



Youcef RAACH

youcef.raach@univ-djelfa.dz

TP Méthodes Numériques

Avant-propos

Ce polycopié, intitulé « **TP Méthodes Numériques** », est destiné aux étudiants de deuxième année Sciences et Technologie (ST). Il a pour objectif principal de compléter les connaissances théoriques acquises en cours et en travaux dirigés, à travers des applications pratiques permettant de mieux comprendre et maîtriser les principales méthodes numériques.

Les travaux pratiques proposés visent à familiariser les étudiants avec les différentes techniques de résolution numérique, telles que la résolution des équations non linéaires, la résolution des systèmes d'équations linéaires, l'interpolation polynomiale, l'approximation des fonctions ainsi que les méthodes numériques d'intégration et de dérivation, selon le programme enseigné. Une attention particulière est accordée à la mise en œuvre des algorithmes, à l'interprétation des résultats obtenus et à l'étude des erreurs liées aux calculs numériques en utilisant le langage de programmation Matlab.

Sommaire

Chapitre 01:Généralités sur Matlab

Fenêtres principales du Matlab.....	01
Commandes de base Matlab.....	03

Chapitre 02: Résolution des équations non linéaires

Méthode de bisection (Méthode de dichotomie).....	10
Méthode de Newton-Raphson.....	13
Méthode du point fixe.....	16

Chapitre 03:L'intégration numérique

Méthode des rectangles.....	21
Méthode des trapèzes.....	24
Méthode des Simpson.....	26

Chapitre 04:Resolution des équations différentielles ordinaires

Méthode d'Euler.....	30
Méthode de Heun (Euler améliorée).....	33
Méthode Runge-Kutta	35

Chapitre 05 : Interpolation polynomiale

Méthode de Lagrange.....	39
Méthode de Newton (Méthode des différences divisées).....	42
Références bibliographiques.....	44

Chapitre 01:

Généralités Sur

Matlab

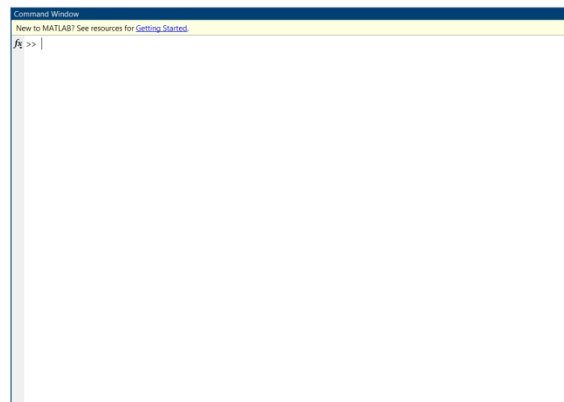
Introduction

MATLAB (MATrix LABoratory) est un environnement de programmation et de calcul numérique développé par MathWorks. Il est largement utilisé dans les domaines scientifiques, techniques et industriels pour l'analyse de données, la modélisation, la simulation et le développement d'algorithmes. Grâce à ses outils performants, MATLAB permet de réaliser rapidement des calculs complexes, des visualisations graphiques et des simulations.

+ Fenêtres principales du Matlab

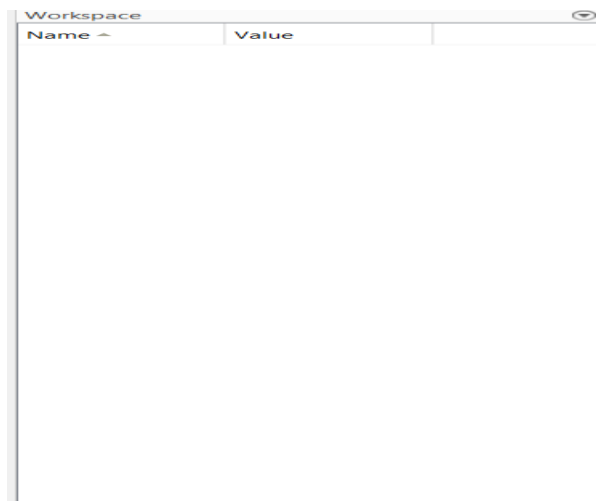
Command Window

Permet d'exécuter directement les commandes MATLAB.



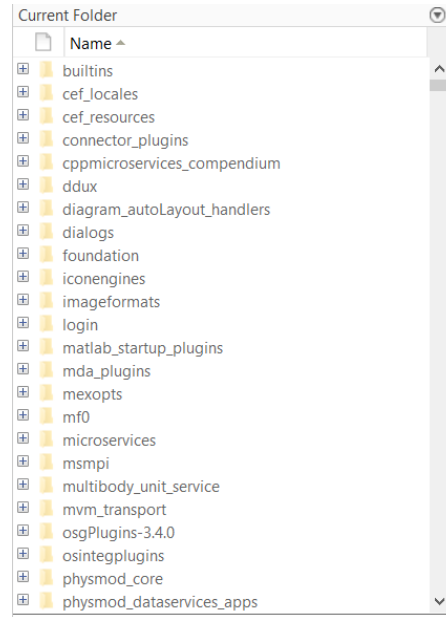
Workspace

Contient toutes les variables créées durant la session.



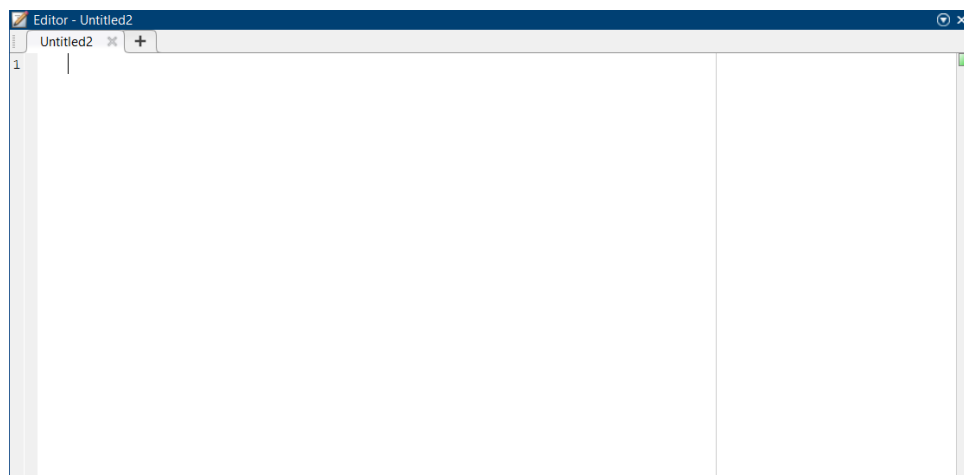
Current Folder

Affiche les fichiers du répertoire de travail.



Editeur

Permet de créer et modifier des scripts et fonctions.



Commandes de base MATLAB

Opérations arithmétiques

```
Command Window
New to MATLAB? See resources for Getting Started.
>> a=5;
>> b=2;
>> c=a+b

c =

     7

>> v=a*b

v =

    10

>> y=a/b

y =

    2.5000

fx >> |
```

Vecteurs

```
Command Window
New to MATLAB? See resources for Getting Started.
>> v=[ 1 5 6 8 10]

v =

     1     5     6     8    10

>> v=[ 1;5;6;8;10]

v =

     1
     5
     6
     8
    10

>> v=[ 1 5 6 8 10]';

v =

     1
     5
     6
     8
    10

fx >> |
```

Matrices

```
Command Window
New to MATLAB? See resources for Getting Started.

>> A=[ 2 3 5 7 9
      10 8 9 6 3
      -2 4 2 3 -3
      0 2 3 -6 10
      -8 9 2 0 3]

A =

     2     3     5     7     9
    10     8     9     6     3
    -2     4     2     3    -3
     0     2     3    -6    10
    -8     9     2     0     3

>> B=[ 1 3 5 3 9
      -2 8 3 6 0
      -4 8 12 13 -4
      1 2 1 -4 13
      -1 4 6 0 13]

B =

     1     3     5     3     9
    -2     8     3     6     0
    -4     8    12    13    -4
     1     2     1    -4    13
    -1     4     6     0    13

fx >> |
```

Opérations sur les matrices

```
Command Window
New to MATLAB? See resources for Getting Started.

>> l=A+B

l =

     3     6    10    10    18
     8    16    12    12     3
    -6    12    14    16    -7
     1     4     4   -10    23
    -9    13     8     0    16

>> q=A*B

q =

   -26    120    140     61    206
   -39    190    206    171    171
   -12     36     11     32    -26
   -32     68     96     75     40
   -37     76     29     56    -41

>> b=A/B

b =

    2.6618    0.7938   -1.4135   -3.1567    1.5713
   -0.8124   -4.4388    9.2427   21.2713  -17.6343
   -0.6640    0.2264    0.5303    0.8152   -0.4230
   -1.8517   -1.3348    2.2876    5.5438   -2.7888
   -1.9697    1.9770   -1.5775   -3.6388    4.7478

fx
```

Fonctions matricielles

```
Command Window
>> zeros(3,3)

ans =

    0    0    0
    0    0    0
    0    0    0

>> ones(3,3)

ans =

    1    1    1
    1    1    1
    1    1    1

>> eye(3,3)

ans =

    1    0    0
    0    1    0
    0    0    1

>> rand(3,3)

ans =

    0.0975    0.9575    0.9706
    0.2785    0.9649    0.9572
    0.5469    0.1576    0.4854
```

```
>> A(2,4)

ans =

    6

>> A(2,:)

ans =

    10     8     9     6     3

>> A(:,3)

ans =

    5
    9
    2
    3
    2
```

