Le programme :

Nous deux séances de travaux pratiques sont consacrées à l'étude des :

- ***Problèmes de précision dans le calcul numérique.***
- ***Méthodes de résolution des systèmes linéaires.***
- ***Méthodes d'interpolation et d'approximation polynômiales.***

Les modalités :

Les élèves pourront préparer les séances de TP en réfléchissant sur les fragments de programmes proposés (disponibles en ligne) et en répondant aux questions théoriques.
Les questions Q1 à Q10 font l'objet de l'étude de la première séance et les questions Q11 à Q20, l'objet de la dernière séance.

Les étapes soumises à la validation et marquées par un V doivent être commentées devant un enseignant. Seules les réponses aux questions marquées de ✍ sont à rédiger et à remettre à la fin de la séance.

Objectifs :

1. ***Limites de la représentation des réels simple et double précision.***
2. ***Propagation des erreurs à travers un algorithme itératif.***

Les limites de la représentation des nombres engendrent des problèmes de précision. Plus exactement, nous avons vu en cours que la précision de l'opération d'addition pouvait être altérée lorsque les termes d'une somme n'étaient pas du même ordre de grandeur. Nous mettons en évidence dans cette première partie, les conséquences de la propagation de ce type d'erreur à l'issu d'un calcul itératif.

CAS D'UNE SIMPLE ADDITION

Q1. Le programme ci-contre réalise la somme de deux flottants simples u et v tels que :

u=1 et v=2^{-p}.

- **Rappeler** la représentation des réels selon la norme IEEE754 en simple précision.
- Examiner l'influence de p sur la précision de la somme en exécutant ce programme.
- Pour quelle valeur de p le calcul de la somme devient faux ? Justifier cette valeur.
- *Noter le rapport v/u correspondant ?*

Nom1 | Précision opération.cpp

```
#include <stdio.h>
#include <math.h>
float u, v, s; short p;

main()
{
u=1; printf("\t u=%1.10e\n",u);
for (p=0; p<=30; p=p+1)
{
v=pow(2,-p); s=v+u;
printf("p=%d\t v=%1.10e\t v+u=%1.20e\n",p,v,s);
}
getchar();
}
```

PROPAGATION DES ERREURS

Nous examinons ici la propagation de ce type d'erreur à travers le développement en série de la fonction e^x pour x<<0. Les algorithmes de calcul étant étudiés en cours, les élèves sont conduits vers une analyse de leurs performances.

Q2. Nous développons l'algorithme proposé en cours dans le programme ci-contre.

- A quoi correspondent les variables Tmin et Tmax ? Et le fichier termg1.txt ?
- Quelle est le résultat de l'exécution de la ligne : **fprintf(fp,"%e\n", term);** ?
- Exécuter ce programme pour plusieurs valeurs de x et comparer le résultat du calcul avec la valeur théorique de e^x.
- Consigner les résultats dans un tableau.

x	e^x(cal)	e^x(théo)	k_{MAX}	erreur
-10				
-15				
-20				
-25				
-30				

- Quelles sont les valeurs de x qui révèlent les limites de notre premier algorithme ?

gOptimisé.cpp | propagErr.cpp

```
#include <stdio.h>
#include <math.h>
double x,y,R; int n,Q;

double expo(double x, double eps)
{       double sum=1, term=1; long k=1;
        double Tmin=term, Tmax=0;
        FILE *fp; fp=fopen("termg1.txt", "w");
        do  {term=term*x/k; sum=sum+term; k=k+1;
            if (Tmin>fabs(term)) {Tmin=fabs(term);}
            if (Tmax<fabs(term)) {Tmax=fabs(term);}
            fprintf(fp,"%e\n", term);
            printf("k=%d \t term=%e\n",k, term);}
        while (fabs(term)>eps);
        fclose(fp);
        printf("Tmin=%e\t Tmax=%e\n\n",Tmin,Tmax);
        return sum;             }

main()
{
printf("-------Algorithme initial---------\n\n");
x=-30; y=expo(x,1e-20);
printf("expo(%2.0f",x); printf(")=%e\n\n",y);
getchar();
}
```

OPTIMISATION D'UN ALGORITHME

Q3. La trame d'une amélioration possible de l'algorithme de calcul est proposée dans la fenêtre ci-contre.

