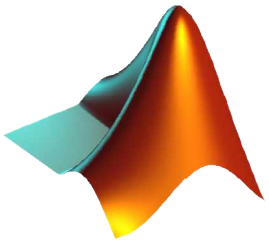
HANAT François
N° étudiant : 20300509

Méthode Numériques
Niveau 1



Rapport de Travaux Pratiques

UNIVERSITÉ de Picardie Jules Verne

SOMMAIRE

I. Calcul d'intégrales par la méthode de Simpson

1. Présentation du problème

Le problème consiste au calcul du travail fourni par une machine suivant un cycle de Carnot. On considère un gaz diatomique supposé parfait. L'utilisateur fournit les valeurs de pression et de volume.

2. Contraintes

- Ø On demande d'effectuer certains calculs d'intégrales par la méthode de Simpson.
- Ø On demande d'élaborer un programme utilisant plusieurs fonctions.

3. Analyse du problème

Le cycle thermodynamique est déterminé par 4 états i, auxquels sont associés une pression p(i) et un volume V(i).

On imagine guider élégamment l'utilisateur dans la phase d'entrée de ces valeurs par l'affichage d'un diagramme de Clapeyron du cycle considéré.

On prendra garde aux unités. On demandera les pression en bars, que l'on convertira ensuite en Pascal, et les volumes en m^3.

Un cycle de Carnot étant constitué de deux types de transformations (isotherme, adiabatique), il est tout d'abord imaginable de structurer le problème avec deux fonctions, ayant pour but de traiter le calcul des travaux spécifiques à ces deux transformations.

- Ø Pour une adiabatique, on utilisera la seconde loi de Joule (Q=0, W=U=f(T)).
- Ø Pour une isotherme, une intégrale en 1/x se présentera. Nous la résoudrons par la méthode de Simpson.

On imagine enfin organiser le programme autour d'un script principal (que nous nommerons « Carnot »), à partir duquel toutes les opérations seront effectuées.

4. 1ème décomposition

a. Entrée des données

- Ø Affichage d'un diagramme de Clapeyron
- Ø Fenêtre pour l'entrée des 8 valeurs de p et V.
- Ø Entrée de n

b. Traitement des données

- Ø Affichage du diagramme et collecte des valeurs : on créera 2 scripts (fonction sans variables d'Entrée/Sortie), qu'on appellera fonct pour le diagramme et inter pour la fenêtre d'entrée des valeurs p/V.

Ø Pour un travail d'isotherme, d'un état 1 à un état 2

$$W = p1V1.\int_1^2 \frac{dV}{V}$$

$\int_1^2 \frac{dV}{V}$ nécessitera l'appel à la méthode de simpson.

On appelle pas une fonction globale pour le travail d'une isotherme, qui elle-même appellerait la fonction Simpson. En effet, cela pourrait trop ficeler le problème.

Ø Pour un travail d'adiabatique, entre état 1 et un état 2

$W = \frac{C_v}{R}(p2V2 - p1V1)$ avec Cv/R=5/2 pour un gaz parfait diatomique.

La fonction correspondante aura besoin de p1, p2, V1 et V2. On l'appellera adiab.

Ø Fonction Simpson

Elle a besoin des bornes a et b, de n, et de la fonction correspondante f(x). On l'appellera simpsonV.

Ø Fonction… fonction !

On crée une fonction qui calculera pour tout x son image par f, en l'occurrence 1/x pour nous. On l'appellera inV, référence à la fonction inverse, et au traitement de Volumes.

c. Affichage du résultat

Afficher les 4 travaux intermédiaires, et le travail total.

5. 2ème décomposition

¤ **fonction simpsonV(a,b,n)**

s'assurer que n est pair : n=2n
calcul du pas d'intégration h=(b-a)/n

$\int_a^b f(x) = \frac{h}{2}\left[f(a) + f(b) + 4\sum_{i=1}^{n-1} f(i) + 2\sum_{i=2}^{n-2} f(i) + \right]$ d'après Simpson.

Pour la différenciation des indices pairs et impairs, on calculera le terme par une boucle pour de 1 à n-3 par pas de 2. On ajoutera ensuite le terme f(a+(n-1)*h) en dehors de la boucle, car il ne peut être calculé sans autres termes dans les bornes choisies.

¤ **fonction inV(x)**

à chaque x elle renvoie 1/x.

¤ **fonction adiab(p1,V1,p2,V2)**

elle calcule $W = \frac{C_v}{R}(p2V2 - p1V1)$ avec Cv/R=2.5.

¤ **script fonct**

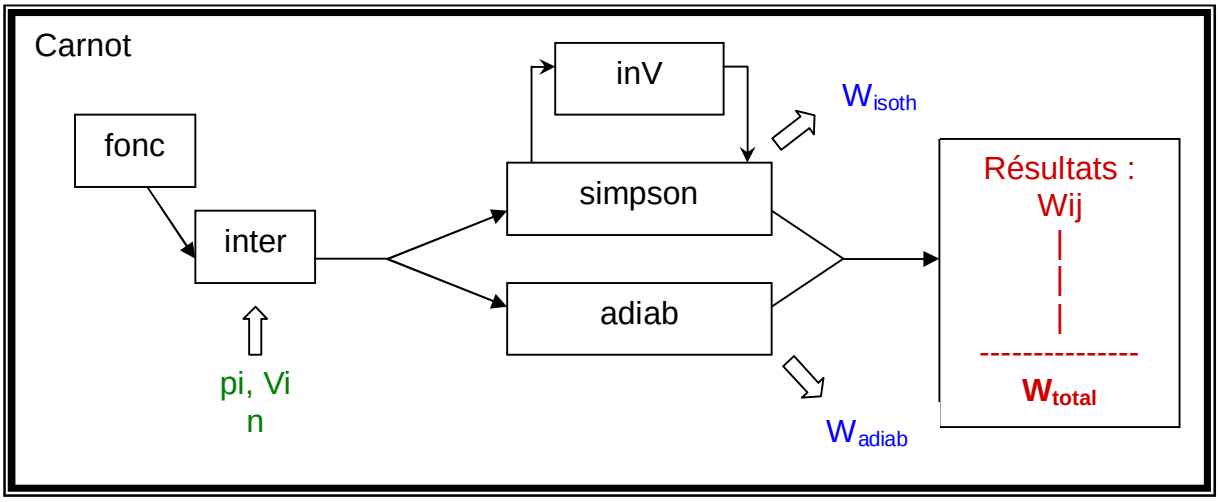
Afficher un graphique avec 2 fonctions en a/x, et deux en a/x^n. Symboliser les 4 états, légender, titrer.

¤ **script inter**

ouvrir une fenêtre demander les 4 pressions et les 4 volumes, en précisant les unités demandées.

6. Structure et algorithmes

Voici la structure du programme global :



Voici les algorithmes des différentes fonctions :

¤ **fonction inV**

```
fonction inV(x réel)=res
        res=1/x
        retourne res
fin
```

¤ **fonction simpson**

```
fonction simpson (a réel,b réel,n entier)=I réel

    variables locales somme réel, h réel
    n=2n
    h=(b-a)/n
    somme=inv(a)+inv(b)
    pour i de 1 à n-3 par pas de 2
          somme = somme +4.inV(a+ih)+2.inV(a+(i+1)h)
```

```
    fin pour
    somme = somme + f(a+(n-1)h)
    I=h/3.somme
    retourne I
fin
```

¤ fonction adiab

```
fonction adiab(p1,V1,p2,V2   réels)=w réel
        w=2.5.(p2V2-p1V1)
        retourne w
fin
```

¤ script Carnot

```
script Carnot
        appel fonct
        appel inter
        écrire « Entrer n »
        lire n
        Wab=-pAVA.simpson(VA,VB,n)
        Wbc=adiab(pB,VB,pC,VC)
        Wcd=-pCVC.simpson(VC,VD,n)
        Wda=adiab(pD,VD,pA,VA)
        écrire « Les valeurs des travaux sont » Wab Wbc Wcd Wda
        écrire « Le travail total est » Wab + Wbc + Wcd + Wda
fin
```

¤ script fonct

```
script fonct
        tracer y1=10/x, y2=5/x, y3=25/x^3, y4=25/x^3
        axes « volume », « pression »
        légende y1 : « isotherme », y3 :« adiabatique »
fin
```

¤ script inter

```
script inter
        ouvrir fenêtre (8 champs)
        pour champ de 1 à 4 écrire pA, pB, pC, pD (en bars)
                Lire pA, pB, pC, pD
                pour i (A,B,C,D)
        p(i)=100000.p(i)
        pour champ de 5 à 8 écrire VA, VB, VC, VD (en m^3)
                Lire VA, VB, VC, VD
        retourne pA, pB, pC, pD, VA, VB, VC, VD
fin
```

remarque : les scripts **fonct** et **inter** faisant appel aux possibilités particulières de Matlab, les algorithmes sont peu explicites, en comparaison avec les scripts en langage Matlab correctement commentés (voir 8.b et 8.c)

7. Vérifications de la méthode numérique

On se propose de vérifier l'exactitude de la méthode numérique utilisée, la méthode de Simpson.