Fonctions mathématiques

```
>> sqrt(16)

ans =

     4

>> abs(-5)

ans =

     5

>> sin(pi/2)

ans =

     1

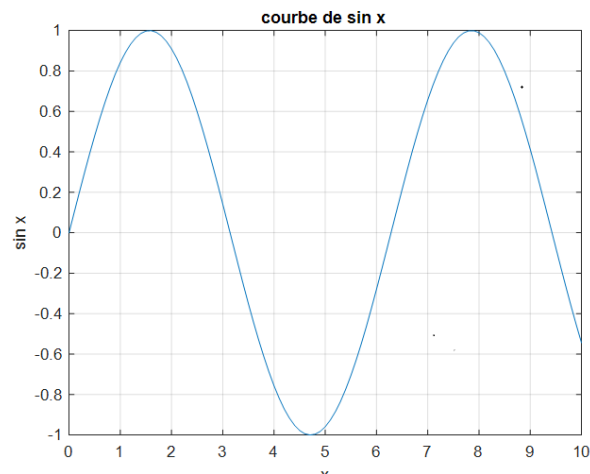
>> cos(pi)

ans =

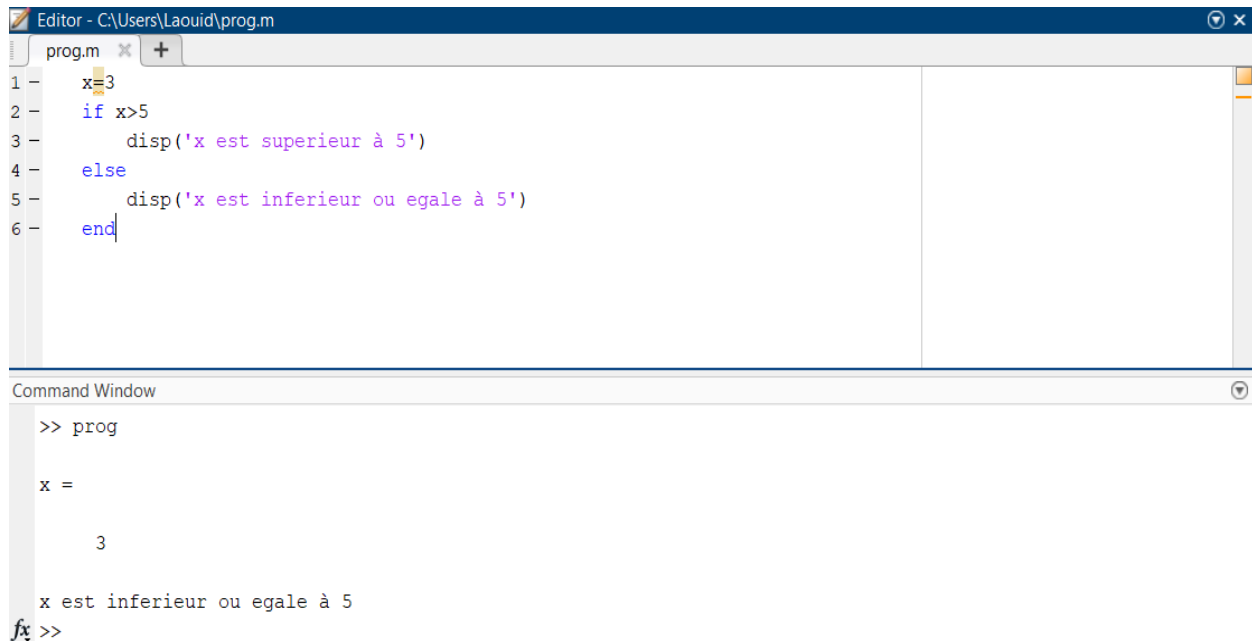
    -1
```

Tracé de graphiques

```
x=0:0.1:10;
y=sin(x);
plot(x,y);
grid on
title('courbe de sin x')
xlabel('x')
ylabel('sin x')
```



Structures conditionnelles



The screenshot shows the MATLAB Editor window with a file named 'prog.m'. The code in the editor is as follows:

```
1 - x=3
2 - if x>5
3 -     disp('x est superieur à 5')
4 - else
5 -     disp('x est inferieur ou egale à 5')
6 - end
```

Below the editor is the Command Window, which shows the execution of the 'prog' script:

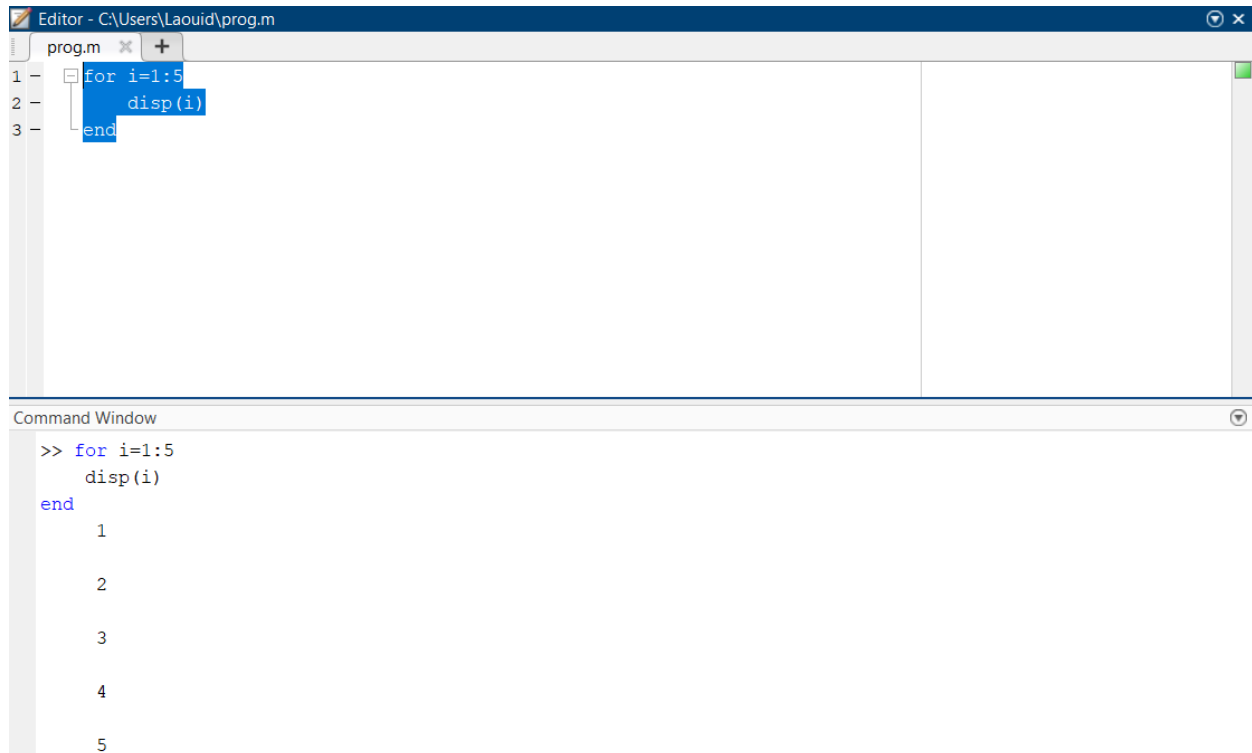
```
>> prog

x =

     3

x est inferieur ou egale à 5
fx >>
```

Boucle for



The screenshot shows the MATLAB Editor window with a file named 'prog.m'. The code in the editor is as follows:

```
1 - for i=1:5
2 -     disp(i)
3 - end
```

Below the editor is the Command Window, which shows the execution of the 'for' loop:

```
>> for i=1:5
    disp(i)
end

1

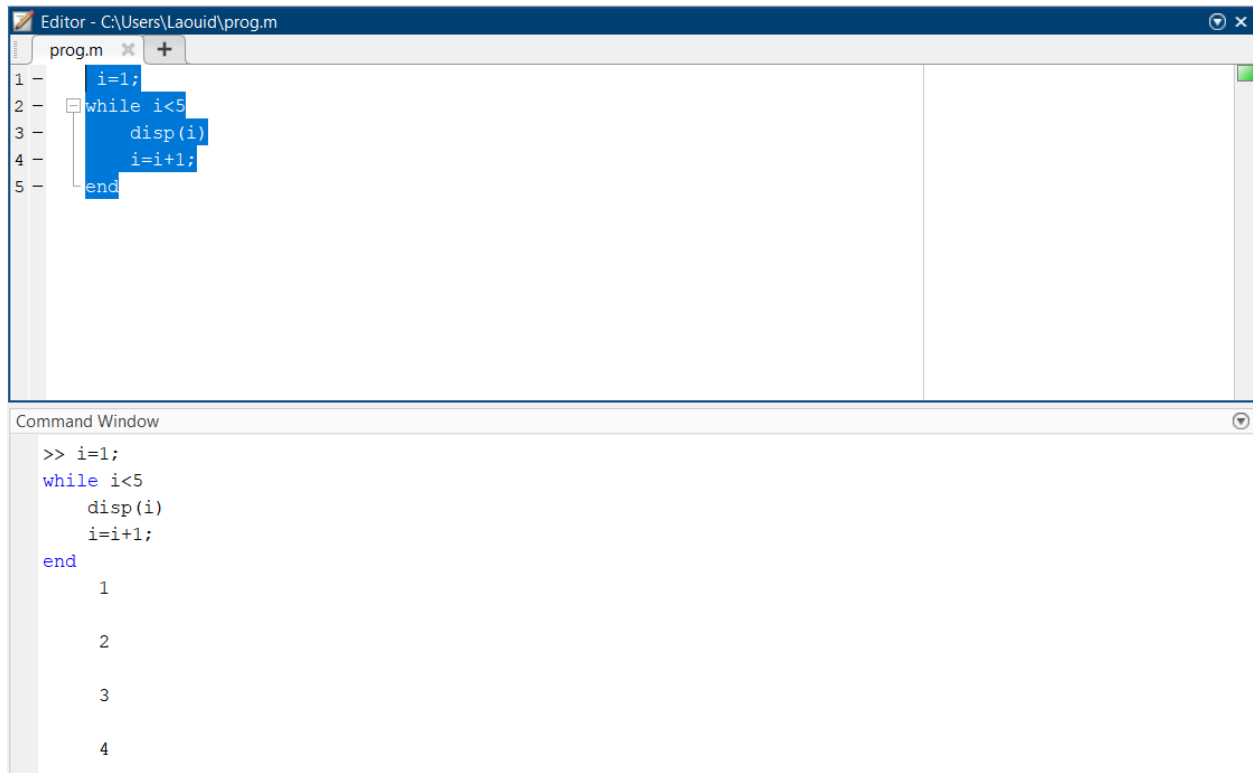
2

3

4

5
```

Boucle while



The screenshot shows the MATLAB environment. The Editor window displays a script named 'prog.m' with the following code:

```
1 i=1;
2 while i<5
3     disp(i)
4     i=i+1;
5 end
```

The Command Window shows the execution of the script, displaying the output:

```
>> i=1;
while i<5
    disp(i)
    i=i+1;
end
1
2
3
4
```

Nettoyage de l'environnement

Pour nettoyer l'environnement, on utilise les commandes : Clc, Clear et Close all.

