"Analyse Numérique et programmation"

11/02/2019

programme

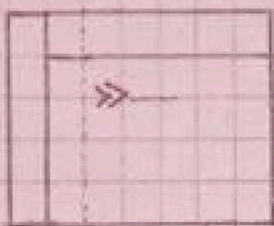
Entré → • les données

• Résolution de problème

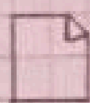
• les résultats → sorté

On utisent (MaTLab)

Sargent chef.



Editeur ✓



• après ecriver le programme et fait Regestrée par exemple (Mebarki Zineb)

→ ne utilise pas les signe mathematique $\frac{I}{x}$ ...

→ ما نديروش فراغ بين الحروف

1

→ ما نبداوش برقم

Exp: • ISS + M ×
• ISS فراغ M ×
• 7ISS ×

• la partie 01 (les Données)

lasque je declare une variable en majuscale M = 2

نبقى في كامل البروقرام نكتب M كبيرة ✓

• في Sargent chef نرجع بالاسم الي خبيته لي قبل

• x=7 ;   y=3 ;

• Z = x+y

• Z=10

→ Ecrire le vecteur suivant

V=(1,2,3) ⇒ [1 2 3] (vide ou virgule)

$S=\begin{pmatrix}3\\4\\5\end{pmatrix}$ ⇒ [3;4;5]

→ Matrice : $\begin{pmatrix}1 & 5 & 2\\ -3 & 2 & 1\\ 0 & 5 & 6\end{pmatrix}$

• M=[1 5 2 ;-3 2 1; 0 5 6]
vide

: ⇒ permet la discrétisation d'intervalle.

Exemple:

• V=1:5

V = 1 2 3 4 5

2

V=1 3 5 7 → V=1:2:7

V=3,5,7 → V=3:2:7

$M=\begin{pmatrix}a & b & c\\ d & E & \boxed{F}\\ g & h & I\end{pmatrix}$ ⇒ =M(2,3)   S=F

$M=\begin{pmatrix}1 & 2 & 3 & 4 & 5\\ 6 & 7 & 8 & 9 & 10\\ 11 & 12 & 13 & 14 & 15\end{pmatrix}$ ⇒ S=M(2:3, 1:4)
3 – 2   4 – 1

• les format possible sur Matlab:

• format short:

>> $\frac{5}{6}$ = 0,8333   أرقام بعد الفاصلة 4

• format short e :

$\gg \frac{5}{6} = 8,3333e-0001$

• format blank :

$\gg \frac{5}{6} = 0,83$ رقمين بعد الفاصلة

• format Rat :

$\gg \frac{12}{9} = \frac{4}{3}$ الاختزال

- كيف نكتب شرطي "MATLAB" :

condition :

syntaxe : if (condition)
exécution (وش حاب)
f(x) (نكتب دالة)

Exp : <u>EXO</u> : écrire dans MATLAB :

③

$$f(x) \begin{cases} x+1 & \text{si } x \geq 1 \\ x-1 & \text{si } x < 1 \end{cases}$$

$\underset{y}{\underline{f(x)}} = \{$

✓.. 1ère Méthode ..✓

<u>Programme</u> :

ماتكتبهاش في الطلب الا يتوا أهلها

```
• clc
• clear all
• x = input ('x = ');
• if x >= 1
      y = x + 1;
• end;
• if x < 1
      y = x - 1;
• end;
```

F5 + pour Enregistrer

$$f(x) = \begin{cases} x+1 & \text{si } x>1 \\ 2 & \phantom{\text{si }} x=2 \\ x-1 & \phantom{\text{si }} x<1 \end{cases}$$

- clc
- clear all
-

la 2ème méthode :

```
• clc
• clear all
• x=input ('x=')
• if  x>1
      y=x+1;
• if  x<1
      y=x-1;
• else  y=2
end;
```

Exp : Ecrire un programme qui Resoudre l'équation suivante :

$$ax + b = 0$$

b و a مجاهيل 2

[illegible]

- les données a . b ←

نديرو شرط لـ a,b

Si $\begin{cases} a=0; \\ b=0; \end{cases}$ → pas de sens.

Si $\begin{cases} a=0 \\ b\neq 0 \end{cases}$ → impossible de resoudre l'équation.