On compare pour différents nombres d'intervalles les résultats fournis par la fonction simpsonV, et la fonction de matlab log(x), logarithme néperien (résultat naturel de la primitive de 1/V).

On calcule les résultats entre 1 et 10000, par pas de 10, on prendra l'écart maximum pour juger.

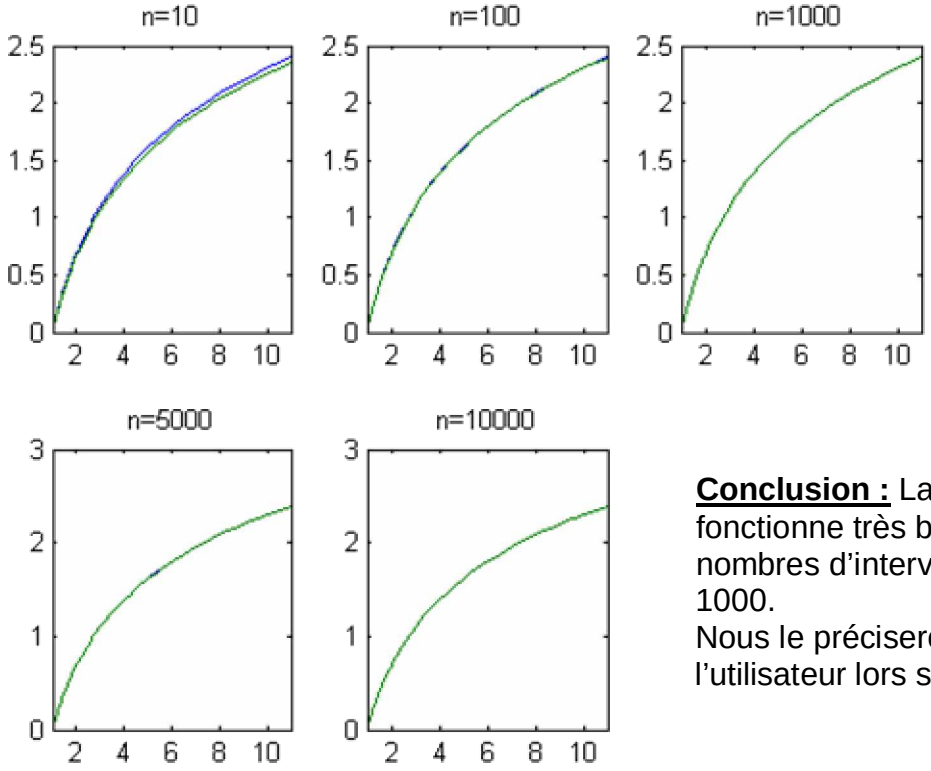
Voici les lignes de codes entrées, et les résultats :

Script des commandes :

```
% Test de la fonction simpsonV

x=[1:10:10000]; % Intervalle d'étude
n=[10,100,1000,5000,10000]; % Vecteur des différents pas à tester
for i=1:5
   M(i)=abs((log(x)-simpsonV(1,x,n(i)))/log(x)); % Calcul du pourcentage
d'écart pour chaque valeur de x
   disp(['Pour n=',num2str(n(i)),'
max(écart)=',num2str(100*max(M(i))),'%']) % Affichage de l'écart maximal
   m(i)=max(M(i))*100;
end

% Affichage des superpositions des courbes entre valeur réelles et
valeurs
% calulées par la méthode de Simpson
figure
subplot(2,3,1), fplot('[log(x) ,
simpsonV(1,x,10)]',[1:10:10000]),title('n=10')
subplot(2,3,2), fplot('[log(x) ,
simpsonV(1,x,100)]',[1:10:10000]),title('n=100')
subplot(2,3,3), fplot('[log(x) ,
simpsonV(1,x,1000)]',[1:10:10000]),title('n=1000')
subplot(2,3,4), fplot('[log(x) ,
simpsonV(1,x,5000)]',[1:10:10000]),title('n=5000')
subplot(2,3,5), fplot('[log(x) ,
simpsonV(1,x,10000)]',[1:10:10000]),title('n=10000')
```

Aperçu du Command History sous Matlab :

```
>>testsimpson

Pour n=10 max(écart)=1005.311904%
Pour n=100 max(écart)=77.664207%
Pour n=1000 max(écart)=2.312824%
Pour n=5000 max(écart)=0.039826%
Pour n=10000 max(écart)=0.003471%
```



Conclusion : La fonction Simpson fonctionne très bien pour des nombres d'intervalles supérieurs à 1000.
Nous le préciserons donc à l'utilisateur lors se son choix pour n.

8. Listing du Programme

a. Carnot.m

```
% Carnot.m Script Principal

% Ce Programme permet à l'utilisateur d'obtenir le travail échangé avec
le
% Milieu Extérieur par une machine thermique faisant suivre à un gaz
% parfait diatomique un cycle de Carnot.

% Présentation du programme à l'utilisateur
disp('                        ]>Carnot<[')
disp('Programme calculant le travail fournit par une machine suivant un
cycle')
disp('de carnot, avec une mole d''un gaz diatomique considéré comme
parfait.')
disp('Un cycle de Carnot comporte deux transformations isothermes, et
deux')
disp('adiabatiques.')
disp('Vous allez pouvoir entrer vos données en suivant le diagramme ')
disp('pression-température du cycle de carnot qui vous sera présenté.')
disp('Pressez une touche pour continuer');
pause

% Appel du diagramme et de la fenêtre d'entrée des valeurs
fonct;
inter;

% Entrée du nombre d'intervalle souhaité pour la méthode de Simpson
n=input('Entrez le nombre d''intervalles :  conseillé >1000  ');

% Calcul des travaux, par les fonctions simpsonV et adiab
```

```
Wab=-pA*VA*simpsonV(VA,VB,n);
Wbc=adiab(pB,VB,pC,VC);
Wcd=-pC*VC*simpsonV(VC,VD,n);
Wda=adiab(pD,VD,pA,VA);

% Calcul du Travail total
Wcycle=Wab+Wcd+Wbc+Wda;

% Présentation des travaux intermédiaires à l'utilisateur
M=[Wab Wbc Wcd Wda];
disp(' ')
disp('Voici les valeurs des travaux intermediares, en J')
disp('    W(a=>b)   W(b=>c)   W(c=>d)   W(d=>a)')
disp([M])

% Présentation du travail total
disp(['Le travail total échangé avec le milieu extérieur est
',num2str(Wcycle),' J']);
```

b. fonct.m

```
% Affichage d'un diagramme p.V  d'un cycle de carnot, qui guide
% l'utilisateur pour entrer ses valeurs de pression et de volume.
clear

% Fond du graphique en blanc
figure('Color',[1 1 1]);

% Blocage de la fenêtre graphique
hold on

% Définition du titre et des légendes des axes
title('Diagramme de Clapeyron du cycle de Carnot','color','r')
xlabel('Volume')
ylabel('Pression')

% Tracé des courbes isothermes et des adiabatiques
fplot('10/x',[1,4,1,7],'r')
fplot('25/x^3',[1,4,1,7],'b')
fplot('5/x',[1,4,1,7],'r')
fplot('70/x^3',[1,4,1,7],'b')

% Pointage des points A, B, C et D sur le diagramme
text(1.6,6.5,'A')
text(2.7,4,'B')
text(3.8,1.5,'C')
text(2.1,2.0,'D')

% Symbolisation de la valeur du Travail fourni par le cycle
text(2.1,3.5,'|W(cycle)|')

% Affichage de la légendes des courbes
legend('isotherme','adiabatique')
```

c. inter.m

```
% Script ouvrant une boîte de dialogue, organisant l'entrée des
% valeurs de pression et de volume. L'utilisateur est guidé
% par le diagramme de Clapeyron qui lui est présenté juste
% auparavant.
```

```
% Ouverture de la boîte de dialogue, définition des champs
prompt={'pA',['pB'],['pC'],['pD'],['VA'],['VB'],['VC'],['VD']};

% Définition du titre de la boîte de dialogue
title=['Veuillez entrer...'];

% Definition des valeurs par défaut de chaque champ, on y affiche les
% unités des valeurs désirées
defAns={'(en Bar)','(en Bar)','(en Bar)','(en Bar)','(en m^3)',...
    '(en m^3)','(en m^3)','(en m^3)'};

% Champ réglé à une seule ligne
lineNo=1;

% Création d'un matrice reponse qui contiendra les
% valeurs entrées par l'utilisateur
reponse = inputdlg(prompt,title,lineNo,defAns);

% Assignation au variables des valeurs données par
% l'utilisateur.
pA=str2num(reponse{1})*100000; % On effectue la conversion Bars/Pascals
pB=str2num(reponse{2})*100000;
pC=str2num(reponse{3})*100000;
pD=str2num(reponse{4})*100000;
VA=str2num(reponse{5});
VB=str2num(reponse{6});
VC=str2num(reponse{7});
VD=str2num(reponse{8});
```