Aide MATLAB

En cas de besoin d'aide des commande Matlab, les commandes ***help plot*** et ***doc plot*** sont recommandés.

Chapitre 02:
Résolution des
équations non
linéaires

Méthode de bisection (Méthode de dichotomie)

Introduction

La méthode de bisection, également appelée méthode de dichotomie, est une méthode numérique utilisée pour résoudre les équations non linéaires de la forme :

$$f(x) = 0$$

Cette méthode est basée sur le théorème des valeurs intermédiaires et permet de déterminer une approximation de la racine d'une fonction continue.

Principe Théorique

Soit une fonction $f(x)$ continue sur un intervalle $[a, b]$. Si :

$$f(a) \times f(b) < 0$$

alors il existe au moins une racine r dans l'intervalle $[a, b]$. La méthode consiste à :

- Calculer le milieu de l'intervalle :

$$c = (a + b) / 2$$

- Évaluer $f(c)$
- Choisir le sous-intervalle qui contient la racine :
- Si $f(a) \times f(c) < 0$ alors la racine est dans $[a, c]$.
- Sinon la racine est dans $[c, b]$.
- Répéter le processus jusqu'à atteindre la précision souhaitée.

Exemple

Résoudre l'équation suivante en utilisant la méthode de bisection :

$$f(x) = x^3 - x - 2$$

Comme premier pas, on doit vérifier l'intervalle

Pour $x = 1$; $f(1) = -2$

Pour $x = 2$; $f(2) = 4$

Comme :

$$f(1) \times f(2) = -8 < 0$$

La racine est comprise entre 1 et 2.

Itération 1

$$c = (1+2)/2 = 1,5$$

$$f(1.5) = -0,125$$

Nouvel intervalle :

$$[1,5 ; 2]$$

Itération 2

$$c = (1,5+2)/2 = 1,75$$

$$f(1.75) = 1,609375$$

Nouvel intervalle :

$$[1,5 ; 1,75]$$

Itération 3

$$c = (1,5+1,75)/2 = 1,625$$

$$f(1.625) = 0,666016$$

Nouvel intervalle :

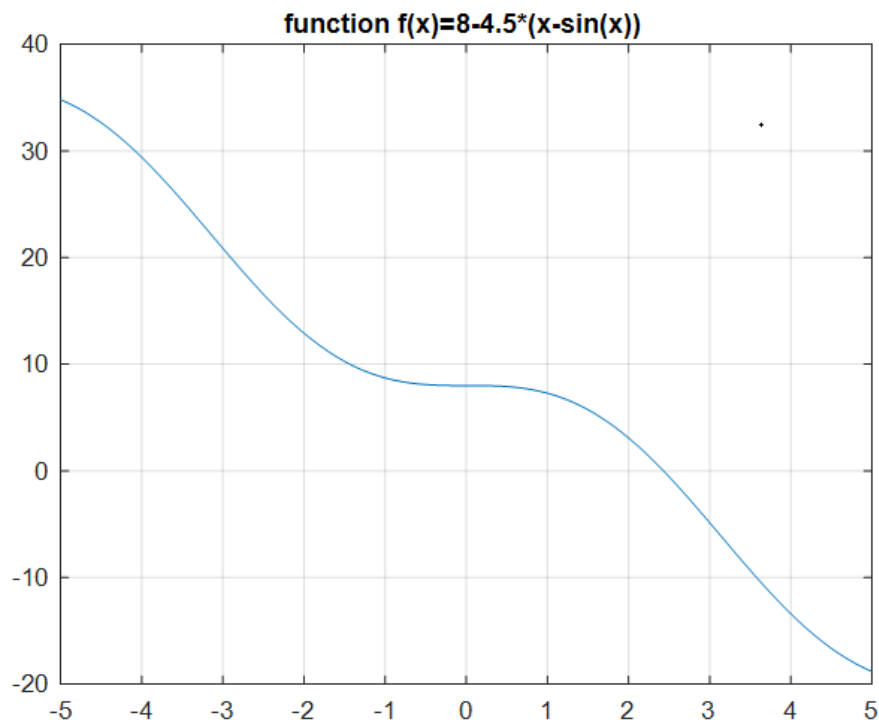
$$[1,5 ; 1,625]$$

Après plusieurs itérations :

$$x \approx 1,52138$$

Code Matlab

```
Editor - D:\teaching\licence\2\Numerical methods\2017-2018\m files\biss.m
teskimi.m x simulation_TES_serpentin.m x biss.m x biss.m x +
1 clear all, close all, clc
2 x=-5:0.1:5;
3 f=@(x) 8-4.5*(x-sin(x))
4 plot(x,f(x));grid on
5 title('function f(x)=8-4.5*(x-sin(x))');
6 format short
7 a=1;b=5;k=0;eps=1e-5;
8 disp('iteration a b Solution f(c) ')
9 while (abs(f((a+b)/2))>eps)
10
11 c=(a+b)/2;
12 k=k+1;
13 fprintf('%5i %11.8f %11.8f %11.8f %11.8f\n', k, a, b, c, abs(f(c)));
14 if (f(a)*f(c)<0)
15 b=c;
16 else
17 a=c;
18 end
19 end
20
```



Méthode de Newton-Raphson

Introduction

La méthode de Newton-Raphson est une méthode numérique utilisée pour déterminer une solution approchée d'une équation non linéaire de la forme :

$$f(x) = 0$$

Elle est basée sur le développement de la tangente à la courbe de la fonction et permet généralement d'obtenir une convergence beaucoup plus rapide que la méthode de bisection.

Principe Théorique

Soit une fonction $f(x)$ dérivable et une approximation initiale x_0 proche de la racine recherchée.

L'équation de la tangente à la courbe au point x_n est utilisée pour obtenir une meilleure approximation x_{n+1} .

La formule de récurrence de Newton-Raphson est :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

où :

- $f(x_n)$ est la valeur de la fonction.
- $f'(x_n)$ est la dérivée de la fonction.
- x_n est l'approximation courante.
- x_{n+1} est l'approximation suivante.

Le processus est répété jusqu'à ce que l'erreur devienne suffisamment faible.

Conditions de Convergence

Pour assurer une bonne convergence :

- La fonction doit être dérivable.
- La dérivée ne doit pas être nulle au voisinage de la racine.
- Le point initial doit être suffisamment proche de la solution recherchée.

Interprétation Géométrique

La méthode de Newton-Raphson consiste à :

- Tracer la tangente à la courbe $y = f(x)$ au point x_n .
- Déterminer l'intersection de cette tangente avec l'axe des abscisses.
- Cette intersection fournit une nouvelle approximation x_{n+1} de la racine.

Ainsi, la méthode progresse rapidement vers la solution.

Exemple

Déterminons une racine de l'équation ci-dessous en utilisant la méthode de Newton-Raphson :

$$f(x)=x^3 - x - 2$$

La dérivée est :

$$f'(x)=3x^2 - 1$$

Choisissons :

$$x_0 = 1.5$$

Itération 1

Calcul de la fonction :

$$f(1.5)=-0.125$$

Calcul de la dérivée :

$$f'(1.5)= 5,75$$

Application de la formule :