Si $\begin{cases} a\neq 0 \\ b\neq 0 \end{cases}$ → $x=-\frac{b}{a}$

- clc
- clear all
-
-
-
-
-
-

5

Programme :

```
• clc
• clear all
• input ('a=')
• input ('b=')
• if a==0
• if b==0
•   x=input ('pas de sens l'équation')
• else
•   x=input ('impossible de Resoudre l'équation')
• end;
• else
    x=-b/a
• end;
```

F5 → pour Enregistrer

a=0 b=0
('pas de sens de l'equation')

a=0 b≠0
('impossible de Resoudre l'équation')

a≠0 b≠0
(a=1 b=3)
$(-\frac{3}{1}=-3)$

EXO : Ecrire le programme qui résoudre l'équation suivante :

$\rightarrow ax^2 + bx + c = 0$

18/02/2009

a, b, c

si a==0
si b==0
si c==0
$\Rightarrow$ l'equation aucun sens
sinon impossible
exo la semaine passé
sinon
deltat = b*b + 4*a*c
if deltat > 0
2 sol $x_1 = \frac{-b+\sqrt{\Delta}}{2a}$

$x_2 = \frac{-b-\sqrt{\Delta}}{2a}$

else if deltat = 0
$x = -\frac{b}{2a}$

else pas sol dans $\mathbb{R}$
end;

6

Programme :

```
clc
clear all
a=input('a=')    b=input('b=')    c=input('c=')

if a==0
  if b==0
    if c==0
      x=input('pas de sens à l'eqt')
    else
      x=input('impossible')
    end
  else
    x=inputs('sol unique')
    x = -c/b
end
```

```
else
    deltat = b*b-4*a*c

if deltat >0
        x12 = input('2 sol possibles');
        x1 = (-b - sqrt(deltat))
                   (2*a)
        x2 = " + "
                 "

else if deltat <0
        x=input('pas sol dans R')
  else
        x12 =input('sol double')
        x12=-b/(2*a)
  end
end
```

7

• **Les boucles algorithmiques:**

for ≡ pour ; while ≡ tant que ; repeat ≡ répète

• **La boucle for (pour):**

Syntaxe :

```
for  cdt        while  cdt
                          |
   -act 1                 |
   -act 2                 |
   -act 3                 |
     |                    |
     |                    |
   end                  end
```

• Repeat (répète)

```
repeat
    act 1
    act 2
      :
  until  cdt
end
```

exemple : écrire un programme qui calcule la somme suivante

$$S = 1 + 2 + 3 + \cdots + 100 = \sum_{i=1}^{100} i$$

l'act pour la somme

som = som + act

initiation

programme :

```
clc
clear all
s=0;
 for i=1:100
     s=s+i;
end
s
```

$S = 1 + 3 + 5 + 7 \cdots$

كي نحسبو الحدود حتى لمية حد

programme :

```
clc
clear all
s=0;
 for i=1:2:100
     s=s+i;
end
s
```

كي يقولنا احسبو المئة حد الأولى (100)

programme :

```
clc
clear
s=1;
for i=1:100
    s= 2*i+1
end
s
```

$$S = 1 - 2 + 3 - 4 + 5 - 6 \cdots = \sum_{i=1}^{n} (-1)^{i} i$$

$= 1 + (-1)2 + 3 + (-1)4 + \cdots$

s = s + act

solution :

programme :

```
clc
clear all
n=input('n=');
s=0;
al=1;
for i=1:n
 s=s+al*i;
 al=-al;
end
s
```

المحاكاة:

$i=1 \rightarrow s=0+1*1=1$

$\quad \hookrightarrow al=-1$

$i=2 \rightarrow s=1+(-1)*2=-1$

$\quad \hookrightarrow al=-(-1)=1$

$i=3 \rightarrow S=\underbrace{1-2}+1.3$

+ produit : $\boxed{prod * act}$

prod → valeur initial de produit

exemple 1 :

$$p=1\times2\times3\times4\times5\times \cdots = \prod_{i=1}^{n=5} i$$

programme :

```
clc
clear all
n=input('n=');
p=1;
for i=1:n
 p=p*i
end
p
```

9

exemple 2 :

ecrire un programme qui calcule la somme suivante :

$$S=1+x+x^2+x^3+x^5+\cdots$$

$$=1+x+x*x+x*x*x+\cdots$$

Solution :