d. inV.m

```
function res=inV(x)
res=1./x;
end
```

e. simpsonV.m

```
function I=simpsonV(a,b,n)

% simpsonV renvoie le résultat de l'intégrale _/dV/V, entre a et b

n=2*n; %pour avoir un nb de pas pair
h=(b-a)/n; % longueur de l'intervalle élémentaire

% Calcul du terme intégral partiel
somme=inV(a)+inV(b);
for i=1:2:n-3
    somme=somme+4*inV(a+i*h)+2*inV(a+(i+1)*h);
end
somme=somme+inV(a+(n-1)*h); % Addition du terme résiduel hors-boucle

% Détermination de la valeur finale de l'intégrale
I=h.*somme./3;
end
```

f. adiab.m

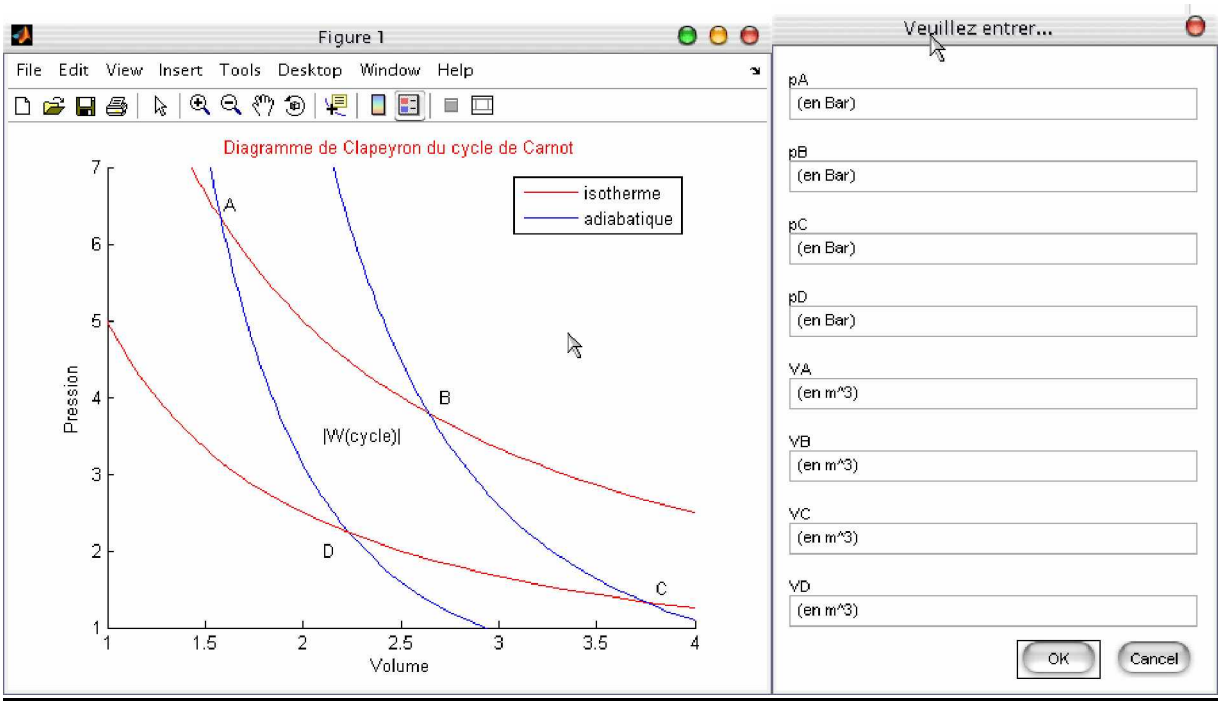
```
function w=adiab(p1,V1,p2,V2)
% La fonction calcule le travail correspondant à une transformation
```

```
% adiabatique (d'un gaz parfait diatomique), entre les états 1 et 2,
% caractérisés par leur pression p et volume V.

% W=Cv/Rx(p2*V2-p1*V1) avec Cv/r=5/2=2.5 pour un gaz parfait diatomique
w=2.5*(p2*V2-p1*V1);
end
```

9. Captures d'écran

Voici quelques captures d'écran illustrant l'interface proposée à l'utilisateur :

```
  To get started, select MATLAB Help or Demos from the Help menu.

>> Carnot
                        ]>Carnot<[
Programme calculant le travail fournit par une machine suivant un cycle
de carnot, avec une mole d'un gaz diatomique considéré comme parfait.
Un cycle de Carnot comporte deux transformations isothermes, et deux
adiabatiques.
Vous allez pouvoir entrer vos données en suivant le diagramme
pression-température du cycle de carnot qui vous sera présenté.
Pressez une touche pour continuer
```



```
>> Carnot
                        ]>Carnot<[
Programme calculant le travail fournit par une machine suivant un cycle
de carnot, avec une mole d'un gaz diatomique considéré comme parfait.
Un cycle de Carnot comporte deux transformations isothermes, et deux
adiabatiques.
Vous allez pouvoir entrer vos données en suivant le diagramme
pression-température du cycle de carnot qui vous sera présenté.
Pressez une touche pour continuer
Entrez le nombre d'intervalles :  conseillé >1000  5000

Voici les valeurs des travaux intermediares, en J
    W(a=>b)    W(b=>c)    W(c=>d)    W(d=>a)
  1.0e+006 *

   -0.5429    -1.5900     0.2289     1.4550

Le travail total échangé avec le milieu extérieur est -448935.455 J
>>
```

II. Ajustement des données expérimentales par la méthode des moindres carrés

1. Présentation du problème

Le programme consiste à effectuer un ajustement polynomial sur un ensemble de valeurs expérimentales, en utilisant la méthode des moindres carrés. Il faudra fournir à l'utilisateur les indicateurs permettant d'indiquer la qualité de l'ajustement.

L'utilisateur pourra choisir le degré du polynôme.

2. Contraintes

L'utilisateur pourra choisir le degré du polynôme.

On utilisera toutes les fonctions présentées en cours, c'est-à-dire somme1, somme2, pivot, som_expl, som_tot, som_res, moyenne, g, var_res, var_expl, détermination, facteur, varcovar. On devra utiliser de plus les fonctions matrice, permutligne et student.

3. Analyse du problème

Il nous faudra tout d'abord gérer correctement l'acquisition des séries de mesures. L'acquisition des données au format texte nous a été présentée en cours. Il s'avère qu'à l'usage, ce format est peu souple. On pensera donc à gérer aussi l'acquisition de données au format Excel, tout en donnant le choix du format à l'utilisateur.

L'ajustement des mesures par les méthodes des moindres carrés nous amène à la gestion de matrices, représentant les équations des systèmes. Il nous faudra résoudre ce système en utilisant la méthode du pivot de Gauss.

Cela impliquera donc la création de toutes les matrices nécessaires, dont la matrice M, contenant à la fois les valeurs x et y, ainsi que le degré du polynôme que l'utilisateur aura choisi. C'est cette matrice qui sera traitée par la méthode du pivot. Une fonction pivot correspondante nous fournira les différents coefficients de l'ajustement.

L'utilisateur devra donc disposer de ces coefficients, ainsi que des différents indicateurs reflétant la qualité de l'ajustement (r^2, F). On lui donnera aussi la possibilité d'avoir des informations sur les incertitudes affectées aux différents coefficients. On lui donnera aussi la possibilité d'obtenir le coefficient de corrélation entre n'importe quels coefficients du polynôme.

4. 1ème décomposition

a. Entrée des données

- Ø Acquisition des valeurs expérimentales sous .txt ou .xls
- Ø Entrée du degré du polynôme

- Ø Opérations d'entrée sortie, type chaînes de caractères

b. Traitement des données

- Ø Détermination du nombre N de mesures
- Ø Création de la matrice M
- Ø Calcul de y moyen, de ymc calculés par la méthodes des moindres carrés
- Ø Application de la méthode du pivot sur cette matrice
- Ø Récupération des coefficients dans un vecteur
- Ø Calcul des trois sommes relatives aux écarts
- Ø Calcul de r^2, F, incertitudes sur les a(i), et calculs des coefficients de corrélation (matrice variance-covariance, coefficient de Student).

c. Affichage du résultat

- Ø Afficher les différents coefficients du polynôme
- Ø Affichage de r^2, de F.
- Ø Proposition d'affichage des incertitudes sur les coefficients
- Ø Proposition d'affichage des coefficients de corrélation entre indices des coefficients choisis par l'utilisateur.