$$x_1= 1,5 - (-0,125 / 5,75)$$

$$x_1= 1,521739$$

Itération 2

$$f(1.521739) = 0,002137$$

$$f'(1.521739) = 5,947$$

$$x_2 = 1,521739 - (0,002137 / 5,947)$$

$$x_2 = 1,521380$$

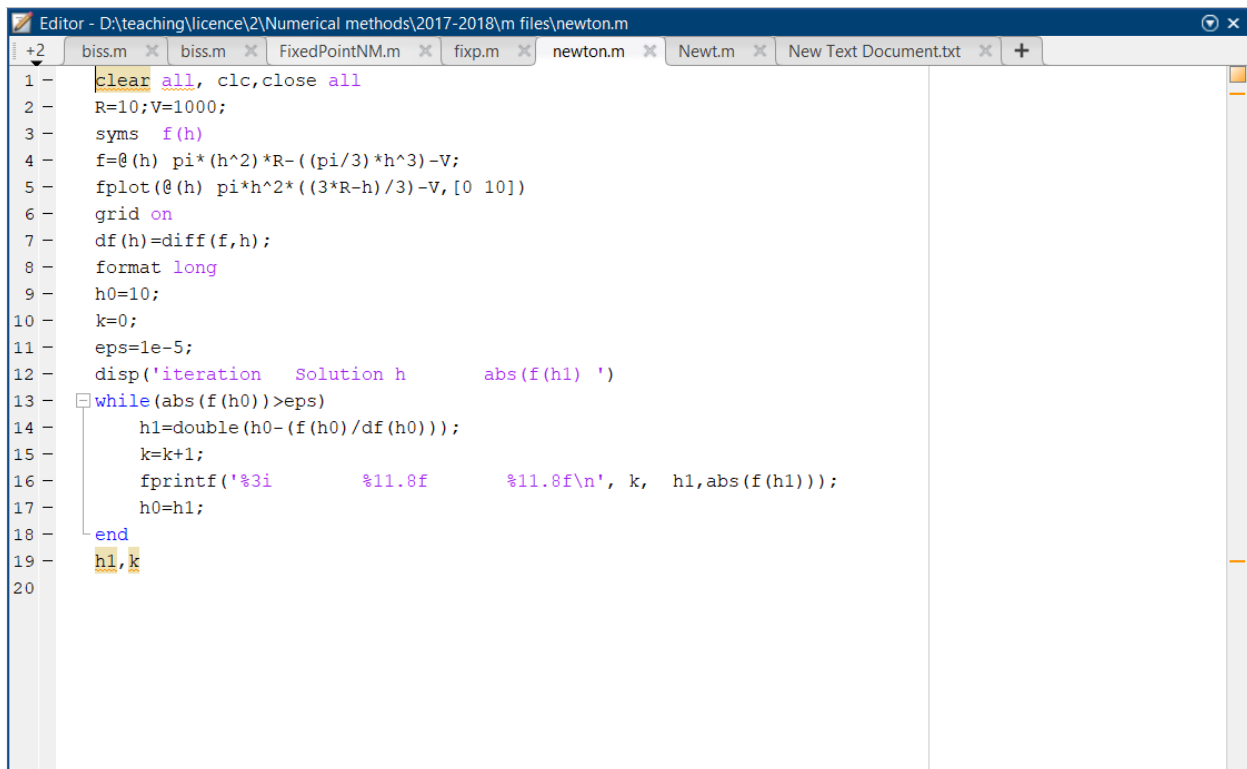
Itération 3

$$f(1.521380) \approx 0$$

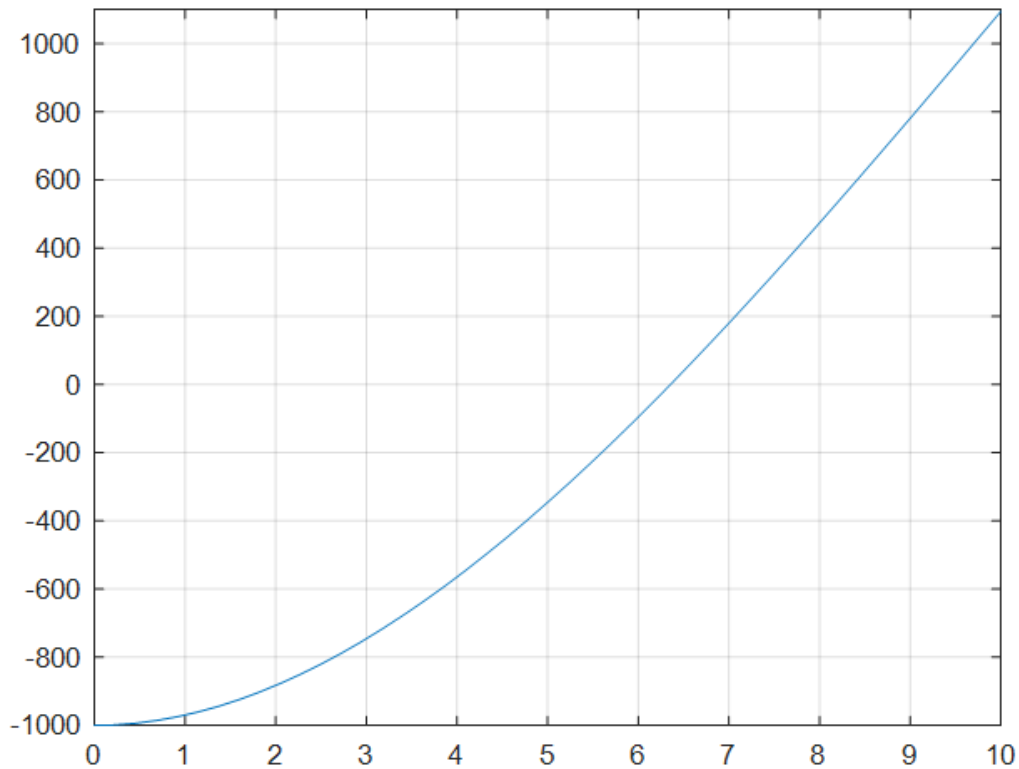
La solution est donc :

$$x \approx 1,521380$$

Code Matlab



```
Editor - D:\teaching\licence\2\Numerical methods\2017-2018\m files\newton.m
+2  biss.m  biss.m  FixedPointNM.m  fixp.m  newton.m  Newt.m  New Text Document.txt  +
1 - clear all, clc, close all
2 - R=10;V=1000;
3 - syms f(h)
4 - f=@(h) pi*(h^2)*R-((pi/3)*h^3)-V;
5 - fplot(@(h) pi*h^2*((3*R-h)/3)-V,[0 10])
6 - grid on
7 - df(h)=diff(f,h);
8 - format long
9 - h0=10;
10 - k=0;
11 - eps=1e-5;
12 - disp('iteration   Solution h       abs(f(h1)) ')
13 - while(abs(f(h0))>eps)
14 -     h1=double(h0-(f(h0)/df(h0)));
15 -     k=k+1;
16 -     fprintf('%3i           %11.8f           %11.8f\n', k, h1,abs(f(h1)));
17 -     h0=h1;
18 - end
19 - h1,k
20
```



Méthode du Point Fixe

Introduction

La méthode du point fixe est une méthode numérique utilisée pour résoudre les équations non linéaires. Elle consiste à transformer une équation de la forme $f(x) = 0$ en une équation équivalente :

$$x = g(x)$$

La solution recherchée est appelée **point fixe** de la fonction $g(x)$, car elle vérifie :

$$g(x) = x$$

Principe Théorique

Considérons l'équation : $f(x) = 0$. On la transforme sous la forme $x = g(x)$. À partir d'une approximation initiale x_0 , on construit la suite :

$$x_{n+1} = g(x_n)$$

Si cette suite converge, alors sa limite α vérifie :

$$\alpha = g(\alpha)$$

α est la solution de l'équation.

Conditions de Convergence

La méthode converge vers la solution α si :

$$|g'(\alpha)| < 1$$

Plus généralement, si :

$$|g'(x)| < 1$$

Dans un voisinage de la racine, alors la convergence est assurée. Cette condition signifie que la pente de la fonction $g(x)$ doit être suffisamment faible autour du point fixe.

Exemple

En utilisant la méthode du point fixe, résoudre l'équation suivante :

$$x^3 - x - 2 = 0$$

On peut écrire :

$$x = \sqrt[3]{x+2}$$

Ainsi :

$$g(x) = (x+2)^{1/3}$$

Choisissons :

$$x_0 = 1,5$$

Itération 1

$$x_1 = (1,5 + 2)^{(1/3)}$$

$$x_1 = 1,518294$$

Itération 2

$$x_2 = (1,518294 + 2)^{(1/3)}$$

$$x_2 = 1,520935$$

Itération 3

$$x_3 = (1,520935 + 2)^{(1/3)}$$

$$x_3 = 1,521316$$

Itération 4

$$x_4 = (1,521316 + 2)^{(1/3)}$$

$$x_4 = 1,521371$$

Itération 5

$$x_5 = (1,521371 + 2)^{(1/3)}$$

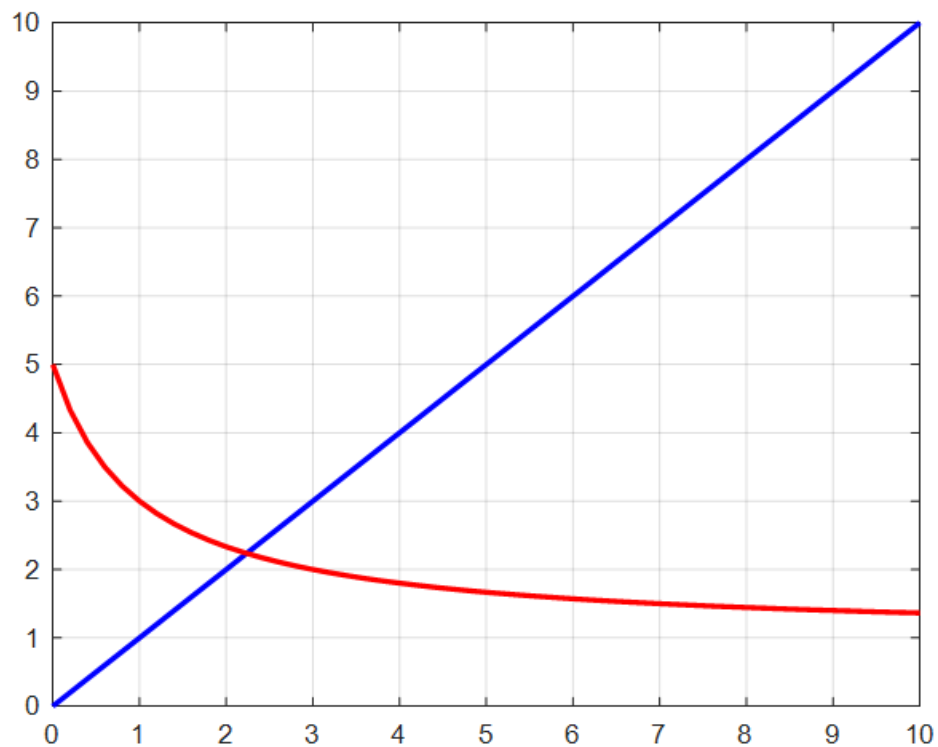
$$x_5 = 1,521379$$

La solution est donc :

$$x \approx 1,52138$$

Code Matlab

```
Editor - D:\teaching\licence\2\Numerical methods\2017-2018\m files\FixedPointNM.m
teskimi.m  simulation_TES_serpentin.m  biss.m  biss.m  FixedPointNM.m  fixp.m  +
1 - clear all;clc
2 - x=[0:0.2:10];
3 - f=@(x) x.^2-5;
4 - g=@(x) (x+5)./(x+1);
5 - ylin=x;
6 - plot(x,ylin,'b-','LineWidth',2)
7 - hold on
8 - plot(x,g(x),'r-','LineWidth',2)
9 - grid on
10 - format long
11 - x0 = 3.0; %initial guess
12 - eps = 1.0e-5;
13 - x1=g(x0)
14 - k = 0;
15 - disp('iteration    solution    f(x0)')
16 - while (abs(x1-x0)>eps)
17 -     x0 = x1;
18 -     x1=g(x0);
19 -     k = k + 1;
20 -
21 -     fprintf('%5i    %15.8f %15.8f\n', k, x1, f(x1));
22 -
23 - end
24 -
25 - x1
26 - k
```