25/02/2019

Programme

```
clc
clear all
s=1
p=1
% x=sym('x');
x=input('x=');
n=input('n=');
for i=1:n
p=p*x;
s=s+p;
end;
s
```

Programme

```
clc
clear all
s=1
p=1
x=sym('x');
%x=input('x=');
n=input('n=');
for i=1:n
p=p*x;
s=s+p;
end;
s
```

Exo:

Ecrire un programme qui calcul la somme suivante:

$$s = 1 + x^2 + x^4 + x^6 + x^8 \ldots$$

10

Solution :

Programme.

```
clc
clear all
s=1
p=1
%x=sym('x')
x=input('x=').
n=input('n=');
for i=1:n
    p=p*x*x;
    s=s+p;
end
s
```

Programme

```
clc
clear all
s=1
p=1
x=sym('x')
% x=input('x=');
n=input('n=');
for i=1:n
    p=p*x*x;
    s=s+p;
end
s
```

Exo: Ecrire un programme qui calcule la somme suivante:

$S = 1 + 2! + 3! + 4! + 5! + \ldots$

programme:

```
clc
clear all
n=input('n=')
S=1;
fact=1
for i=2:n
 fact=fact*i;
 S=S+fact;
end
S
```

$n=4$, $S=1$, $fact=1$

$i=2 \rightarrow fact = 1\times 2 = 2$
$\quad \rightarrow S = 1+2$

$i=3 \rightarrow fact = 2\times 3$
$\quad \rightarrow S = 1+2+6$

$i=4 \rightarrow fact = 2\times 3\times 4$
$\quad \rightarrow S = 1+2+6+24$

⋮

Exo:

$S = 1! + \underset{i=1}{3!} + \underset{i=2}{5!} + \underset{i=3}{7!}$

11

Programme:

```
clc
clear all
S=1;
for i=1:n
 fact=1;
 for j=1:2*i+1
  fact=fact*j
 end
 S=S+fact
end
S
```

Exo: ecrire un programme qui calcule le developement limité d'un "exp":

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} = \sum_{n=0}^{\infty} \frac{(x)^n}{n!}$$

programme:

```
clc
clear all
n=input('n=')
% x=sym('x')
x=input('x=')
s=1;
p=1;
fact=1;
for i=1:n
  p=p*x;
  fact=fact*i
s=s+p/fact
end
s
```

Exo:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!}$$

Exo:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!}$$

12

programme

```
clc
clear all
n=input('n=')
al=-1
s=1
p=1
x=sym('x')
% x=input('x=')
for i=1:n
p=p*x*x
 fact=1;
 for j=1:2*i
  fact=fact*j
 end
s=s+al*p/fact
 al=-al
end
s
```

Chapitre I : Système d'eq^t linéaire

Introduction :

Considérons un système d'eq^t linéaires (a_{ij}, b_j : données).

$$\begin{aligned} a_{11} x_1 + a_{12} x_2 - - - - - - + a_{1n} x_n &= b_1 \\ a_{12} x_1 + a_{22} x_2 - - - - - + a_{2n} x_n &= b_2 \quad \cdots (1) \\ \hline a_{n1} x_1 + a_{n2} x_2 - - - - - - + a_{nn} x_n &= b_n \end{aligned}$$

et cherchons sa solution $x_1, \ldots x_n$. Très souvent il est commode d'utiliser la notation matericielle $Ax = b$ --- (2).

l'eq^t (2) possède une solution unique si et seulement si ($\det A \neq 0$)

dans ce système, on a $A = |a_{ij}|$ connues, $b = b_i$ connues et $x = x_i$ inconnues

exemple : soit la matrice $A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & -1 & 0 \\ 5 & 4 & 3 \end{pmatrix}$. et $b = \begin{pmatrix} 6 \\ 0 \\ -1 \end{pmatrix}$

La solution numérique du système est $\det(A) \neq 0$.

$$Ax = b \Rightarrow x = \frac{b}{A} = A^{-1} b.$$

A^{-1} en matlab $\Rightarrow$ inverse de la matrice A.

```
>> inv(A) = -0,0682    0,2500    0,1136
            -0,1364   -0,1500    0,2273
             0,2955    0,2500   -0,1591
```