5. 2ème décomposition

¤ Script **regr**

- Ø Affichage de la présentation du programme
- Ø Demande du format d'éntrée
  - .txt => première procédure
  - .xls => seconde procédure
- Ø Demande du degré du polynôme
- Ø Création du vecteur A par la fonction pivot : fonction *pivot*
- Ø Calcul de ym (y moyen) : fonction *moyenne*
- Ø Calcul de ymc (y par la méthodes des moindres carrés) ; fonction *g*
- Ø Affichage des courbes : script *graphiques*
- Ø Calcul des trois sommes relatives aux écarts : fonctions *som_expl*, *som_tot*, *som_res*.
- Ø Calcul de r^2 : fonction *determination*
- Ø Calcul de F : fonction *facteur*
- Ø Affichage des coefficients
- Ø Proposition d'informations sur les coefficients : fonction *varcovar*

¤ Fonction **pivot**

- Ø Création de la matrice M : fonction *matrice*

boucle pour i=1 jusqu'à r+1 (décrit toutes les lignes)

Vérification de la valeur du pivot
test si |pivot|<=eps
signal d'un pb sur le pivot

repérage de la ligne i posant problème, par la valeur indice
recherche d'un pivot plus adéquat sur une ligne comprise de i+1 à r+1
boucle tant que (2) (|pivot(indice,i)|<=eps)&&(indice<r+2)
indice++
on utilise un drapeau pour savoir l'état du pivot
si indice ==r+2 = test=0
sinon, test=1, permutation des lignes i et indice, pour travailler sur un pivot suffisamment grand.

si test==1 on a dispose d'un pivot suffisamment grand, application de la méthode de Gauss.
boucle pour de j=i+1 jusqu'à r+2 (toutes les colonnes après celle en cours)
division des indices M(i,j) par M(i,i) (le pivot)
M(i,i) mis à 1

boucle pour de i=1 à r+1 toutes les lignes SAUF celle du pivot (boucle si)
coeff=M(indice,i) on sauvegarde la valeurs du terme qu'on va ensuite annuler
M(indice,i)=M(indice,i)-coeff*M(i,i) La colonne est mise à 0, sauf le pivot, toujours à 1.

boucle si test==1
le vecteur **A** prend les valeurs de la dernière colonne de M

si toutes les boucles sur test sont négatives, on affiche un message d'erreur.

¤ fonction **matrice**
premier terme (1,1) mis à N
boucle pour de 1=2 jusqu'à r+1
chaque terme prend la valeur désirée par la fonction *somme1*
on repère la dernière puissance de la colonne par la valeur max
copie suivant la symétrie diagonale sur le colonne suivante
Le dernier terme de la colonne prend la valeur de somme1, mais à la puissance max.
La dernière colonne est traitée par la fonction *somme2*

¤ fonction **permutligne**
boucle pour sur toutes les colonnes compteur de 1 à r+2.
stockage dans la valeur aux du terme M(i,compteur)
copie du terme M(j,compteur) dans M(i,compteur)
copie de aux dans M(j,compteur)

¤ fonction **som_expl**
Boucle pour de i=1 à N du terme de la somme des (ymc(i)-ym)^2
Stocké sous Se

¤ fonction **som_res**

Boucle pour de i=1 à N du terme de la somme des (y(i)-ymc(i))^2
Stocké sous Sr

¤ fonction **som_tot**

Boucle pour de i=1 à N du terme (y(i)-ymc(i))^2
Stocké sous St

¤ fonction **somme1**

Boucle pour de i=1 à N pour le calcul de la somme des x(i)^r

¤ fonction **somme2**

Boucle pour de i=1 à N pour le calcul de la somme des y(i).x(i)^r

¤ fonction **g**

Boucle pour de i=1:r+1 de la somme des A(i).x^(i-1)

¤ fonction **determination**

Calcul de Se/St
Stocké sous r2

¤ fonction **var_res**

Calcul de Sr/(N-r-1)
Stocké sous vr

¤ fonction **var_expl**

Calcul de Se/r
Stocké sous ve

¤ fonction **facteur**

Calcul de ve/vr
Stocké sous F

¤ fonction **varcovar**

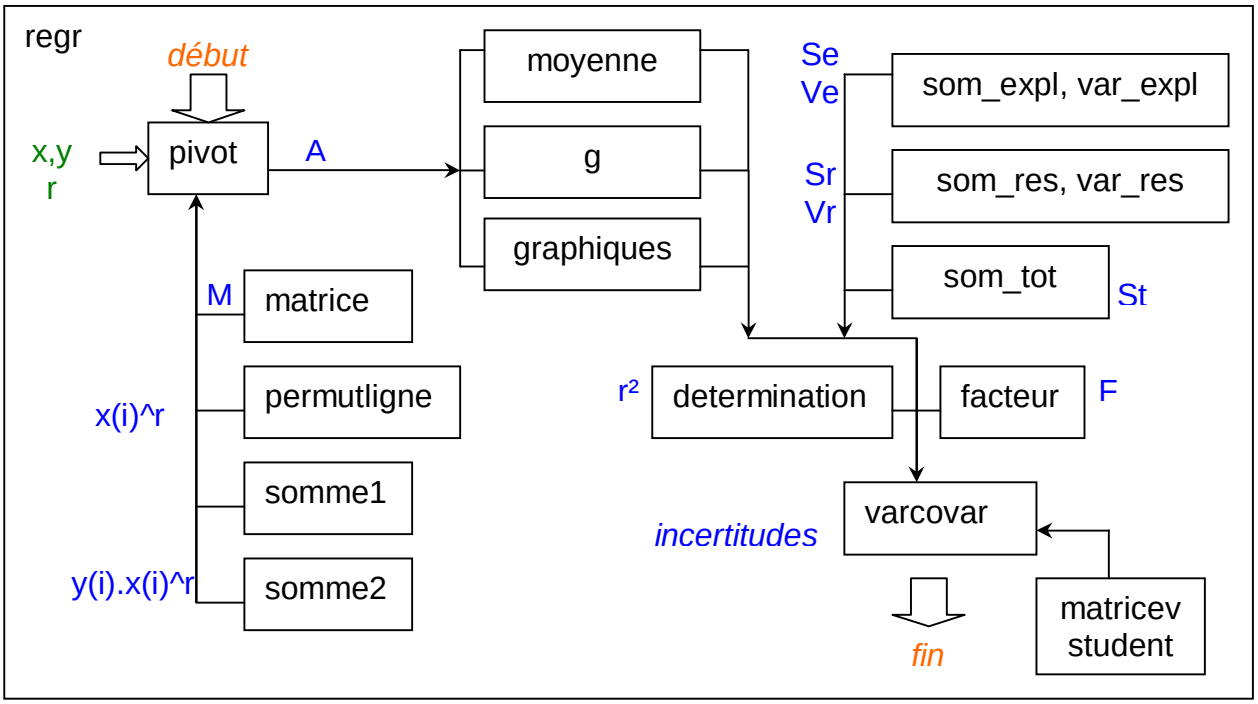
création d'une Matrice M, carré, n'ayant pas les termes y.X^r dans sa dernière colonne. fonction *matricev*
calcul de la matrice variance-covariance $V=Vr.M^{-1}$
détermination du coefficient de Student en fonction du nombre de mesures N
Calcul de l'écart type sur chaque élément i sigma(i)=V(i,i)^0.5
Affichage de la chaîne de caractère donnant le coefficient a(i) +/- t.sigma(i)%
Détermination du coefficient de corrélation sur chaque paramètre, selon les indices voulus par l'utilisateur.

¤ script **graphiques**

affichage des courbes expérimentales, théoriques, et de la superposition des deux.