Chapitre 03:
L'intégration
numérique

Méthode des rectangles

L'**intégration numérique** (parfois appelée quadrature numérique) est un ensemble de techniques mathématiques et algorithmiques utilisées pour calculer une valeur approximative de l'intégrale définie d'une fonction :

$$I = \int_a^b f(x) dx$$

En termes simples, cela revient à calculer l'**aire géométrique** située entre la courbe d'une fonction $f(x)$, l'axe des abscisses, et les droites verticales $x = a$ et $x = b$

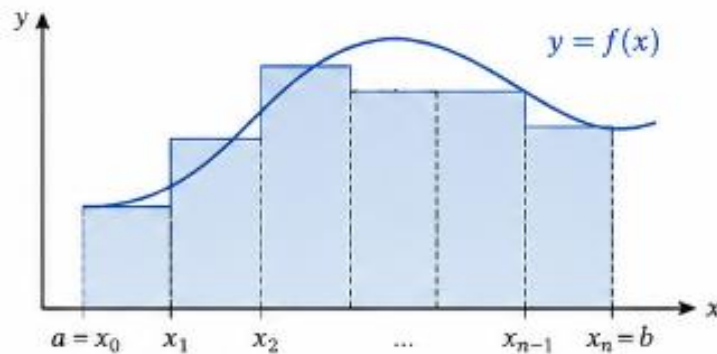


Figure 01 : Interprétation géométrique de la méthode des rectangles

Méthode des rectangles

L'idée principale est de remplacer la surface sous la courbe par une somme de rectangles. On divise l'intervalle $[a, b]$ en n sous-intervalles égaux :

$$h = \frac{b - a}{n}$$

Pour calculer la valeur de l'intégrale en utilise la relation suivante :

$$\int_a^b f(x) = h \sum_{i=0}^{n-1} f(a + ih)$$

Exemple

Calculez l'intégrale suivant par la méthode des rectangles en prenant **n=4** :

$$I = \int_0^1 x^2 dx$$

Solution

Les points seront respectivement :

$$x_0 = 0$$

$$x_1 = 0.25$$

$$x_2 = 0.5$$

$$x_3 = 0.75$$

Leurs valeurs sont

$$f(0) = 0$$

$$f(0.25) = 0.0625$$

$$f(0.5) = 0.25$$

$$f(0.75) = 0.5625$$

Donc :

$$I = 0.25(0 + 0.0625 + 0.25 + 0.5625)$$

$$I = 0.21875$$

Code Matlab

```

1 - clear, clc
2 - % f=@(x) (x.^2);
3 - f=@(x) 1./(x+2);
4 - a = -1;
5 - b = 1 ;
6 - % step size
7 - n = 120;
8 - h = (b-a)/n ; % the counter
9 - %Rectangle Method:
10 - x = a:h:b;
11 - F = f(x);
12 - Rec=(sum(F(1:n)))*h
13
14

```

Command Window

New to MATLAB? See resources for [Getting Started.](#)

Rec =

1.104188419720372

```

1 - clear, clc
2 - % f=@(x) (x.^2);
3 - f=@(x) 1./(x+2);
4 - a = -1;
5 - b = 1 ;
6 - % step size
7 - n = 8;
8 - h = (b-a)/n ; % the counter
9 - %Rectangle Method:
10 - s=0;
11 - for i = 0:n-1
12 -     xn= a + (i*h);
13 -     s = s + f(xn);
14 - end
15 - Rectangle = h * s

```

Command Window

New to MATLAB? See resources for [Getting Started.](#)

Rectangle =

1.186544011544012

✚ Méthode des Trapèzes

La méthode des trapèzes est une méthode d'intégration numérique utilisée pour approximer une intégrale définie :

$$I = \int_a^b f(x) dx$$

L'aire à calculer est toujours divisé en **n** intervalles égaux de base **h**.

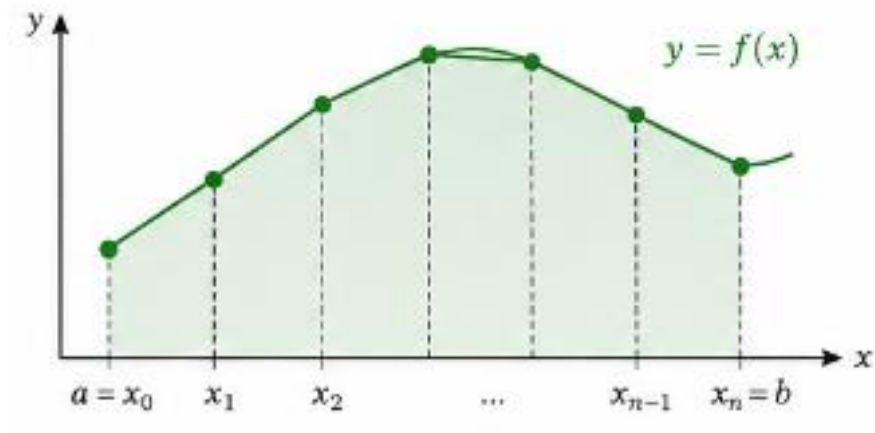


Figure 02 : Interprétation géométrique de la méthode des trapèzes

Pour calculer la valeur d'intégrale, on doit appliquer la relation suivante :

$$\int_a^b f(x) dx = \frac{h}{2} \sum_{i=0}^{n-1} f(a + ih) + f(a + ih + h)$$

Exemple

Calculer la valeur de l'intégrale suivant par la méthode des trapèzes avec **h=0.1**

$$I = \int_0^1 \frac{x}{1+x^2} dx$$

Solution

h=0.1 donc le nombre des répartitions sera :

$$n = \frac{b-a}{h} = \frac{1-0}{0.1} = 10$$

x	$f(x)$
0	0
0.1	0.0990
0.2	0.1923
0.3	0.2752
0.4	0.3448
0.5	0.4
0.6	0.4412
0.7	0.4698
0.8	0.4878
0.9	0.4972
1	0.5

$$I = \frac{0.1}{2} [(0 + 0.5) + 2 * (0.0990 + 0.1923 + \dots + 0.4878 + 0.4972)]$$

$$I = 0.3460$$

Code Matlab

```

Editor - D:\teaching\licence\2\Numerical methods\2017-2018\m files\trapez.m
+3  biss.m  FixedPointNM.m  fixp.m  newton.m  Newt.m  New Text Document.txt  symvar.m  trapez.m
1 - clear, clc
2 - % f=@(x) (x.^2);
3 - f=@(x) 1./(x+2);
4 - a = -1;
5 - b = 1 ;
6 - % step size
7 - n = 8;
8 - % step size
9 - h = (b-a)/n ; % the counter
10 - x = a:h:b;
11 - F = f(x);
12 - I = (h/2) * (F(1)+F(n+1)) + h*sum(F(2:n))
13 - % TrapEq5=(b-a)*I/(2*n)

Command Window
New to MATLAB? See resources for Getting Started.

I =

1.103210678210678
  
```

✚ Méthode des Simpson

La méthode de Simpson est une méthode d'intégration numérique utilisée pour calculer une approximation d'une intégrale définie :

$$I = \int_a^b f(x) dx$$

Contrairement aux méthodes des rectangles et des trapèzes qui utilisent respectivement des rectangles et des segments de droite, la méthode de Simpson utilise des **paraboles** pour approximer la courbe. Cette méthode est généralement beaucoup plus précise.

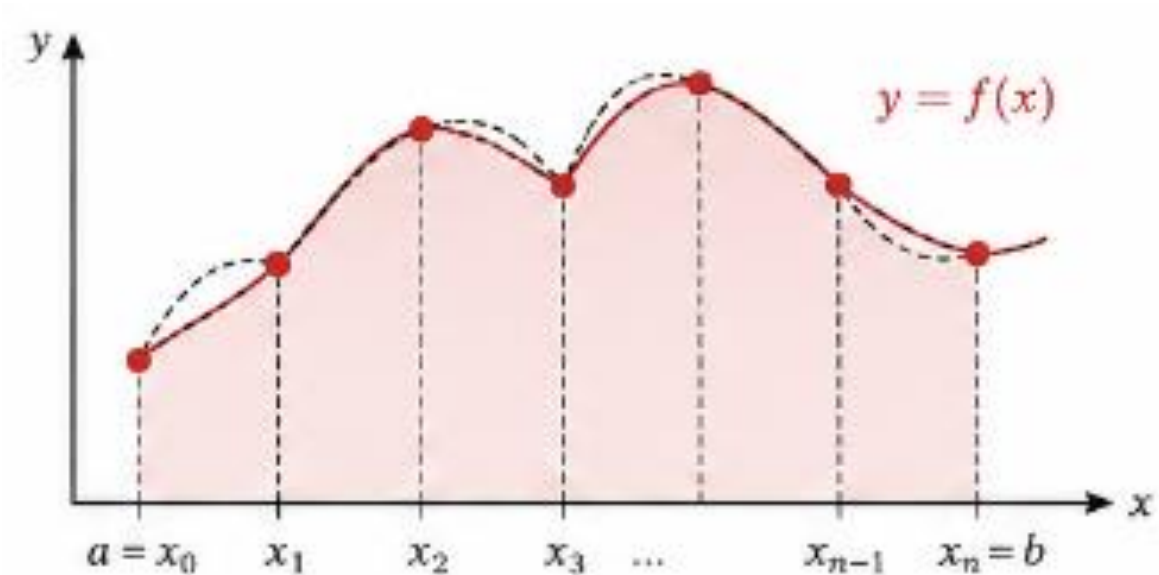


Figure 03 : Interprétation géométrique de la méthode des Simpson

Pour calculer la valeur d'intégrale, on doit appliquer la relation suivante :

$$\int_a^b f(x) dx = \frac{h}{6} \left[f(a) + f(b) + 2 \sum_{k=1}^{n-1} f(x_k) + 4 \sum_{k=1}^{n-1} \left(\frac{x_{k+1} + x_k}{2} \right) \right]$$