- **Compléter ce code** et enregistrer les résultats numériques dans un fichier que nous appellerons termg2.txt.
- Exécuter le programme en notant les nouvelles performances de calcul.

```
main()
{
printf("--------Algorithme optimise--------\n\n");
x=-30;
Q=.......; R=.........;
printf("Q=%d\t R=%e\n\n",Q,R);
y=.......*expo(.,1e-20);
printf("expo(%2.0f",x); printf(")=%e\n\n",y);
printf("-------------------------------------\n");
getchar();
}
```

COMPARAISON DES ALGORITHMES

Q4. Le programme AN08_COMP.m permet de charger les fichiers termg1.txt et termg2.txt dans les matrices val1 et val2 afin de les analyser plus confortablement au moyen des outils graphique de MATLAB.

- Exécuter ce programme dans MATLAB et optimiser les repères en mode log-log.
- Commentaires et comparaison des deux algorithmes.

```
%---Lecture des données du 1er calcul
%-------------------------------------
fid1=fopen('termgn1.txt', 'r');
val1=fscanf(fid1, '%f'); fclose(fid1);

%---Lecture des données du 2nd calcul
%-------------------------------------
fid2=fopen('termgn2.txt', 'r');
val2=fscanf(fid2, '%f'); fclose(fid2);
```

Objectifs :

1. *Conditionnement d'un système linéaire et stabilité.*
2. *Résolution par la méthode du pivot de Gauss.*
3. *Résolution par les méthodes itératives.*

I. CONDITIONNEMENT ET STABILITE D'UN SYSTEME LINEAIRE

Nous exploitons ici quelques outils programmés de MATLAB dans le but de mettre en évidence l'importance du conditionnement d'un problème. L'équation Ax=b sera donc tout simplement résolue par inversion de la matrice A dans MATLAB puisque le vecteur inconnu s'écrit : $x=A^{-1}.b$.

INVERSION D'UNE MATRICE

Q5. Le petit programme ci-contre simule et résout le système linéaire **A.x=b** à 5 inconnues avec :

$$\mathbf{A}=\begin{bmatrix} 1 & a_1 & a_2 & a_3 & a_4 \\ -a_4 & -1 & -a_1 & -a_2 & -a_3 \\ a_3 & a_4 & 1 & a_1 & a_2 \\ -a_2 & -a_3 & -a_4 & -1 & -a_1 \\ a_1 & a_2 & a_3 & a_4 & 1 \end{bmatrix} \quad \mathbf{x}=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \quad \mathbf{b}=\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

où : $a_1=\frac{1}{2}$ $a_2=\frac{1}{4}$ $a_3=\frac{1}{8}$ $a_4=\frac{1}{16}$

$b_1=b_5=1$ $b_2=b_4=2$ $b_3=3$

```
1 -  clc; clf;
2 -  n=5; x=zeros(n,1);
3
4 -  a1=1/2; a2=1/4;
5 -  a3=1/8; a4=1/16;
6 -  a1=a1*1.0;
7
8 -  A=[1   a1   a2   a3   a4;
9 -    -a4 -1 -a1 -a2 -a3;
10 -   a3   a4   1    a1   a2;
11 -   -a2 -a3 -a4 -1   a1;
12 -   a1   a2   a3   a4   1];
13
14 - b=[1; 2; 3; 2; 1];
15
16 - N=norm(A); M=norm(inv(A));
17 - C=cond(A); D=det(A);
18
19 - %--Résolution par inversion de A
20 - x=inv(A)*b
```

- Exécuter ce programme et relever sa sortie x.
- Relever la valeur des variables N, M, C et D.
- A quoi correspondent ces grandeurs ? Commenter qualitativement ces valeurs numériques.
- Examiner et mesurer l'influence d'une variation de 1% du coefficient a_1 ($a_1=a_1*1.01$) sur le calcul de x.