La solustion par matlab est :

```
>> x = A^(-1)*b

x = 0,5227
   -1,0455
    1,9318
```

Problèmes mal conditionnés :

il existe un certain nombre d'eq^t linéaires solvables, mais leurs solution sont incertaines a cause des erreurs d'arrondi.

lors des calculs. des pb de ce type sont appelés "problèmes mal conditionnées". l'arrondi durant des calculs ou le petits changement dans les coéfficients calculés peuvent induire des erreurs significative dans la résolution d'un problème

exemple : soient des système d'eq^t suivant :

$$\begin{cases} 0,12065\, x_1 + 0,98775\, x_2 = 2,01045 \\ 0,12032\, x_1 + 0,98755\, x_2 = 2,00555. \end{cases}$$

la solution de ce système est :

$$x_1 = 14,7403 \quad , \quad x_2 = 0,23942.$$

pour monter l'erreur introduite sur les solution en modifiant des coefficients (arrondi). On donne au coeff '2,01045' de la 1er eq^t une légère augmentation de '0,001'. l'eq^t ① devient : l'eq^t ② reste la même.

$$0.12065\, x_1 + 0,98775\, x_2 = 2,01145,$$

la solution est : $x_1 = 17,9756$, $x_2 = -0,15928$.

ceci met en evidence l'erreur engendrée par un arrondi.

3- Méthode directe (Méthode du Pivot) :

soit à resoudre le système suivant de 3 eq^t à 3 inconnues par la méthode du Pivot (dans matlab) :

$$\begin{pmatrix} -0,04 & 0,04 & 0,12 \\ 0,56 & -1,56 & 0,32 \\ -0,24 & 1,24 & -0,28 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 0 \end{pmatrix}$$

determiner x_1, x_2, x_3

solution :

1- On definit tout d'abord la matrice argument dans matlab par :

```
>> A = [-0,04 ␣ 0,04 ␣ 0,12 ␣ 3 ; 0,56 ␣ -1,56 ␣ 0,32 ␣ 1 ;
-0,24 ␣ 1,24 ␣ -0,28 ␣ 0 ] .
```

2- On choisit la ligne 1 comme pivot : a(1,1) = -0,104.

on divise la ligne 1 par : a(1,1) $\longrightarrow$ b(1,:) = a(1,:)/a(1,1).

$\rightarrow$ nouvelle ligne, on annule le 1er terme de la ~~2ème~~ et ligne et le 1er terme de la ligne 3 :

$\rightarrow b(2,:) = a(2,:) - b(1,:) * a(2,1)$.

$\rightarrow b(3,:) = a(3,:) - b(1,:) * a(3,1)$.

On obtient après calculs :

$$b = \begin{pmatrix} 1 & -1 & -3 \\ 0 & -1 & 2 \\ 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} -75 \\ 43 \\ -18 \end{pmatrix}$$

3- On choisit la ligne 2 comme ligne pivot.

On divise cette ligne par b(2,2) = -1, on obtient $\rightarrow$ c(2,:) = b(2,:)/

$\rightarrow$ nouvelle ligne = 0 1 -2 -43.

On annule le terme b(1,2) de la 1ère ligne et le terme b(3,2) de la ligne 3 :

c(1,:) = b(1,:) - c(2,:) * b(1,2) $\rightarrow$ 1 0 -5 -118

c(3,:) = b(3,:) - c(2,:) * b(3,2) $\rightarrow$ 0 0 1 25

4) On choisit la ligne 3 comme ligne pivot c(3,:).

On divise cette ligne par c(3,3) = 1, on obtient :

$\rightarrow d(3,:) = c(3,:)/c(3,3)$ $\rightarrow$ nouvelle ligne = 0, 0 1 25.

On annule dans la ligne 1 c(1,3) et dans la ligne 2 c(2,3)

$\rightarrow d(1,:) = c(1,:) - d(3,:) * c(1,3) \longrightarrow$ 1 0 0 7

$d(2,:) = c(2,:) - d(3,:) * c(2,3) \longrightarrow$ 0 1 0 7

D'où la matrice d' s'écrit :

$$d = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 7 \\ 7 \\ 25 \end{pmatrix}$$

et la solution est :

$x_1 = 7$, $x_2 = 7$, $x_3 = 25$.