Exemple

Calculer la valeur de l'intégrale suivant par la méthode des trapèzes avec $h=0.1$

$$I = \int_0^1 \frac{x}{1+x^2} dx$$

Solution

$h=0.1$ donc le nombre des répartitions sera :

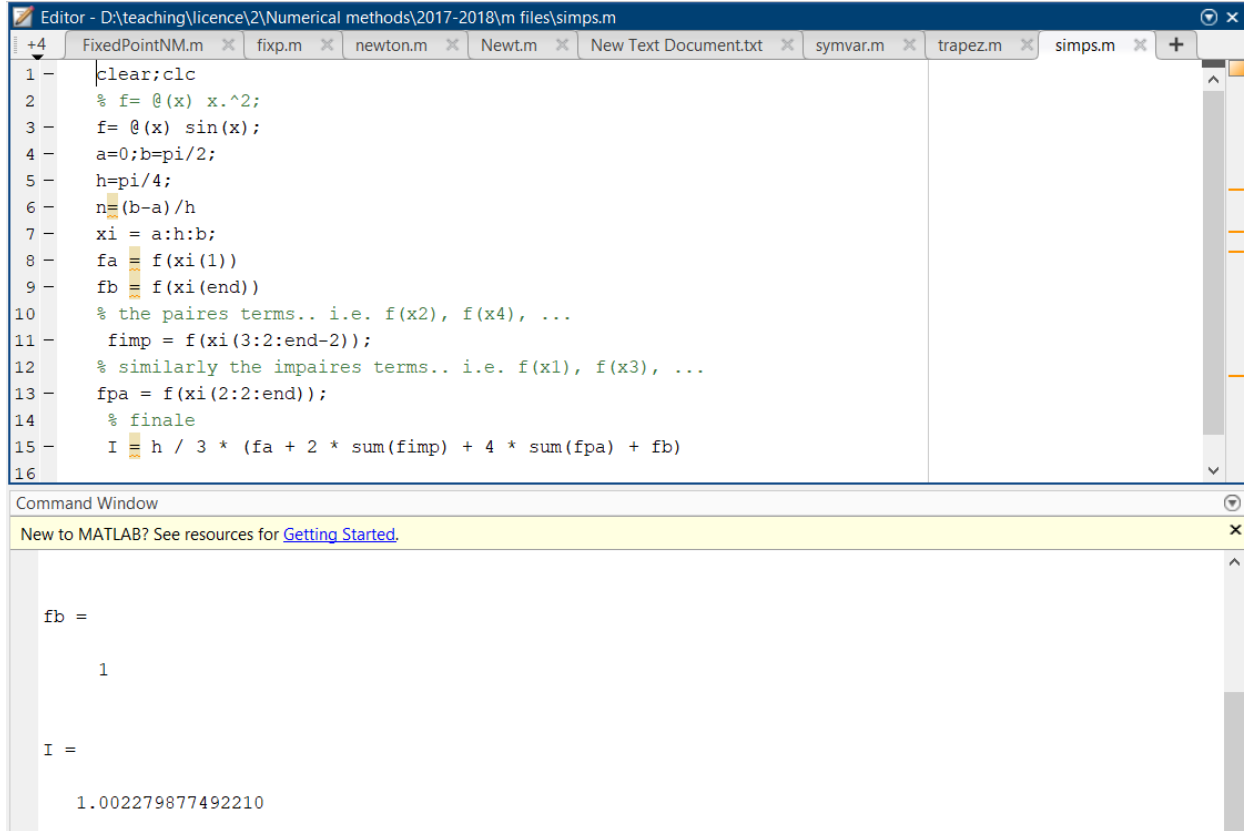
$$n = \frac{b-a}{h} = \frac{1-0}{0.1} = 10$$

x	$f(x)$
0	0
0.1	0.0990
0.2	0.1923
0.3	0.2752
0.4	0.3448
0.5	0.4
0.6	0.4412
0.7	0.4698
0.8	0.4878
0.9	0.4972
1	0.5

$$I = \frac{0.1}{6} [(0 + 0.5) + 4 * (0.0990 + 0.2752 + 0.4 + 0.4698 + 0.4972 + 2 * (0.1923 + 0.3448 + 0.4412 + 0.4878))]$$

$$I = 0.3466$$

Code Matlab



The image shows a MATLAB environment with the Editor window displaying a script named `simps.m` and the Command Window showing the results of the script execution.

Editor - D:\teaching\licence\2\Numerical methods\2017-2018\m files\simps.m

```
1 clear;clc
2 % f= @(x) x.^2;
3 f= @(x) sin(x);
4 a=0;b=pi/2;
5 h=pi/4;
6 n=(b-a)/h
7 xi = a:h:b;
8 fa = f(xi(1))
9 fb = f(xi(end))
10 % the paires terms.. i.e. f(x2), f(x4), ...
11 fimp = f(xi(3:2:end-2));
12 % similarly the impaires terms.. i.e. f(x1), f(x3), ...
13 fpa = f(xi(2:2:end));
14 % finale
15 I = h / 3 * (fa + 2 * sum(fimp) + 4 * sum(fpa) + fb)
16
```

Command Window

New to MATLAB? See resources for [Getting Started](#).

fb =
1

I =
1.002279877492210

Chapitre 04:
Résolution des
équations
différentielles
ordinaires

✚ Méthode d'Euler

Introduction

On appelle ´équation différentielle ordinaire (EDO) une relation entre une variable réelle indépendante x , une fonction inconnue $x \rightarrow f(x)$ et ses d´érivées ($y, y', y'', \dots$). L'ordre d'une EDO est défini comme étant l'ordre de la d´érivée la plus élevée figurant dans l'équation. Ainsi, une équation différentielle d'ordre n se présente sous la forme $f(x, y, y', y'', \dots, y^n) = 0$.

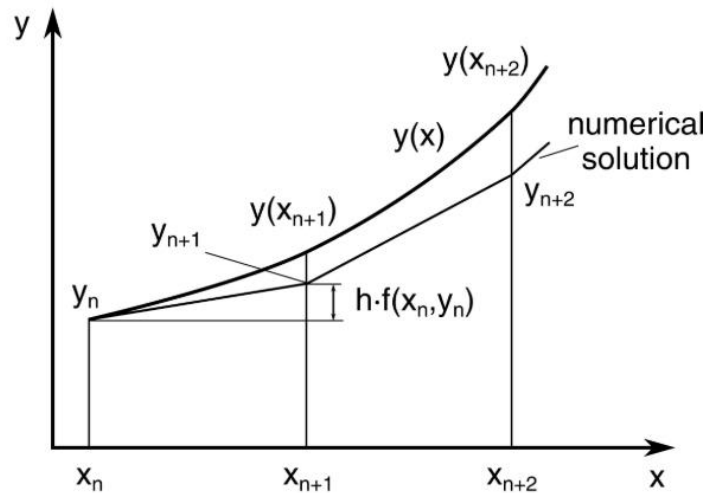


Figure 01 : Interprétation géométrique de la méthode d'Euler

Méthode d'Euler

La **méthode d'Euler** est une méthode numérique simple qui permet d'obtenir une approximation de la solution d'une équation différentielle du premier ordre. On prend le cas d'une équation différentielle du premier ordre de la forme :

$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases}$$

Où :

- $f(x, y)$ est une fonction connue ;
- x_0 est la valeur initiale de la variable indépendante ;
- y_0 est la condition initiale.

Principe de la méthode d'Euler

L'idée consiste à remplacer la courbe de la solution par une succession de petits segments de droite. À chaque étape, on utilise la pente donnée par l'équation différentielle pour calculer la valeur suivante.

Si h désigne le pas de calcul, alors :

$$x_{n+1} = x_n + h$$

Et la formule d'Euler est :

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

Où :

- x_n abscisse du point courant ;
- y_n approximation de la solution au point x_n ;
- h pas d'intégration ;
- $f(x_n, y_n)$ pente de la courbe au point (x_n, y_n) .

Exemple

Résoudre approximativement par la méthode d'Euler l'équation différentielle suivante :

$$y' = x + y ; \quad y(0) = 1 ; \quad h=0.1$$

Solution

La première étape (conditions initiales) :

$$x_0 = 0; y_0 = 1$$

La pente vaut :

$$f(x_0, y_0) = 0 + 1 = 1$$

Donc :

$$y_1 = y_0 + h * f(x_0, y_0)$$

$$y_1 = 1 + 0.1 * 1$$

$$y_1 = 1.1$$

$$(x_1, y_1) = 0.1, 1$$

La deuxième étape :

$$f(x_1, y_1) = 0.1 + 1.1 = 1.2$$

$$y_2 = y_1 + h * f(x_1, y_1)$$

$$y_2 = 1.1 + 0.1 * 1.2$$

$$y_2 = 1.22$$

La troisième étape :

$$f(x_2, y_2) = 0.2 + 1.22 = 1.42$$

$$y_3 = y_2 + h * f(x_2, y_2)$$

$$y_3 = 1.22 + 0.1 * 1.42$$

$$y_3 = 1.364$$

On peut classer les résultats dans un tableau :

n	x_n	y_n
0	0	1.000
1	0.1	1.100
2	0.2	1.220
3	0.3	1.364
4	0.4	1.5282
5	0.5	1.7210

Méthode de Heun (Euler améliorée)

C'est l'extension de la méthode classique d'Euler (version améliorée)

$y_0 =$ La valeur initiale

$$y_{n+1} = y_n + \frac{h}{2} (f(x_n, y_n) + f(x_{n+1}, y_n + h * f(x_n, y_n)))$$

Exemple :

Résoudre par la méthode de Heun l'équation différentielle suivante :

$$\begin{cases} y' = -2xy \\ y(0) = 1 \end{cases}$$

Prenez : $h = 0.1$

Solution :

$$x_1 = x_0 + h = 0.1$$

$$y_{n+1} = y_n + \frac{h}{2} (f(x_n, y_n) + f(x_{n+1}, y_n + h * f(x_n, y_n)))$$