6. Structure et algorithmes

Voici la structure du programme :



¤ Script **regr**

```
début
    écrire « Ajustement polynomial »
    écrire « Quel format ? (texte ou excel) t/e »
        lire rep
                si rep==t
                        data = acquérir valeurs.txt
                sinon
                        data = acquérir valeurs.xls
                fin si
    x=data(:,1) ; y=data(:,2) ;
    N=taille(data,1)
    écrire « Entrez le degré du polynôme souhaité »
        lire r
    A=pivot(x,y,N,r)
    écrire A
    ym=moyenne(y,N)
    ymc=g(A,x,r)
    Se=som_expl(ymc,ym,N)
    St=som_tot(y,ym,N)
    Sr=som_res(y,ymc,N)
    r2=determination(Se,St)
    F=facteur(vr,ve)
    écrire « r²= ,r2 »
    écrire « F= ,F)
```

```
        lancer graphiques
        écrire « voulez-vous avoir des informations sur les coefficients ? o/n »
            lire rep
            si rep == o lancer varcovar(x,N,r,Sr,A)
    fin si
    fin
```

¤ fonction **pivot**

```
    fonction pivot (x vect, y vect, r entier, N entier) = A vect
            début
                    variables locales (indice entier, test entier, coeff réel)
      test=1
      M=matrice(x,y,N,r)
      pour i=1 à r+1
            si |M(i,i)|<=eps
            alors   indice=I

                    tant que (|M(I,i)|<=eps)&&(indice<r+2)
                            indice++
                    fin tant que

                    si indice==r+2
                    alors test=0
                    sinon M=permutligne(r,I,indice,M)
                    fin si
            fin si

            si test==1
                    pour j=i+1 à r+2
                            M(i,j)/m(i,i)
                    fin pour

                    M(i,i)=1

                    pour indice= 1 à r+1
                            si indice≠i
                            alors  coeff=M(indice,i)
                                    pour j=1 à r+2
                                            M(indice,j)=M(indice,j)-coeffxM(i,j)
                                    fin pour
                            fin si
                    fin pour
            fin si
      fin pour

      si test==1
            pour i=1 à r+1
                    A(i)=M(i,r+2)
```

```
            fin pour
        fin si
        retourne A
```

¤ fonction **matrice**

```
fonction matrice(x vect, y vect, N entier, r entier)=M matrice
début
        variables locales i entier, j entier, max entier

        M(1,1)=N
        Pour i= 2 à r+1
                M(i,1)=somme1(x,N,i-1)
        fin pour

        max=r

        Pour j=2 à r+1
                Pour i=1 à r
                        M(I,j)=M(i+1,j-1)
                fin pour

                max++
                M((i+1),j)=somme(x,N,max)
        fin pour

        Pour i=1 à r+1
                M(I,r+2)=somme2(x,y,N,i-1)
        fin pour
retourne M
```

¤ fonction **permutligne**

```
        fonction permutligne(r entier, i entier, j entier,M matrice)=M matrice
début
        variables locales : compteur entier, aux réel

                pour compteur = 1 à r+2
                        aux=M(i,compteur)
                        M(i,compteur)= M(j,compteur)
                        M(j,compteur)=aux
                fin pour
        retourne M
fin
```

¤ fonction **som_expl**

```
        fonction som_expl(ymc vect,ym réel,N entier)
début
        variables locales : i entier
        Se=0
```

```
        pour i= 1 à N
                Se=Se + (ymc(i)-ym)^2
        fin pour
        retourne Se
fin

¤ fonction som_res
        fonction som_expl(y vect,ymc vect,N entier)
début
        variables locales : i entier
        Sr=0
        pour i= 1 à N
                Sr=Sr + (y(i)-ymc(i))^2
        fin pour
        retourne Sr

fin

¤ fonction som_tot
        fonction som_expl(y vect,ym réel,N entier)
début
        variables locales : i entier
        St=0
        pour i= 1 à N
                St=St + (y(i)-ym))^2
        fin pour
        retourne St

fin


¤ fonction somme1

         fonction somme1(N entier, r entier, x vect)=res réel
début
        variables locales i entier
                res=0
                pour i=1 à N
                        res=res+x(i)^r
                fin pour
        retourne res

fin

¤ fonction somme2

         fonction somme1(N entier, r entier, x vect, y vect)=res réel
début
        variables locales i entier
                res=0
```

```
            pour i=1 à N
                res=res+x(i)^r.y(i)
            fin pour
        retourne res

fin
```

¤ fonction **g**

```
        fonction g(A vecteur, x vect, r entier)=ymc vect
début
        variables locales i entier

ymc=0
pour i=1 à r+1
   ymc=ymc+A(i).x^(i-1);
fin pour
fin
```

¤ fonction **détermination**

```
        fonction détermination(Se réel, St réel)=r2 réel
début
        variables locales aucune

        r2=Se/St
        retourne r2

fin
```

¤ fonction **var_res**

```
        fonction var_res(Sr réel, N entier, r entier)=vr réel
début
        variables locales
        vr=Sr/(N-r-1)
        retourne vr

fin
```

¤ fonction **var_expl**

```
        fonction var_expl(Se réel, r entier)=vr réel
début
        variables locales
        ve=Se/r
        retourne ve

fin
```

¤ fonction **facteur**

```
        fonction var_expl(vr réel, ve réel)=vr réel
début
        variables locales
        F=ve/vr
        retourne F

fin
```

¤ fonction **varcovar**

```
        fonction var_expl(x vect, N entier, r entier, Sr réel, A vect)
début
        variables locales :
M matrice, V matrice, t réel, i entier, sigma réel, rep char, ans char
        M=matricev(x,N,r)
V=Sr.M^-1/(N-r-2)
t=student(N,r)
pour i= 1 à r+1
        sigma(i)=[V(i,i)]^1/2
fin pour
var=caractère(A(1))

pour i=1 à r+1
        varp=caractère(+ x^caractère(i).caractère(A(i)) +/- caractère(t.sigma(i,i))%)
        var=concaténation(var,varp)
fin pour

écrire « Incertitudes sur les coefficients : var »
écrire « Voulez vous tester l'indépendance entre les coefficients ? o/n »
        lire rep

ans=o
tant que ((rep==o)&&(ans=o))
        écrire « indice du premier coefficient » lire i
        écrire « indice du premier coefficient » lire i
        rc=V(i+1),j+1)/V(i+1,i+1)/[V(i+1,i+1).V(i+1,j+1)]^1/2
        écrire « le coefficient de corrélation est rc »
        écrire « voulez vous tester l'indépendance entre d'autres coefficients ?  o/n»
        lire ans
fin tant que

fin
```

¤ fonction **matricev**

```
fonction matricev(x vect, N entier, r entier)=M matrice
début
        variables locales i entier, j entier, max entier
```

```
        M(1,1)=N
        Pour i= 2 à r+1
                M(i,1)=somme1(x,N,i-1)
        fin pour

        max=r

        Pour j=2 à r+1
                Pour i=1 à r
                        M(I,j)=M(i+1,j-1)
                fin pour

                max++
                M((i+1),j)=somme(x,N,max)
        fin pour

retourne M
```

¤ fonction **student**

```
fonction student(N entier, r réel)=t réel
début
        variables locales : aucune

            S(1)=12.706     S(11)=2.201     S(21)=2.080
            S(2)=4.303      S(12)=2.179     S(22)=2.074
            S(3)=3.181      S(13)=2.160     S(23)=2.069
            S(4)=2.776      S(14)=2.145     S(24)=2.064
            S(5)=2.571      S(15)=2.131     S(25)=2.060
            S(6)=2.447      S(16)=2.120     S(26)=2.056
            S(7)=2.365      S(17)=2.110     S(27)=2.052
            S(8)=2.306      S(18)=2.101     S(28)=2.048
            S(9)=2.262      S(19)=2.093     S(29)=2.045
            S(10)=2.228     S(20)=2.086     S(30)=2.042
                                            S(31)=1.960

        si (N-r-1)<31
                t=S(N-r-2)
        sinon
                t=S(31)
        fin si
        retourne t

fin
```

¤ script **graphiques**

```
début
        tracer y en fonction de x, titre « Courbe expérimentale »
```

tracer ymc en fonction de x, titre « Courbe ajustée »
tracer y et ymc en fonction de x, titre « Courbes superposées »
fin

7. Vérification de la méthode numérique

On choisit de tester la véracité de la fonction pivot, donnant les coefficients d'un polynôme dont le degré est choisi. Voici les résultats en fonction des valeurs entrées :

remarque : on a tout d'abord choisi les valeurs suivant l'équation $y=2-5x+x^2$, puis des valeurs « bruitées », pour simuler des mesures réelles.

0	2
1	-2
2	-4
3	-4
4	-2
5	2
6	8
7	16
8	26
9	38

M =

1	0	0	2
0	1	0	-5
0	0	1	1

0	1,9
1	-2,2
2	-3,9
3	-4
4	-1,9
5	1,95
6	8,1
7	15,8
8	25,7
9	38,2

M =

1	0	0	1.9109
0	1	0	-4.9688
0	0	1	0.9970

On voit que la dernière colonne porte bien les coefficients, et que la matrice est unitaire sur ses trois premières colonnes.