4- Methode itératives :

a) Methode de Jacobie : Resoudre le système suivant par la methode itérative de Jacobie.

$$\begin{cases} 5x_1 - x_2 - x_3 - x_4 = -4 \\ -x_1 + 10x_2 - x_3 - x_4 = 12 \\ -x_1 - x_2 + 5x_3 - x_4 = 8 \\ -x_1 - x_2 - x_3 + 10x_4 = 34 \end{cases} \Rightarrow \begin{pmatrix} 5 & -1 & -1 & -1 \\ -1 & 10 & -1 & -1 \\ -1 & -1 & 5 & -1 \\ -1 & -1 & -1 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -4 \\ 12 \\ 8 \\ 34 \end{pmatrix}$$

soit a résoudre le système : $Ax = b$ où $A = (a_{ij})$ est une matrice carré d'ordre n et b le vecteur de composante b_i.

En supposant les élément diagonaux non nuls $a_{ij} \neq 0$.

On decompose la matrice A en $A = D - E - F$ ou $D = (a_{ii})$ est une matrice diagonale de A, $E = (-a_{ij})$; $j < i$, $i = 1, \dots n$ est la matrice triangulaire inferieur stricte deduit de A et $F = (-a_{ij})$; $i < j$, $j = 1, 2, \dots n$ est la matrice triangulaire supérieur stricte deduire de A.

2) on cherche la matrice de Jacobie qui s'ecrit sous la forme suivante : $J = D^{-1}(E+F)$.

de forme explicite :

$$J = \begin{pmatrix} 0 & -a_{12}/a_{11} & -\dfrac{a_{13}}{a_{11}} & \dots & -a_{1n}/a_{11} \\ -a_{21}/a_{22} & 0 & -a_{23}/a_{22} & \dots & -a_{2n}/a_{22} \\ - & - & - & - & - \\ -\dfrac{a_{n1}}{a_{nn}} & -\dfrac{a_{n2}}{a_{nn}} & -\dfrac{a_{n3}}{a_{nn}} & \dots & 0 \end{pmatrix}$$

3) en notant par $x_i^{(P+1)}$; $1 \leq i \leq n$, des composante de $(P+1)$ ième itération : L'algorithme de système est le suivant :

$$x_i^{(P+1)} = -\sum_{j=1}^{n} \left(\frac{a_{ij}}{a_{ii}}\right) x_j^{P} + \sum \frac{b_i}{a_{ii}} \; ; \quad a(i,i) \neq 0$$

solution :

La forme matricielle de l'algorithme de Jacobi est :

$$x^{(p+1)} = \begin{pmatrix} 0 & 1/5 & 1/5 & 1/5 \\ \frac{1}{10} & 0 & \frac{1}{10} & \frac{1}{10} \\ 1/5 & \frac{1}{5} & 0 & \frac{1}{5} \\ 1/10 & \frac{1}{10} & \frac{1}{10} & 0 \end{pmatrix} x^{(p)} + \begin{pmatrix} -4/5 \\ 12/10 \\ 8/5 \\ 34/10 \end{pmatrix}$$

en partant du vecteur initial nul. $(p=0)$.

$$x^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

La solution du système est le suivant :

p	0	1	2	3	4	5	6	7	- - - -
x_1^p	0	-0,8	0,44	0,716	0,8828	0,94788	0,977276	0,9900188	- - - -
x_2^p	0	1,2	1,62	1,84	1,929	1,96912	1,986474	1,9940896	- - -
x_3^p	0	1,6	2,36	2,732	2,8796	2,94812	2,977148	2,9900444	- - -
x_4^p	0	3,4	3,6	3,842	3,9288	3,96914	3,986472	3,9940898	- - -

La solution du système est donc :

$x_1 = 1$, $x_2 = 2$, $x_3 = 3$, $x_4 = 4$.

Chapitre 2 : Polynômes et interpolation polynomiale
Resolution des eqt non linéaires.

1) Introduction :

l'objectif principal de l'interpolation est d'interpoler des données connues à partir des points discrets. Dans ce cas, la valeur de la fct entre ces points peut être estimée. Cette methode d'estimation peut être étendue et utilisée dans divers domaines, à savoir la dérivation et l'integration numerique des polynômes.

2) Opérations sur des polynômes dans Matlab :

Dans Matlab, les polynomes sont représentés sous forme de vecteurs lignes dont les composantes sont données par ordre des puissances décroissantes. Un polynome de degré n est représenté par un vecteur de taille (n+1).