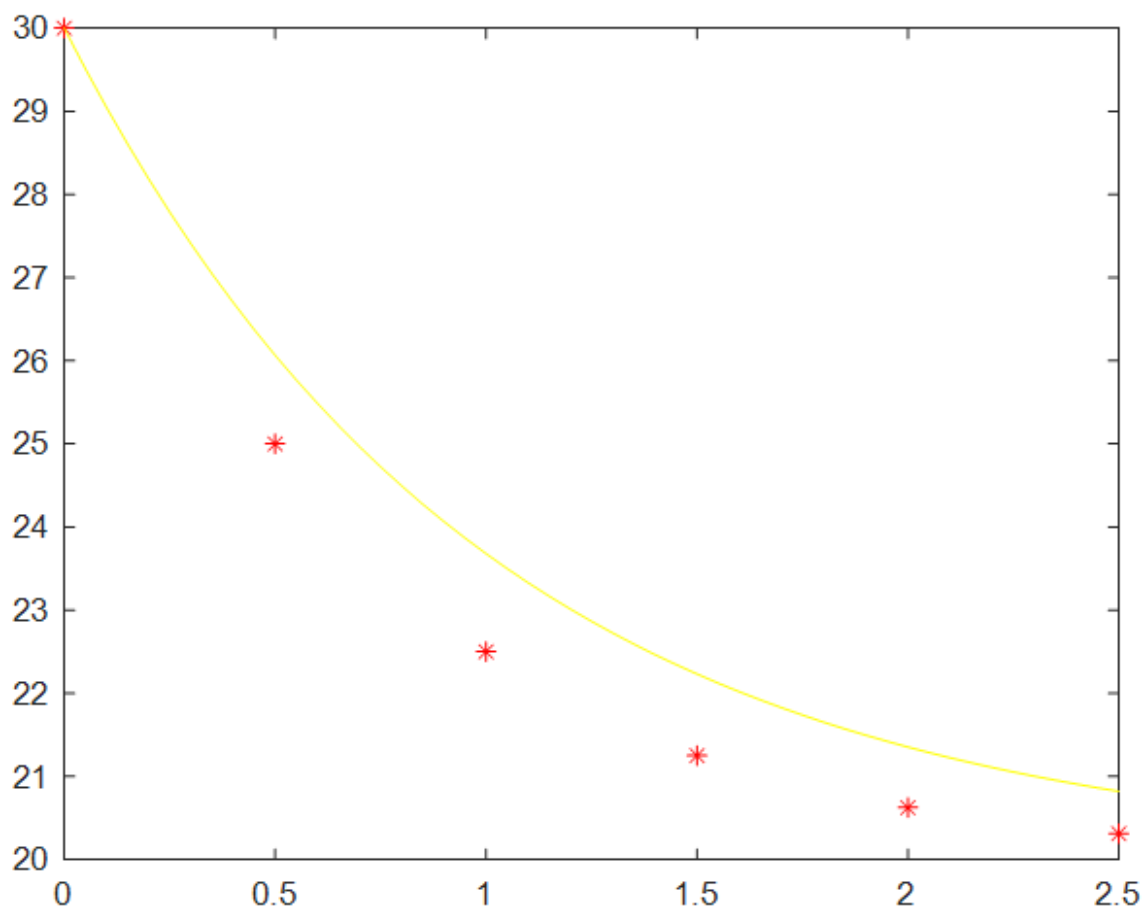
$$y_1 = 1 + \frac{0.1}{2} (f(0, 1) + f(0.1, 1 + 0.1 * f(0, 1)))$$

$$y_1 = 1 + \frac{0.1}{2} (-2 * 0 * 1) + (-2 * 0.1 * 1)) = 0.990$$

x_n	Solution exacte $y = \exp(-x^2)$	Euler	Euler améliorée
0	1	1	1
0.1	0.9900	1	0.9900
0.2	0.9608	0.98	0.9607
0.3	0.9139	0.9408	0.9138
0.4	0.8521	0.8843	0.8520
0.5	0.7788	0.8135	0.7788

Code Matlab

```
Editor - D:\teaching\licence\2\Numerical methods\2017-2018\m files\euler.m
+7 Newt.m x New Text Document.txt x symvar.m x trapez.m x simps.m x rectangle.m x rectangle_method.m x euler.m x
1 - clear,clc,close all
2 - f=@ (x,y) -y+20
3 - a=0,b=2.5;
4 - h=0.5;
5 - % x(1)=a;
6 - % y(1) = 3.0;
7 - x(1)=a;
8 - y(1) = 30;
9 - N = (b - a)/h;
10 - x = a:h:b;
11 - % f= @ (x,y) -1.2*y+7*exp(-0.3*x);
12
13 - for i = 1:N
14 -     x(i+1) = x(i) + h
15 -     y(i+1) = y(i) + f(x(i),y(i))*h
16 - end
17 - xp=0:0.05:2.5;
18 - yp=20+10*exp(-xp)
19 - plot(x,y,'*r',xp,yp,'y')
```



Méthode de Runge-kutta

La méthode de Runge-Kutta c'est l'amélioration de la méthode d'Euler à des ordres supérieurs à un (**ordre élevé**). Cette méthode donne plus de précision (elles génèrent des solutions numériques plus proches des solutions analytiques) que la méthode d'Euler.

La relation de la méthode de Runge-kutta d'ordre 2 :

$$y_{n+1} = y_n + h * f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$

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

Il existe aussi la méthode de Runge-Kutta d'ordre 4 :

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

Avec :

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

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

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

$$k_4 = h * f(x_n + h, y_n + k_3)$$

Exemple

En utilisant la méthode de Runge-kutta d'ordre 4, résoudre l'équation différentielle suivante :

$$y' = 2xy ; \quad y(0) = 1 ; \quad h=0.1$$

Solution

D'abord, on calcule les coefficients:

$$k_1 = h * f(x_0, y_0) = 0.1 * f(0, 1) = 0.1 * (2 * 0 * 1) = 0$$

$$k_2 = h * f\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right) = 0.1 * f(0.05, 1) = 0.1 * (2 * 0.05 * 1) = 0.01$$

$$k_3 = h * f\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right) = 0.1 * f(0.05, 1.005) = 0.1 * (2 * 0.05 * 1.005) = 0.01005$$

$$k_4 = h * f(x_0 + h, y_0 + k_3) = 0.1 * f(0.1, 1.01005) = 0.1 * (2 * 0.1 * 1.01005) = 0.020201$$

$$y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$y_1 = y_0 + \frac{1}{6}(0 + 0.02 + 0.02010 + 0.020201) = 1.01005$$

On continue de la même façon, on obtient :

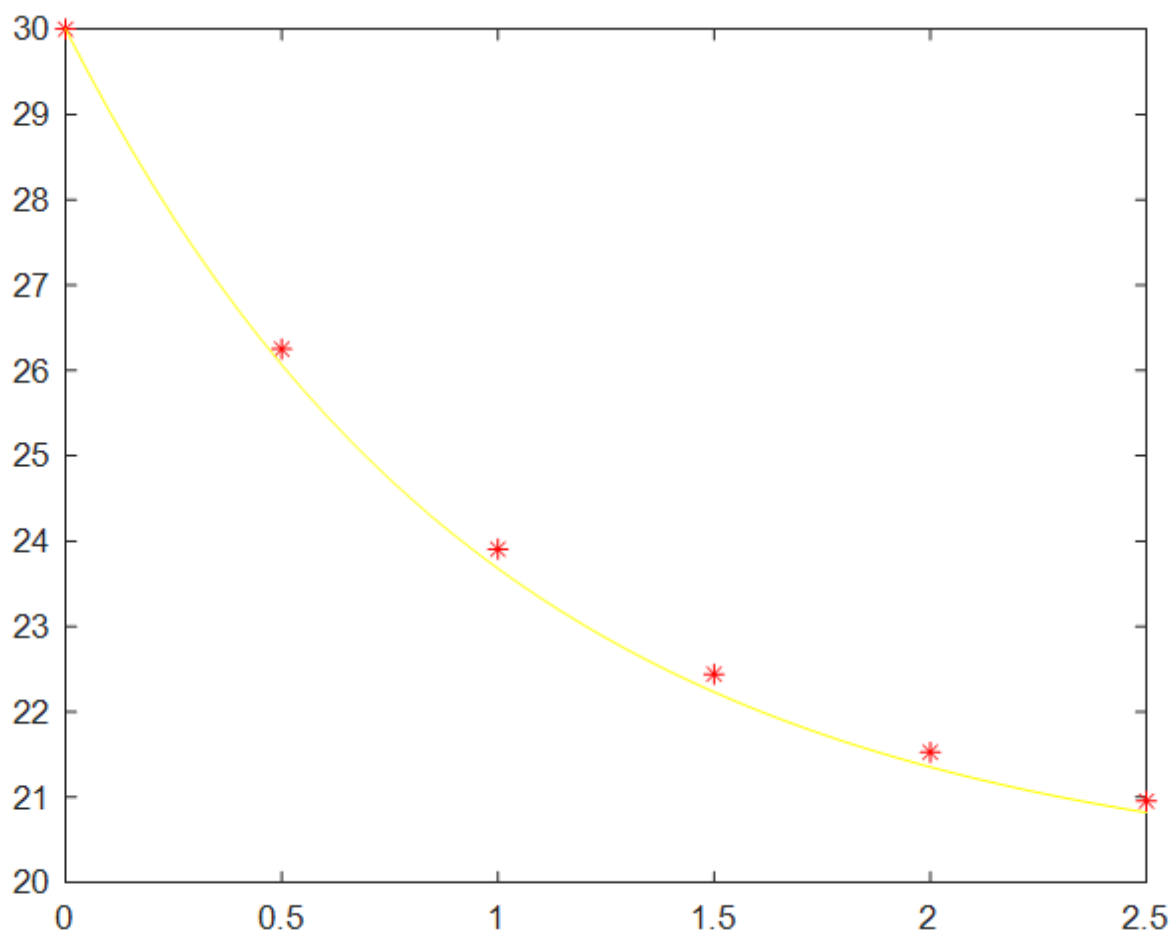
Itération	x_n	y_n
0	0	1
1	0.1	1.01005
2	0.2	1.04081
3	0.3	1.09417
4	0.4	1.17351
5	0.5	1.28403