STABILITE ET PRECISION

Q6. On considère maintenant le système **A'.x=b**. La matrice **A'** est décrite dans l'encadré ci-contre.

$$\mathbf{A'}=\begin{bmatrix} 1 & a_1 & a_2 & a_3 & a_4 \\ -1 & -a_3 & -a_1 & -a_2 & -a_4 \\ 1 & a_3 & a_2 & a_1 & a_4 \\ -1 & -a_2 & -a_1 & -a_3 & -a_4 \\ 1 & a_4 & a_3 & a_2 & a_1 \end{bmatrix}$$

```
Command Window
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 3.372829e-018.
```

- Reprendre l'étude précédente avec la matrice **A'**.
- Relever les nouvelles valeurs de C et de D.
- Examiner et mesurer une nouvelle fois l'influence d'une variation de 1% du coefficient a_1 sur le calcul de x.
- Justifier qualitativement les avertissements (*Warning*) générés par la fenêtre de commande de MATLAB :
- Comparaison avec les résultats de Q5.

III. RESOLUTION D'UN SYSTEME PAR LA METHODE DU PIVOT DE GAUSS

PHASE DE TRIANGULATION

Q7. Nous rappelons avec l'encadré ci-contre l'algorithme de triangulation d'un système de dimension N.

```
% ---------- triangulation ----------
n=5; n1=0;n2=0;n3=0;
for k=1:n-1
    n1=n1+1;
    for i=k+1:n
        Lambda=A(i,k)/A(k,k); n2=n2+1; i
        for j=k:n
            A(i,j)=A(i,j)-Lambda*A(k,j);
            n3=n3+1; j
            At=A
        end
        b(i)=b(i)-Lambda*b(k); bt=b
    end
end
% ------------------------------------
```

- Exécuter ce programme et relever le nouveau système $\mathbf{A_t.x=b_t}$ à la fin de la transformation de triangulation.
- Résoudre en premier lieu le système par simple inversion de $\mathbf{A_t}$ ($x=A_t^{-1}.b_t$). Comparer le résultat à celui obtenu en Q_1.

Q8. Nous avons établit en cours la complexité théorique de l'algorithme de triangulation.

- Relever la valeur finale des variables n1, n2 et n3.
- Combien d'itérations ont été nécessaires pour évaluer la matrice $\mathbf{A_t}$ et le vecteur colonne $\mathbf{b_t}$?
- Comparer ces résultats à l'ordre de complexité de l'algorithme de triangulation (en $n^3/3$; cf. cours).

PHASE DE REMONTEE

Q9. La résolution de la dernière ligne du système que nous venons de modéliser par l'équation $\mathbf{A_t.x=b_t}$ est triviale. Il s'agit maintenant de résoudre le système complet en le remontant ligne par ligne au moyen de l'algorithme ci-contre.

```
% ------------ Résolution------
x=b; n4=0;
for k=n:-1:1
    for j=k+1:n
        x(k)=x(k)-A(k,j)*x(j)
        n4=n4+1
    end
    x(k)=x(k)/A(k,k);
end
```

- Exécuter cet algorithme et relever sa sortie dans le vecteur x.
- Relever et commenter la valeur finale de n4.
- Combien d'itérations ont été finalement nécessaires à la résolution du système par l'algorithme du pivot?

PROBEMES ET SOLUTIONS

Q10. On considère maintenant le système **A''.x=b''** dans lequel **A''** et **b''** sont des matrices déduites de **A** et de **b** par simple permutation des lignes.

$$\mathbf{A''}=\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & 1 \\ -a_2 & -a_3 & -a_4 & -1 & -a_1 \\ a_3 & a_4 & 1 & a_1 & a_2 \\ -a_4 & -1 & -a_1 & -a_2 & -a_3 \\ 1 & a_1 & a_2 & a_3 & a_4 \end{bmatrix} \quad \mathbf{b''}=\begin{bmatrix} b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \end{bmatrix}$$

- Résoudre le système par inversion de la matrice **A''**.
- Examiner la stabilité du système en notant la valeur de det(**A''**) et de cond(**A''**).
- Résoudre le système en employant l'algorithme du pivot. Quelle est l'origine des erreurs générées ici ?
- Corriger l'algorithme du pivot de Gauss.