Exemple : le polynome : $f(x) = 8x^5 + 2x^3 - 3x^2 + 4x - 2$ est représenté par :

```
>> f = [8 0 2 -3 4 -2]
```

2.1- Multiplication des polynômes :

la fonction "Conv" donne le produit de convolution de deux polynômes. l'exemple suivant montre l'utilisation de cette fonction.

Soient
$$\begin{cases} f(x) = 3x^3 + 2x^2 - x + 4 \\ g(x) = 2x^4 - 3x^2 + 5x - 1 \end{cases}$$

le produit de convolution : $h(x) = f(x) \cdot g(x)$ est donné par :

```
>> f = [3 2 -1 4]
>> g = [2 0 -3 5]
>> h = Conv(f,g)  —> h = 6 4 -11 17 10 -19 21 -4
```

d'où $h(x) = 6x^7 + 4x^6 - 11x^5 + 17x^4 + 10x^3 - 19x^2 + 21x$

2.2- Division des polynômes :

la f^{ct} "deconv" donne le rapport de convolution de deux polynômes (deconvolution des coefficients du polynôme). L'exemple suivant montre l'utilisation de cette f^{ct}.

soient les même f^{ct} précédentes $f(x)$ et $g(x)$.

la division de $g(x)$ par $f(x)$ est donnée par la f^{ct} "deconv"

>> h = deconv (g, f) → h = 0.6667 - 0,4444

et le polynôme $h(x)$ obtenue est : $h(x) = 0,6667x - 0,4444$.

3- Interpolation :

Dans le domaine de l'analyse numérique des données, on a souvent besoin d'établir un modèle mathématique liant plusieurs series de données expérimentales. L'interpolation polynomiale consiste à approcher la courbe liant des deux séries de mesures par un polynome.

Une interpolation consiste à relier les points expérimentaux par une courbe sous forme de segments de droites ou de courbes polynomiales. Ceci peut être realisé par plusieurs methode parmais ces methodes "methode de Lagrange".

3.1. Methode de Lagrange :

Soit une f^{ct} $f(x)$ dont on connait $(n+1)$ valeurs $f_0, f_1, \ldots f_n$ aux points $x_0, x_1, \ldots x_n$. le polynome d'interpolation de lagrange $P_n(x)$ de degré n est donné par :

$$P_n(x) = \sum_{k=0}^{n} L_k(x)\, f_k \qquad \text{---(1)}$$

où les $(n+1)$ polynomes $L_k(x)$, $k = 0, \ldots, n$, tous de degré inferieur ou egal à n sont données par : $L_k(x) = \prod_{\substack{j=0 \\ j \neq k}} \frac{(x - x_j)}{(x_k - x_j)}$ --(2)

et vérifient la relation : $l_k(x_i) = \delta_{ki}$

exemple :

Déterminer le polynôme d'interpolation, de degré inférieur ou égal à 3, de la f^{ct} passant par les points :

x	-1	2	4	5
$f(x)$	-2	43	213	376

par la méthode de Lagrange.

solution : si l'on adopte la numérotation suivante pour les points d'interpolation :

k	0	1	2	3
x_k	-1	2	4	5
f_k	-2	43	213	376

il vient :

$$L_0(x) = \frac{(x-2)(x-4)(x-5)}{(-3)(-5)(-6)} = \frac{1}{90}(x^3 - 11x^2 + 38x - 40).$$

$$L_1(x) = \frac{(x+1)(x-4)(x-5)}{3(-2)(-3)} = \frac{1}{18}(x^3 - 8x^2 + 11x + 20)$$

$$L_2(x) = \frac{(x+1)(x-2)(x-5)}{5(-2)(-1)} = \frac{-1}{10}(x^3 - 6x^2 + 3x + 10).$$

$$L_3(x) = \frac{(x+1)(x-2)(x-4)}{6.(-3)(-1)} = \frac{1}{18}(x^3 - 5x^2 + 2x + 8).$$

On en déduit que le polynôme d'interpolation, de degré inférieur ou égal à 3 passant par les points données s'écrit :

$$P(x) = \sum_{k=0}^{3} L_k f_k = -2L_0(x) + 43L_1(x) + 213L_2(x) + 376L_3(x) =$$

$$P_3(x) = 2x^3 + 4x^2 + 5x + 1.$$

Chapitre 3 :

Intégration numérique des fonctions.

