End With
    ActiveChart.HasLegend = False 'propriété "posséder une légende" du graphique à "faux" (booléen)
End Sub

```

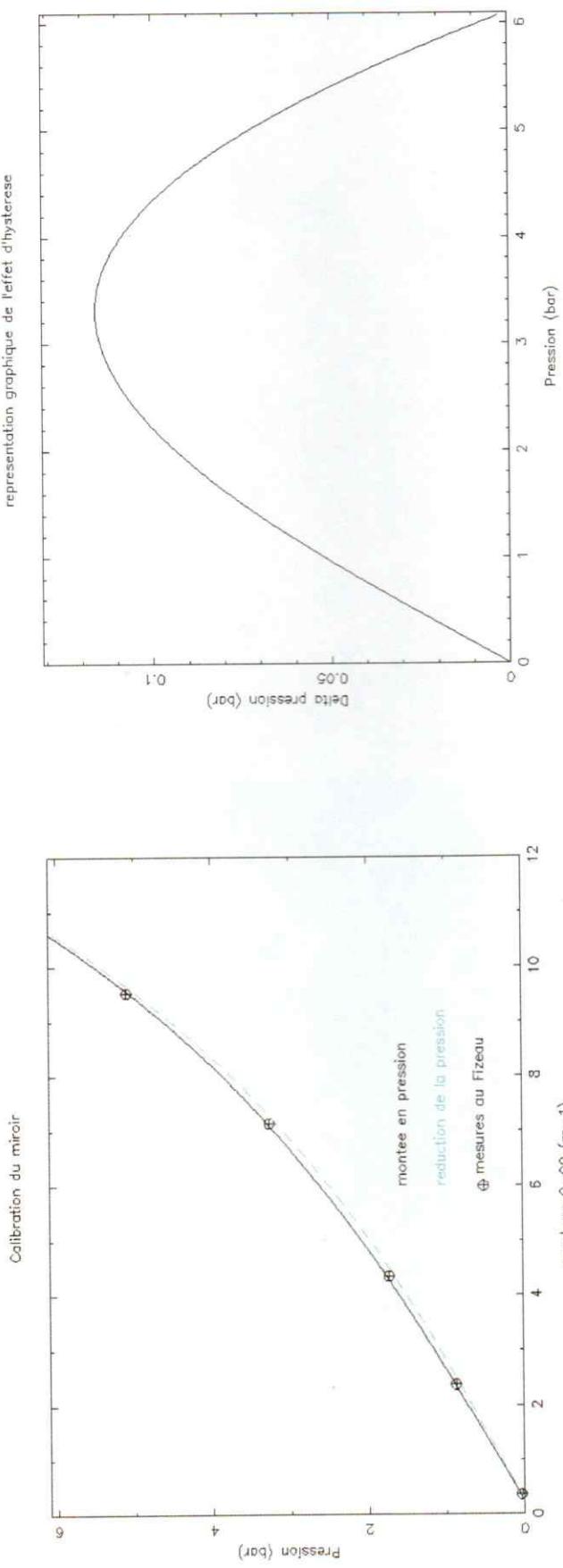
L'évènement « cliquer sur le contrôle pour arrêter l'application» déclanche la procédure suivante :

```
Private Sub CommandButton10_Click()
    MsgBox "Au revoir !", vbOKOnly
End
End Sub
```

Et affiche le message suivant...



Courbes issues de l'application en langage C sous Linux (document annexe)



Relation courbure-pression lors de l'augmentation de la pression (ligne en trait plein) et lors de la diminution (ligne en pointillés), notons la différence effective due à l'effet d'hystérèse, qui est maximale pour une courbure moyenne.

Valeur de delta P en fonction de P, illustre le graphique de gauche, delta P est maximum pour une courbure intermédiaire.