Fin du travail de la première séance ...

I. RESOLUTION DE SYSTEMES PAR LES METHODES ITERATIVES

DECOMPOSITION DE LA MATRICE A

Q11. Les algorithmes étudiés sont basés sur une répartition particulière de la matrice **A** de sorte que : **A= D – (U+L)**.

- Vérifier cette décomposition de **A** en relevant les matrices générées par le petit programme ci-contre.

```
% ------ Décomposition de A -------
n=5; U=zeros(n,n); L=zeros(n,n);
     D=zeros(n,n); UL=zeros(n,n);

for i=1:n
    D(i,i)=A(i,i); iD(i,i)=1/D(i,i);
    for j=1:n
    if (j>i) U(i,j)=-A(i,j); end
    end
end
L=D-U-A; UL=U+L;    %----- UL=D-A;
```

La résolution du système sera alors assurée par la convergence d'une suite de vecteurs x_k vers une limite définie par une récurrence de la forme :

$$x_k=M^{-1}.(N.x_{k-1}+b)$$

M et N étant deux matrices fonction de A, de L et de U.

ALGORITHME DE JACOBI

Q12. L'algorithme de résolution du système **A.x=b** par la méthode de Jacobi est illustrée ci-contre.

```
% ------------ Jacobi -------------
n=5;
e=zeros(n,1); x0=zeros(n,1);
erreur=zeros(1,45); nj=0; eps=1;
while (eps>0.01)
    nj=nj+1; x0=x; x=iD*(UL*x+b);
    eps=((norm(x-x0))/norm(x));
    erreur(nj)=eps;
end
% ---- Représentions graphiques ----
figure(1);subplot(211); plot((erreur));
xlabel('itérations'),ylabel('erreur');
grid; title('Algo de Jacobi: Convergence');
```

- Rappeler (ou déduire du programme) l'expression des matrices **M** et **N**.
- Exécuter ce programme et relever sa solution.
- Combien d'itérations ont été nécessaires à cet évaluation ?
- Quelle est la précision atteinte ? Commenter la figure 1.
- Comparaison avec la méthode du pivot de Gauss.

ALGORITHME DE GAUSS-SEIDEL

Q13. Nous présentons au moyen de l'encadré ci-contre un premier approche de la méthode de Gauss-Seidel dans le but de marquer sa différence avec celle de Jacobi.

```
% ------- Gauss Seidel ppe théorique -------
n=5; x=zeros(n,1);
e=zeros(n,1); x0=zeros(n,1);
erreur=zeros(1,45); ngs=0; eps=1;
while (eps>0.01)
    ngs=ngs+1;
    x0=x; x=inv(D-L)*(U*x+b);
    eps=((norm(x-x0))/norm(x));
    erreur(ngs)=eps;
end
% ------ Représentations graphiques ------
subplot(212); plot(erreur);grid;
xlabel('itérations'),ylabel('erreur');
title('Gauss-Seidel: Convergence théorique');
```

- Quelle est la nouvelle expression des matrices **M** et **N** ?
- Exécuter ce programme et relever sa solution.
- Combien d'itérations ont été nécessaires à cet évaluation ?
- Quelle est la précision atteinte ?
- Comparaison avec la méthode de Jacobi.