1) Introduction :

Dans la plupart des cas, les fonction analytiques, du fait de leurs complexités, ne sont pas intégrables analytiquement. Dans d'autre cas, on a des fonctions qui sont évaluées numériquement en différents points de l'intervalle où ces dernières sont données, et l'integrale de ces types de fonctions ne peut être obtenue que par des approches numériques. Dans ce chapitre, on s'intéressese au méthe la méthe utilisées fréquemment : à savoir la méthode des Trapèzes, de Simpson ....etc.

2) Methode d'integrations numériques :

2.1. Methode de trapèzes :

soit $f(x)$ la f^{ct} à intégrer sur $[a,b]$. l'integrale I de $f(x)$ s'écrit en utilisant la méthode des Trapèzes :

$$I = \int_a^b f(x)\,dx = \frac{h}{2}\left(f_1 + 2f_2 + 2f_3 + \cdots + 2f_i + \cdots + 2f_n + f_{n+1}\right)$$

d'où :

$$\boxed{I = \frac{h}{2}\left(f(x_1) + f(x_{n+1}) + 2\sum_{i=2}^{n} f(x_i)\right)}$$

où : $h = \dfrac{b-a}{n}$; $x_i = a + (i-1)h$; $f_i = f(x_i)$ et $i = 1, 2, \ldots$

2.2. Methode de simpson :

soit I l'integrale de $f(x)$ sur l'intervalle $[a,b]$. par la méthode de simpson, I s'écrit :

$$I = \int_a^b f(x)\,dx = \frac{h}{3}\left(f_1 + 4f_2 + f_3\right) + \frac{h}{3}\left(f_3 + 4f_4 + f_5\right) + \frac{h}{3}\left(f_5 + 4f_6 + f_7\right)$$

$$I = \int f(x)\,dx = \frac{h}{3}\left[f(1) + f(n+1) + 4\sum_{\substack{i=2 \\ i \to \text{pair}}}^{n-1} f(i) + 2\sum_{\substack{i=3 \\ i \to \text{impaire}}} f(i)\right.$$

$$\left[f(0) + f(n) + 4\sum_{0}^{n-1} f_{2i+1} + 2\sum_{i=1}^{n-1} f(2i)\right]$$

$x(i) = a + \frac{i-1}{h}$, $f(i) = f(x_i)$, $h = \frac{b-a}{n}$.

exemple:

calculer l'integral $I = \int_0^3 \frac{dx}{1+x^2}$ par la methode de Trapèze et la methode de Simpson, pour $n = 10$.

solution:

pour $n = 10$, $h = 0,3$.

i	x_i	f_i
0	0	1
1	0,3	0,917431193
2	0,6	0.735294118
3	0,9	0,552486188
4	1,2	0,40983606
5	1,5	0,307692308
6	1,8	0,235849057
7	2,1	0,184842884
8	2,4	0,147928994
9	2,7	0,120627262
10	3,0	0,1

On obtient alors:

par la formule de Trapèze : $I = 1,248596421$

" " de Simpson : $I = 1,249013158$

alors que la Valeur exacte: $\int_0^3 \frac{1}{x^2+1}\,dx = [\text{arctg}\, x]_0^3 = 1,249045772$

Chapitre 4 : Resolution numérique des équations différentielles et des équations aux dérivées partielles.

1- Introduction : le comportement dynamique des système est un sujet très important en physique. En général, les éqt utilisées pour décrire de tels comportements dynamiques, incluent des quantités inconnues représentant des fonctions recherchées et leurs dérivées.
Une equation qui comporte une ou plusieurs dérivées de la fonction inconnue est appelée "eqt différentielle" qui est représentée dans Matl par l'abréviation "ODE"

Les eqt diff peuvent être classée en deux catégories : les eqt diff av des cdt initialle et les equat diff avec des condition aux limites.

2. Equation diff du 1er ordre.

Les eqt du 1er ordre à Conditions initiales peuvent être écrite sous la forme :

$$\begin{cases} y'(t) = f(y,t) \\ y(0) = y_0 \end{cases}$$

$y(0) = y_0$: la cdt initial.