8. Listing du Programme

a. regr.m

```
disp(' ')
disp('                ]>AJUSTEMENT POLYNOMIAL<[')
disp('  ')
disp('Forme canonique : a0+a1.x^1+a2.x^2+...+an.x^n')
disp(' ')
disp('Ce programme vous permet d''importer des donner aux formats texte')
disp('ou excel. Créez un fichier nommé valeurs.(extension : .txt/.xls)')
disp('dans la racine du disque c (c:\valeurs.txt/xls).')
disp(' ')
r=input('Entrez le degré du polynôme pour l''ajustement polynomial : ');
```

```
rep=input('Quel format de fichier voulez vous lire ? texte (.txt) ou
excel (.xls) ?  (t/e)  ','s');

if rep=='t' % Procédure fichier texte

    % Ouverture du fichier de valeurs expérimentales
    fid=fopen('c:\valeurs.txt','r');
    data=fscanf(fid,'%f%f',[2 inf]);
    fclose(fid);

    % Transposée de la matrice data pour les traitements qui suivent
    data=data';

    % Détermination du nombre de valeurs
    N=size(data,1);

    % Création des vecteurs x et y
    x=data(:,1);
    y=data(:,2);
else % Procédure fichier Excel
    data = xlsread('c:\valeurs'); % data prend les valeurs du fichier
excel
    x=data(:,1); % Création des vecteurs x et y
    y=data(:,2);
    N=size(data,1); % Détermination du nombre de valeurs
end

% Création du vecteur A contenant les coefficients du polynôme
A=pivot(x,y,r,N);

% Création du chaîne de caractère "pol", qui contiendra l'expression de
% coefficients du polynôme et les x à la puissance correspondante.
pol=(['y = ',num2str(A(1))]); % Début de la chaîne de caractère, pour
tous les cas.
for i=1:r
    if A(i+1)<0 % But de la boucle : éviter d'afficher " + -a(i)"
        pui=([' ',num2str(A(i+1)),'.x^',num2str(i)]); % Chaîne tampon
"pui", contenant le terme à ajouter, pour coefficient négatif
    else
        pui=([' +',num2str(A(i+1)),'.x^',num2str(i)]); % Chaîne tampon
"pui", contenant le terme à ajouter, pour coefficient positif
    end
    pol=strcat(pol,pui); % Concaténation de "pui" à la suite de "pol"
end

disp(' ')
disp('Résultats de la modélisation :')
disp(pol)

% Calcul de ym (y moyen)
ym=moyenne(y,N);

% Calcul de ymc (y moindres carrés) de l'ajustement pour chaque valeurs
expérimentales x
ymc=g(A,x,r);

% Calcul des trois sommes relatives aux écarts entre valeurs réelles,
% mesurées et calculées.
Se=som_expl(ymc,ym,N);
St=som_tot(y,ym,N);
Sr=som_res(y,ymc,N);
```

```
disp('  ')
r2=determination(Se,St);
% Affichage des coefficients montrant la qualité de l'ajustement
disp(['Le coefficient de détermination r² est ',num2str(r2),'.'])

vr=var_res(Sr,N,r);
ve=var_expl(Se,r);
F=facteur(vr,ve);

disp(['Le facteur de qualité F est ',num2str(F),'.'])
disp(' ')

% Affichage des courbes expérimentales et ajustées
graphiques

% Proposition d'aide sur les facteurs de qualité
rep=input('Voulez-vous de l''aide sur ces coefficients ? o/n ','s');
if isempty(rep),rep='n'; end % Taper entrée revient à dire non. On
considère l'aide comme secondaire
if rep=='o'
    disp('Plus l''ajustement est correct, plus r² tend vers 1, et plus F
tend vers l''infini.')
end
disp(' ')

% Proposition d'une étude statistique
rep=input('Désirez vous connaître les incertitudes sur les coefficients ?
o/n ','s');
if isempty(rep),rep='o'; end % Taper entrée revient ici à dire oui.
if rep=='o'
    varcovar(x,N,r,Sr,A)
end
```

b. pivot.m

```
function A=pivot(x,y,r,N)

% Cette fonction suit la méthode du pivot de Gauss, afin de déterminer
les
% coefficients a(i) de la modélisation.
% Elle prend en charge la gestion d'un pivot trop faible.

test=1;
M=matrice(x,y,N,r);
for i=1:r+1
    % Vérification de la valeur du pivot
    if (abs(M(i,i))<=eps)
        disp('Problème sur le pivot, possibilité de permuter deux lignes
en cours d''étude...')
        indice=i;

        % Recherche d'un pivot plus adéquat
        while ((abs(M(indice,i))<=eps)&&(indice<r+2))
            indice=indice+1;
        end

        % Recherche négative après vérification de toutes les lignes
        if (indice==r+2)
            test=0;
        else
```

```
            disp('permutligne')
            M=permutligne(r,i,indice,M);  % Permutation de la ligne avec
celle du pivot trouvé
        end
    end

    % Méthode de Gauss sur la matrice

    if (test==1)
        for j=i+1:r+2
            % Division de la ligne par le pivot
            M(i,j)=M(i,j)/M(i,i);
        end
        M(i,i)=1;
        % Pivot mis à 1
        for indice=1:r+1
            if (indice~=i)
                coeff=M(indice,i);
                for j=1:r+2
                    % Annulation des termes autres que le pivot sur la
colonne
                    M(indice,j)=M(indice,j)-coeff*M(i,j);
                end
            end
        end
    end

end

% Création de la matrice A, contenant les coefficients du polynôme
if (test==1)
    for i=1:r+1
        A(i)=M(i,r+2);
    end
else
    disp('Erreur de précision sur le pivot')
end

end
```

c. matrice.m

```
function M=matrice(x,y,N,r)

% Cette fonction génère la matrice M, comportant les termes expérimentaux
x
% et y, qui est destinée au traitement par la fonction pivot.

M(1,1)=N; % Initialisation de la première valeur de la matrice

for i=2:r+1
    M(i,1)=somme1(i-1,N,x); % Création de la première ligne
end
max=r; % Repérage de la dernière puissance de la colonne
for j=2:r+1
    for i=1:r
        M(i,j)=M(i+1,j-1); % Copie de la première ligne en respectant la
    end                    % symétrie de la matrice par rapport à la
diagonale
    max=max+1; % Repérage de la dernière puissance de la colonne
```

```
    M(i+1,j)=somme1(max,N,x); % Valeur du dernier élément de la colonne
traitée
end

for i=1:r+1
    M(i,r+2)=somme2(i-1,N,x,y); % Traitement de la dernière colonne de la
matrice
end
end
```

d. permutligne.m

```
function M=permutligne(r,i,j,M)
% Permute les lignes i et j de la matrice M à r+2 colonnes.

for compteur=1:r+2
    aux=M(i,compteur); % Stockage de la première valeur à déplacer
    M(i,compteur)=M(j,compteur); % Ecrasement de la première valeur par
la deuxième
    M(j,compteur)=aux; % Copie de la première valeur à la place de la
seconde
end

end
```

e. som_expl.m

```
function Se=som_expl(ymc,ym,N)

% Calcule la somme expliquée des écarts entre valeurs calculées et
valeur
% moyenne.

Se=0;
for i=1:N
    Se=Se+(ymc(i)-ym)^2;
end
```

f. som_res.m

```
function Sr=som_res(y,ymc,N)

% Calcule la somme résiduelle des écarts entre valeurs expérimentales et
% valeurs calculées.

Sr=0;
for i=1:N
    Sr=Sr+(y(i)-ymc(i))^2;
end
```

g. som_tot.m

```
function St=som_tot(y,ym,N)

% Calcule la somme totale des écarts entre valeurs expérimentales et
valeur
% moyenne.

St=0;
```

```
for i=1:N
    St=St+(y(i)-ym)^2;
end
```

h. somme1.m

```
function res=somme1(r,N,x);

% Génère à chaque appel la somme de 1 à N des termes x(i)^r

res=0;
for i=1:N
    res=(x(i))^r+res;
end

end
```

i. somme2.m

```
function res=somme2(r,N,x,y);

% Génère à chaque appel la somme de 1 à N des termes y(i).x(i)^r

res=0;
for i=1:N
    res=(x(i))^r*y(i)+res;
end

end
```

j. g.m

```
function ymc=g(A,x,r)
% Calcule la valeur des y calculés par la méthode des moindres carrés.

ymc=0;
for i=1:r+1
    % On prend le coefficient dans le vecteur A correspondant et on le
    % multiplie avec le vecteur x. ymc aura la même dimension que x.
    ymc=ymc+A(i).*x.^(i-1)
end
end
```

k. determination.m

```
function r2=determination(Se,St)
% Calcule le coefficient de détermination r².

r2=Se/St;
end
```

l. var_res.m

```
function vr=var_res(Sr,N,r)
% Calcul la variance résiduelle
```

```
vr=Sr/(N-r-1);
end
```

m. var_expl.m

```
function ve=var_expl(Se,r)
% Calcul la variance expliquée
ve=Se/r;
end
```

n. facteur.m

```
function F=F(vr,ve)
% Calcule le facteur F. Plus il est grand, meilleur est l'ajustement.

F=ve/vr;
end
```