Q14. Montrer que nous pouvons éviter l'inversion de la matrice (D-L) en remplaçant la boucle *while* du programme précédant par celle qui est représentée ci contre.

```
while (eps>0.01)
    ngs=ngs+1; x0=x;
    for i=1:n
        somme=0;
        for j=1:n
            somme=somme-UL(i,j)*x(j);
        end
        x(i)=(b(i)-somme)/A(i,i);
    end
    eps=((norm(x-x0))/norm(x));
    erreur(ngs)=eps;
end
```

- Exécuter de programme et comparer son résultat à celui obtenu en Q_9.

Les élèves sont conduits à expérimenter les méthodes d'interpolation et de régression pour modéliser l'évolution d'une valeur boursière de 1996 à nos jours.

Les points de mesures qui serviront à cette étude correspondent au cours affiché au 1er janvier de chaque année. Ces valeurs sont consignées dans les tableaux [année] et [valeur] du fichier ANTP3.m.

Extrait de http://bourse.latribune.fr/: Evolution du CAC40 depuis 1996.

II. INTERPOLATION POLYNOMIALE

SPLINES CUBIQUES DE MATLAB

Q15. Le programme ANTP3.m ci-contre permet d'interpoler de nos observations par la méthode des splines cubiques.

- Rappeler le principe de cette méthode.
- Identifier les variables utilisées dans le programme.
- Exécuter ce programme et commenter l'allure de la courbe obtenue.

```
% ----------------- SPLINES DE MATLAB -
for x=1:(12*(N-1))
Ps(x) = interp1(mois,valeur,x,'spline');
temp(x)=x-1;
end;
figure(1);
subplot(221); %axis([0 12*13 0 8]);
plot(temp,Ps,'r',mois,valeur,'*'); grid;
title('Splines cubiques de Matlab');
```

METHODE DE LAGRANGE

Q16. La trame de l'algorithme de l'interpolation de Lagrange est présentée ci-dessous.

- Rappeler la méthode de Lagrange et compléter ce programme.
- Comparer le résultat de cette interpolation avec celle des splines cubiques de MATLAB.
- Estimer sans faire de calcul, un encadrement de l'ordre du polynôme de Lagrange.
- Justifier les phénomènes de « bords » sur la représentation graphique de $P_L(x)$.

METHODE DE NEWTON

Q17. Réaliser le même travail avec la méthode de Newton. Quelle est la fonction des deux grandes boucles for ?

```
% ---------------- LAGRANGE ----------------------------
l=zeros(1,N);
for x=1:(12*(N-1))
    somme=0;
    for i=1:N
        produit=1;
        for j=1:N
            if (j~=i)        % Test i<>j
            produit=produit*(x-....(j))/(....(i)-mois(
            end;
            l(i)=produit;
        end;
        somme=somme+valeur(i)*l(i);
    end;
    Pl(x)=somme;  temp(x)=x;
end;

subplot(222);
plot(temp,Ps,'r',temp,Pl,'g',mois,valeur,'*');
title('Splines cubiques / Lagrange');
grid;
```