2.1. Methode d'Euler :

la formule d'Euler s'ecrit

$$y_{n+1} = y_n + h f(x_n, y_n). \qquad n = 0, 1, \ldots$$

tq : h : c'est le pas de descritisation.

Exemple 1 :

~~Soit la eqt~~ ~~$y' = \frac{y}{1}$~~

soit l'eqt différentielle : $y' = \dfrac{y}{1+x^2}$ avec $y(0) = 1$
et $x \in [0, 1]$.

Résoudre cette equation par la methode Euler. par pas de $h = 0{,}2$.

Solution : soit $f(x,y) = \frac{y}{1+x^2}$ et $h = 0,2$ et $x \in [0,1]$

$y(0) = 1$

n	x_n	y_n	Sol exact
0	0	1	1
1	0,2	1,2	1,2182258
2	0,4	1,4307692301	1,4630252
3	0,6	1,677453581	1,7167268
4	0,8	1,924137931	1,9635243
5	1	2,158788898	2,193280

$$y_{n+1} = y_n + h \cdot f(x_n, y_n)$$

$$y_{0+1} = y_0 + 0,2\, f(x_0, y_0)$$

La solution exacte. $y_{exact} = e^{Arctg(x)}$

2.2 - méthode de Runge - Kutta d'ordre 2 :

la formule d'itération de Runge-Kutta d'ordre 2 s'écrit :

$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2).$$

avec : $k_1 = h\, f(x_n, y_n)$

$k_2 = h\, f(x_n + h,\ y_n + k_1).$

2.3 - methode de Runge - Kutta d'ordre 4 :

la formule de Runge-Kutta d'ordre 4 s'écrit :

la formule de Runge-Kutta d'ordre 4 s'écrit :

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \text{ avec}$$

$k_1 = h\, f(x_n, y_n)$

$k_2 = h\, f\left(x_n + \frac{h}{2},\ y_n + \frac{k_1}{2}\right)$

$k_3 = h\, f\left(x_n + \frac{h}{2},\ y_n + \frac{k_2}{2}\right).$

$k_3 = h\, f(x_n + h,\ y_n + k_3).$

exemple :

on considère l'eq^{t} différentielle $y' = -2y$, $y(0) = 1$ dont la solution exacte est $y = e^{-2x}$.

Résoudre cette équation par la méthode de Runge-Kutta d'ordre 2, puis d'ordre 4, entre 0 et 0,05 par pas de 0,01.

solution :

1/ La formule d'itération de Rung-Kutta d'ordre 2 s'écrit :

$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2) \text{ avec : } \begin{cases} k_1 = h f(x_n, y_n) \\ k_2 = h f(x_n + h, y_n + k_1) \end{cases}$$

Dans notre cas $f(x,y) = -2y$ est donc :

$k_1 = -2 h y_n$

$k_2 = -2h(y_n + k_1) = -2h(y_n - 2hy_n) = y_n(-2h + 4h^2)$.

et comme $h = 0{,}01$ $\boxed{y_{n+1} = 0980 2\, y_n}$

• La formule de Runge Kutta d'ordre 4 s'écrit :

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \text{ avec : } \begin{cases} k_1 = h f(x_n, y_n) \\ k_2 = h f(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 = h f(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_3 = h f(x_n + h, y_n + k_3) \end{cases}$$

Dans notre cas, ces expressions deviennent :

$k_1 = -2 h y_n$

$k_2 = y_n(-2h + 2h^2)$

$k_3 = y_n(-2h + 2h^2 - 2h^3)$

$k_4 = y_n(-2h + 4h^2 - 4h^3 + 4h^4)$.

d'où : $y_{n+1} = y_n(1 - 2h + 2h^2 - \frac{4}{3}h^3 + \frac{4}{3}h^4) = 0{,}98019868\, y_n$

Les résultats obtenus sont regroupé dans le table ci-dessous :

x_n	y_n exact	y_n par R-K d'ordre 2	y_n par R-K d'ordre 4
0	1	1	1
0,01	0,98019867340	0,9802	0,98019868
0,02	0,9607894392	0,96079204	0,9607894522
0,03	0,9417645336	0,9417683576	0,9417645529
0,04	0,9231163465	0,9231213441	0,9231163716
0,05	0,9048374182	0,9048435414	0,9048374489