o. graphiques.m

```
figure('name',[pol,' ; r²=',num2str(r2),' et F=',num2str(F)],'Color',[1 1
1]);
% La figure a le nom du polynôme trouvé par la méthode des moindres
% carrés, en précisant r² et F. Son fond est de couleur blanche,

% Commandes subplot, 4x4, on prend les blocs 2 et 4 pour la superposition
% des deux courbes.
subplot(2,2,1),plot(x,y,'bx'),grid,axis square,title('Courbe
expérimentale')
subplot(2,2,3),plot(x,ymc,'r-'),grid,axis square,title('Ajustement
polynomial')
subplot(2,2,[2 4]),plot(x,y,'bx',x,ymc,'r-'),grid on,axis
square,title('Superposition'),legend('expérimentale','ajustée')
```

p. varcovar.m

```
function varcovar(x,N,r,Sr,A)

% Donne les incertitudes sur les coefficients déterminés portés par A,
par
% l'étude statistique suivante :
% => Ecart-types déterminés par la matrice de variance-covariance
% => Incertitudes par utilisation du coefficient de Student correspondant

% Création de la matrice carrée M
M=matricev(x,N,r);

% Calcul de la matrice de Variance-Covariance
V=Sr/(N-r-2)*inv(M);

% Détermination du coefficient de Student associé à la quantité de
mesures
t=student(N,r);

% Création du vecteur qui comportera les écarts-types associés à chaque
% coefficients
```

```
for i=1:r+1
    sigma(i)=(V(i,i))^0.5;
end

% Création de la chaîne de caractères qui explicitera les incertitudes à
% l'utilisateur
var=([num2str(A(1)),' +/- ',num2str(t*sigma(1))]);
for i=2:r+1
    varp=([' + x^',num2str(i-1),' .',num2str(A(i)),' +/-
',num2str(t*sigma(i))]);
    var=strcat(var,varp);
end

disp('')
disp('En suivant la statistique de Student, on déduit les incertitudes')
disp('suivantes sur les coefficients de la modélisation :')
% affichage du polynôme corrigé des incertidudes
disp(' ')
disp(var)
disp(' ')

% Obtention des coefficients de corrélation
rep=input('Voulez vous tester l''indépendance entre des coefficients ?
o/n  ','s');
if isempty(rep),rep='o'; end % Taper entrée permet de valider, comme 'o'
ans='o'; % Mise à 'o' d'une seconde chaîne de caractères
while ((rep=='o')&&(ans=='o'))
    i=input(['indice du premier coefficient  (!) <',num2str(r+1),'  :
']); % Demande des indices des coefficients à étudier
    j=input(['indice du second coefficient  (!) <',num2str(r+1),'  : ']);
    rc=real(V(i+1,j+1)/(V(i+1,i+1)*V(i+1,j+1))^0.5); % Calcul du
coefficient de corrélation correspondant
    disp(['Voici le coefficient de corrélation entre les indices
',num2str(i),' et ',num2str(j),' : ',num2str(rc),'.'])
    disp(' ')
    ans=input('Voulez vous tester l''indépendance entre d''autres
coefficients ? o/n ','s');
    if isempty(ans),ans='o'; end

end

end
```

q. student.m

```
function t=student(N,r)

% Fournit le coefficient de Student correspondant aux nombres de valeurs
% expérimentales.
% Le coefficient corrige selon une approche statistique l'écart-type, en
% augmentant la confiance pour un grand nombre de mesure

% ddl : degrès de liberté (=N-r-1)

    S(1)=12.706; % nb de ddl=2    S(11)=2.201;    S(22)=2.074;
    S(2)=4.303;  % nb de ddl=3    S(12)=2.179;    S(23)=2.069;
    S(3)=3.181;  % ...            S(13)=2.160;    S(24)=2.064;
    S(4)=2.776;                   S(14)=2.145;    S(25)=2.060;
    S(5)=2.571;                   S(15)=2.131;    S(26)=2.056;
    S(6)=2.447;                   S(16)=2.120;    S(27)=2.052;
    S(7)=2.365;                   S(17)=2.110;    S(28)=2.048;
```

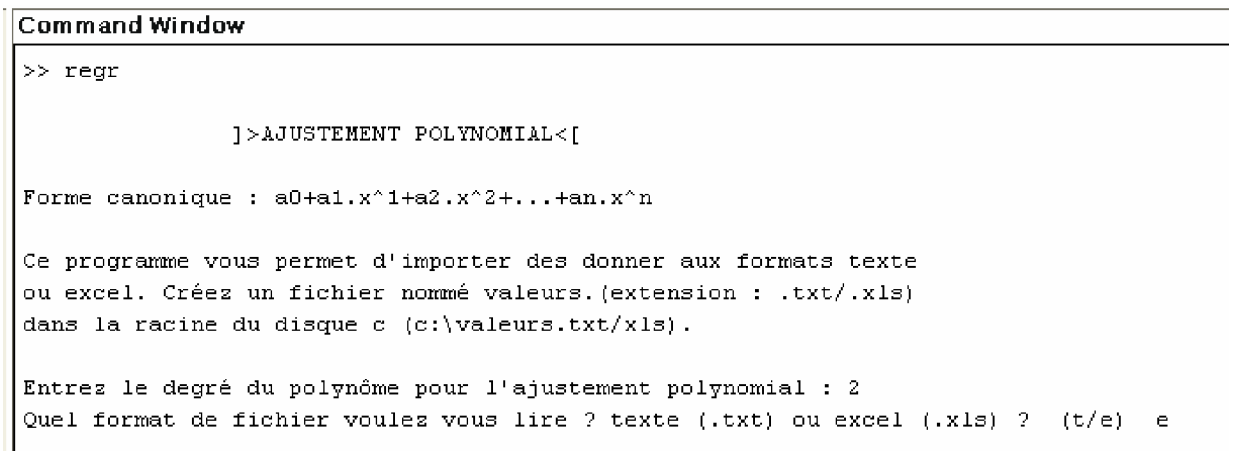
```
    S(8)=2.306;                 S(18)=2.101;    S(29)=2.045;
    S(9)=2.262;                 S(19)=2.093;    S(30)=2.042;
    S(10)=2.228;                S(20)=2.086;    S(31)=1.960; % nb de
                                S(21)=2.080;    ddl=infini

% Si N-r-1 est supérieur à 30, on prendra
% l'indice 1.960 (inf).
if (N-r-1)<31
    t=S(N-r-2);
else t=S(31);

end
```

r. matricev.m

```
function M=matricev(x,N,r)

% Cette fonction procède comme la fonction matrice, mais elle fournira
une
% matrice carrée, sans la dernière colonne faisant intervenir les
valeurs
% y.
% Elle est nécessaire au calcul de la matrice variance-covaraiance.

M(1,1)=N; % Initialisation de la première valeur de la matrice
for i=2:r+1
    M(i,1)=somme1(i-1,N,x); % Création de la première ligne
end
max=r; % Repérage de la dernière puissance de la colonne
for j=2:r+1
    for i=1:r
        M(i,j)=M(i+1,j-1); % Copie de la première ligne en respectant la
    end                    % symétrie de la matrice par rapport à la
diagonale
    max=max+1; % Repérage de la dernière puissance de la colonne
    M(i+1,j)=somme1(max,N,x); % Valeur du dernier élément de la colonne
traitée
end

end
```