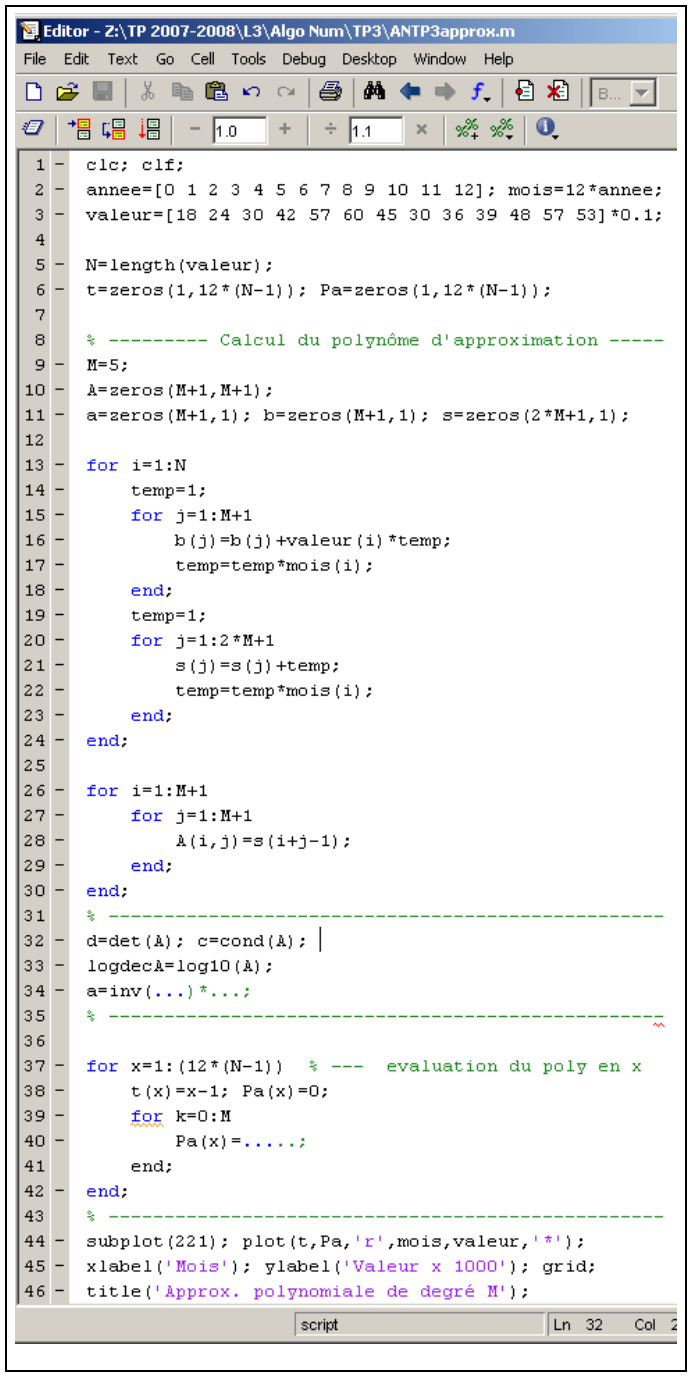
```
% ---------------- NEWTON ----------------------

a=......;    %-- Calcul des coeff par diff. div.
for k=1:(N-1)
    for i=k+1:N
        a(i)=(a(i)-a(k))/(mois(i)-....(k));
    end;
end;
a

for x=1:(12*(N-1))  % ---  evaluation pour x
    Pn(x)=a(N);
    for i=1:(N-1)
        Pn(x)=a(N-i)+(x-mois(N-i))*Pn(x);
    end;
end;

subplot(223);
plot(temp,Ps,'r',temp,Pn,'b',mois,valeur,'*');
title('Splines cubiques / Newton');
grid;
```

III. APPROXIMATION POLYNOMIALE : REGRESSION

DEGRE D'APPROXIMATION

Q8. Nous esquissons dans la fenêtre ci-contre la modélisation de nos mesures au moyen d'une approximation polynomiale.

- **Compléter** de programme (lignes 34 et 40).
- Etablir (ou rappeler) l'expression théorique de la **matrice A** et du vecteur b.
- Quelle est la variable qui représente le degré du polynôme d'approximation ?
- Quel est le tableau qui consigne les coefficients de ce polynôme ?



STABILITE ET PRECISION

Q9. Exécuter le programme pour plusieurs valeur du degré de ce polynôme (1<degré<12).

- Quelle est le polynôme qui offre la meilleure approximation?
- Relever la représentation graphique du polynôme d'approximation pour un degré = 5.
- Relever également les valeur s de d, de c et de logdec(A) à partir de la fenêtre de commande.
- Calculer le rapport A_{66}/A_{11} et commenter les éventuels Warning générés dans la fenêtrer de commande de MATLAB.

Q10. Nous examinons les conséquences de l'instabilité de nos calculs en relevant l'allure du même polynôme (ordre 5) après une variation de 0.001% sur A_{55} :

```
A(5,5)=A(5,5)*1.00001;
```

- Commenter l'allure de la nouvelle représentation graphique du polynôme d'approximation.