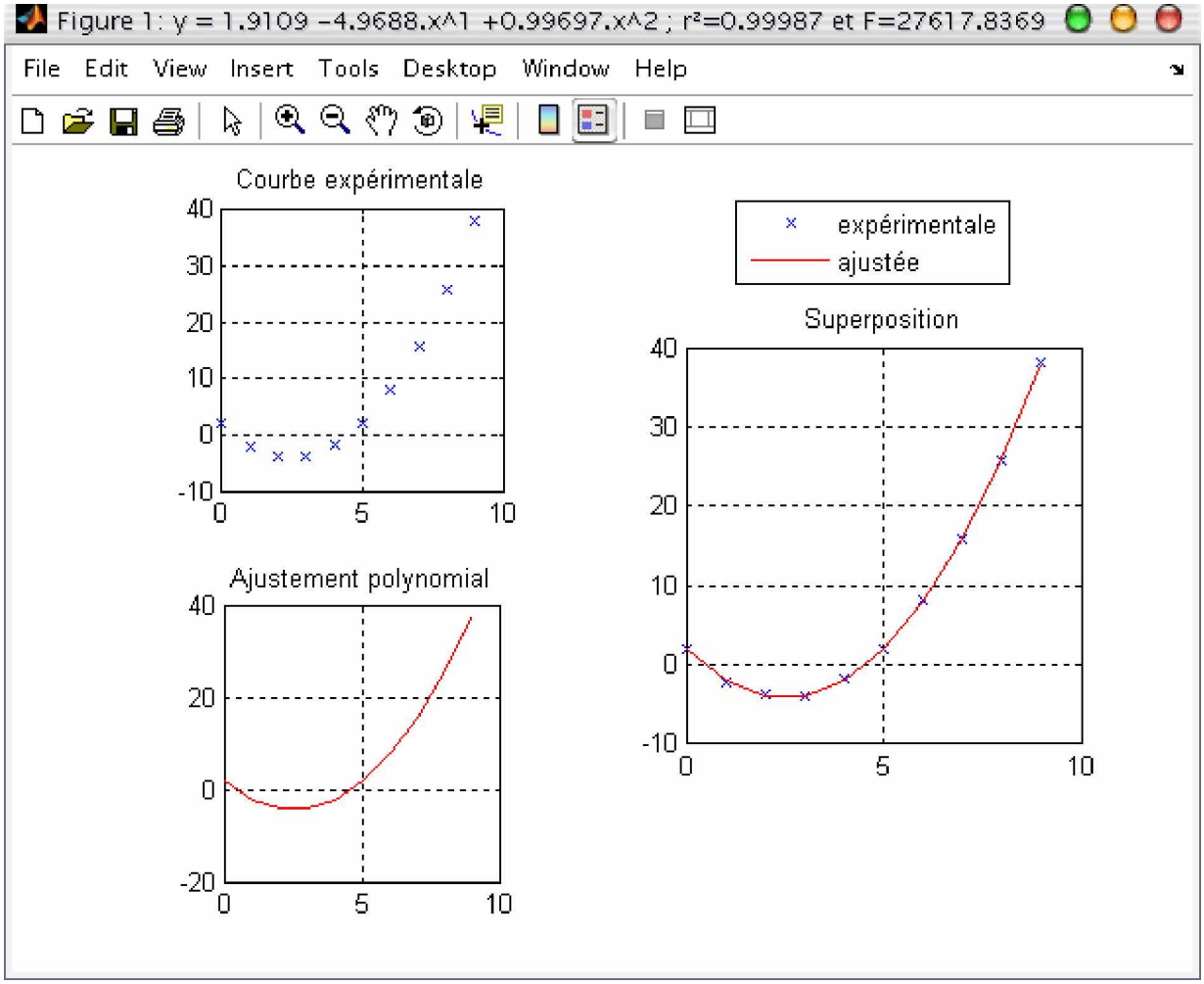
9. Captures d'écran

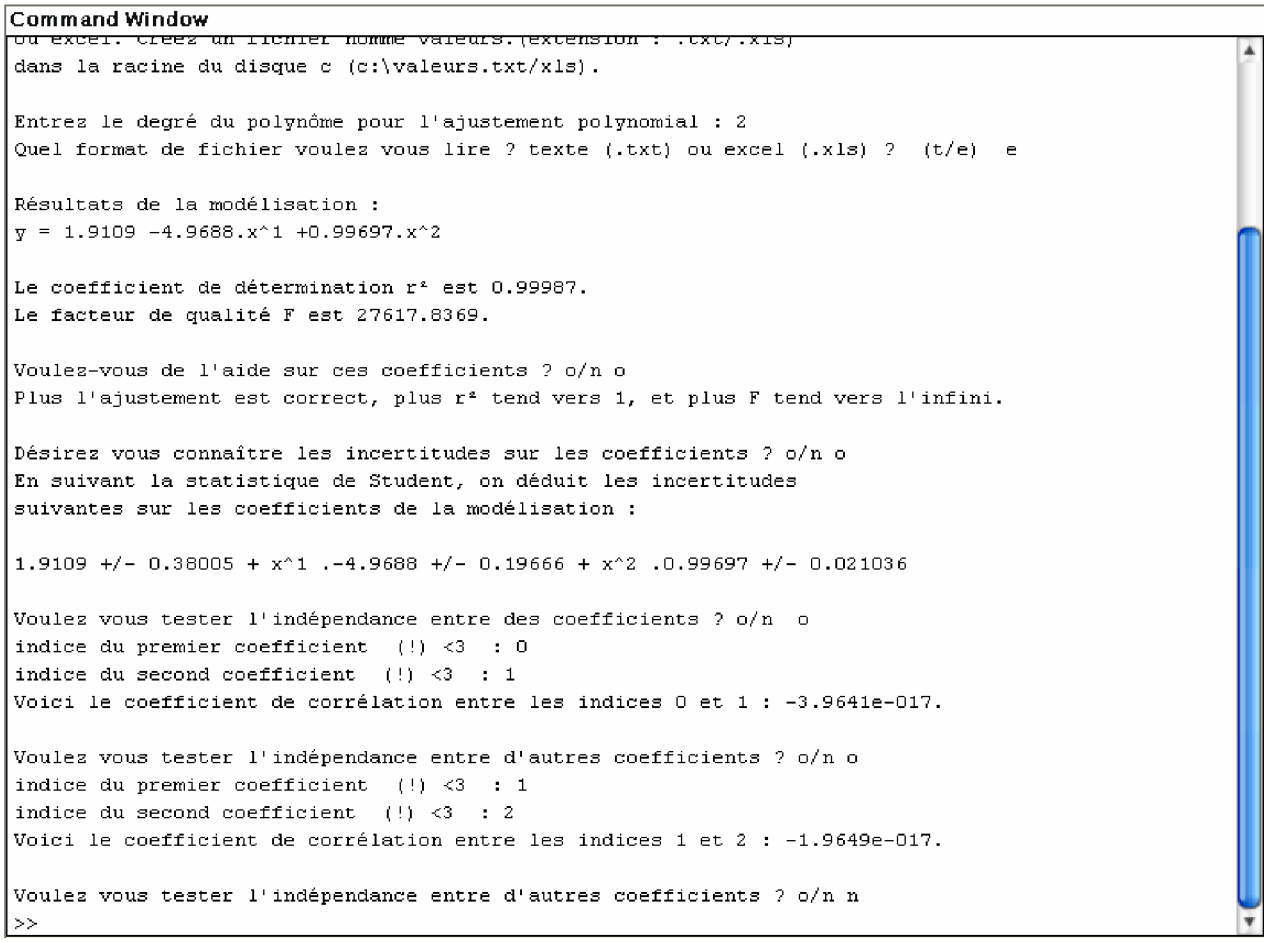
Voici quelques captures d'écran du programme en marche :



Command Window

```
>> regr

                ]>AJUSTEMENT POLYNOMIAL<[

Forme canonique : a0+a1.x^1+a2.x^2+...+an.x^n

Ce programme vous permet d'importer des donner aux formats texte
ou excel. Créez un fichier nommé valeurs.(extension : .txt/.xls)
dans la racine du disque c (c:\valeurs.txt/xls).

Entrez le degré du polynôme pour l'ajustement polynomial : 2
Quel format de fichier voulez vous lire ? texte (.txt) ou excel (.xls) ?  (t/e)  e

Résultats de la modélisation :
y = 1.9109 -4.9688.x^1 +0.99697.x^2

Le coefficient de détermination r² est 0.99987.
Le facteur de qualité F est 27617.8369.

Voulez-vous de l'aide sur ces coefficients ? o/n o
Plus l'ajustement est correct, plus r² tend vers 1, et plus F tend vers l'infini.

Désirez vous connaître les incertitudes sur les coefficients ? o/n
```




Command Window
ou excel. Créez un fichier nomme valeurs.(extension : .txt/.xls)
dans la racine du disque c (c:\valeurs.txt/xls).
Entrez le degré du polynôme pour l'ajustement polynomial : 2
Quel format de fichier voulez vous lire ? texte (.txt) ou excel (.xls) ? (t/e) e
Résultats de la modélisation :
y = 1.9109 -4.9688.x^1 +0.99697.x^2
Le coefficient de détermination r² est 0.99987.
Le facteur de qualité F est 27617.8369.
Voulez-vous de l'aide sur ces coefficients ? o/n o
Plus l'ajustement est correct, plus r² tend vers 1, et plus F tend vers l'infini.
Désirez vous connaître les incertitudes sur les coefficients ? o/n o
En suivant la statistique de Student, on déduit les incertitudes
suivantes sur les coefficients de la modélisation :
1.9109 +/- 0.38005 + x^1 .-4.9688 +/- 0.19666 + x^2 .0.99697 +/- 0.021036
Voulez vous tester l'indépendance entre des coefficients ? o/n o
indice du premier coefficient (!) <3 : 0
indice du second coefficient (!) <3 : 1
Voici le coefficient de corrélation entre les indices 0 et 1 : -3.9641e-017.
Voulez vous tester l'indépendance entre d'autres coefficients ? o/n o
indice du premier coefficient (!) <3 : 1
indice du second coefficient (!) <3 : 2
Voici le coefficient de corrélation entre les indices 1 et 2 : -1.9649e-017.
Voulez vous tester l'indépendance entre d'autres coefficients ? o/n n
>>