Code Matlab

```

Editor - D:\teaching\licence\2\Numerical methods\2017-2018\m files\Range_Kutta2.m
+9 symvar.m x trapez.m x simps.m x rectangle.m x rectangle_method.m x euler.m x Range_Kutta2.m x +
1 - clear,clc,close all
2 - f=@ (x,y) -y+20
3 - % f=@ (x,y) -1.2*y+7*exp(-0.3*x);
4 - a=0,b=2.5
5 - h=0.5;
6 - x(1)=a;
7 - y(1) = 30;
8 - x = a:h:b;
9 - % x(1)=a;
10 - % y(1) = 3.0;
11 - N = (b - a)/h;
12 - for i = 1:N
13 -     k1 = f(x(i),y(i))
14 -     k2 =f((x(i)+h),y(i)+k1*h)
15 -     y(i+1) = y(i) + 1/2*(k1+k2)*h
16 - end
17 - xp=0:0.05:2.5;
18 - yp=20+10*exp(-xp);
19 - % yp=70/9*exp(-0.3*xp) -43/9*exp(-1.2*xp);
20 - plot(x,y,'r',xp,yp,'y')

```



Chapitre 05:
L'interpolation
polynomiale

Introduction

L'interpolation polynomiale permet d'interpoler une série de points par un polynôme qui passe exactement par ces points appelés aussi nœuds. On se donne $n + 1$ points $(x_0, y_0) \dots (x_n, y_n)$ avec les x_i croissants et distincts deux à deux). On se propose de construire un polynôme de degré minimal qui aux abscisses x_i prend les valeurs y_i , ce que la méthode suivante permet de réaliser.

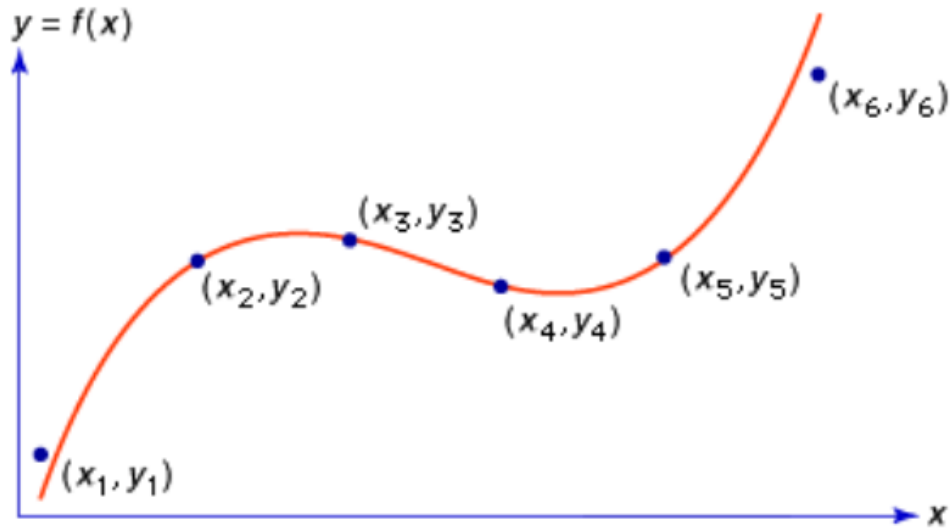


Figure 01 : Polynôme interpolé

✚ Méthode de Lagrange

La méthode d'interpolation de Lagrange est une technique numérique permettant de déterminer un polynôme qui passe exactement par un ensemble de points donnés. Le polynôme d'interpolation de Lagrange s'écrit:

$$P_n(x) = \sum_{i=0}^n y_i L_i(x)$$

Où L_i est le polynôme de base de Lagrange défini par :

$$L_i = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j}$$

Exemple

Déterminer le polynôme de Lagrange qui passe par les points suivants :

$$(1, 2), (2, 3), (4, 1)$$

Solution

Comme premier pas, on doit calculer les polynôme de base L_i :

Pour $x_0 = 1$

$$L_0 = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x - 2)(x - 4)}{(1 - 2)(1 - 4)}$$
$$L_0 = \frac{(x - 2)(x - 4)}{3}$$

Pour $x_1 = 2$

$$L_1 = \frac{(x - 1)(x - 4)}{(2 - 1)(2 - 4)}$$
$$L_1 = \frac{(x - 1)(x - 4)}{-2}$$

Pour $x_2 = 4$

$$L_2 = \frac{(x - 1)(x - 2)}{(4 - 1)(4 - 2)}$$
$$L_2 = \frac{(x - 1)(x - 2)}{6}$$

Maintenant, on passe au deuxième étape ou on construit le polynôme :

$$P(x) = 2L_0(x) + 3L_1(x) + L_2(x)$$

On obtient après développement :

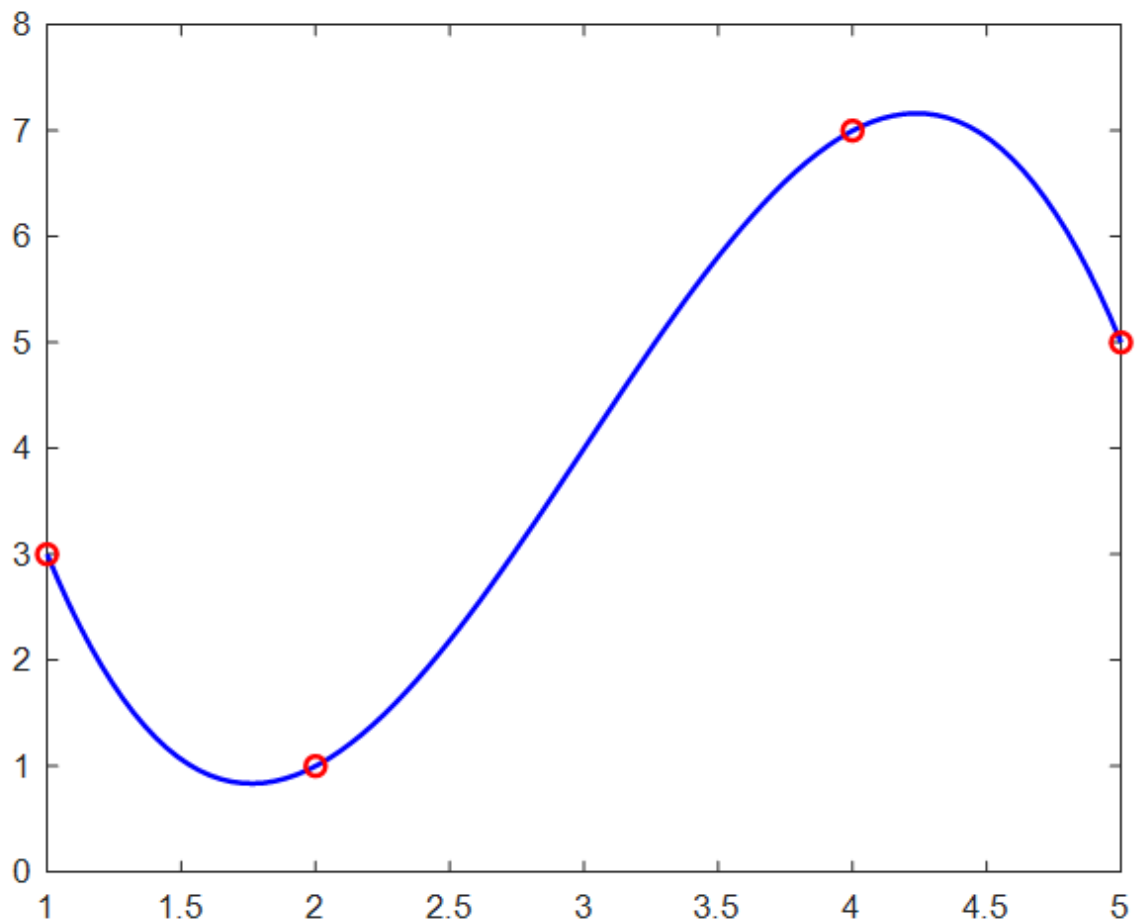
$$P(x) = -\frac{2}{3}x^2 + \frac{7}{3}x + \frac{1}{3}$$

Code Matlab

```
% ==Lagrange Interpolation ==
x = [1, 2, 4, 5]; y = [3, 1, 7, 5]; % Data points
X = linspace(1, 5, 100);           % Query points
Y = X * 0;                         % Initialize output

% The 6-line Lagrange logic
for i = 1:length(x)
    p = 1; k = x; k(i) = [];        % k is all x points except x(i)
    for j = k, p = p .* (X - j) ./ (x(i) - j); end
    Y = Y + y(i) * p;
end

% Plot to verify
plot(X, Y, 'b-', x, y, 'ro', 'LineWidth', 1.5);
```



Méthode de Newton (Méthode des différences divisées)

La méthode d'interpolation de Newton permet de déterminer un polynôme passant par un ensemble de points déterminés. Elle est particulièrement pratique lorsque de nouveaux points sont ajoutés, car il n'est pas nécessaire de recalculer tout le polynôme comme avec la méthode de Lagrange.

La formule du polynôme de Newton est comme suit :

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots \dots \dots a_n(x - x_0)(x - x_1) \dots (x - x_{n-1})$$

Exemple

Construit le polynôme de Newton du deuxième degré qui relie les points suivants :

x_i	1	2	4
$f(x_i)$	1	8	64

Solution

$$\left[\begin{array}{l} x_0 = 1 \quad f(x_0) = 1 \\ x_1 = 2 \quad f(x_1) = 8 \\ x_2 = 4 \quad f(x_2) = 64 \end{array} \quad \begin{array}{l} \frac{8 - 1}{2 - 1} = 7 \\ \frac{64 - 8}{4 - 2} = 28 \quad \frac{28 - 7}{4 - 1} = 7 \end{array} \right]$$

Un tableau récapitulatif

x_0	$f(x_i)$	1ere diff.div	2eme diff.div
1	1	7	7
2	8	28	
4	64		

Donc :

$$a_0 = 1$$

$$a_1 = 7$$

$$a_2 = 7$$

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

$$P_2(x)=1 + 7(x-1) + 7(x-1)(x-2)$$

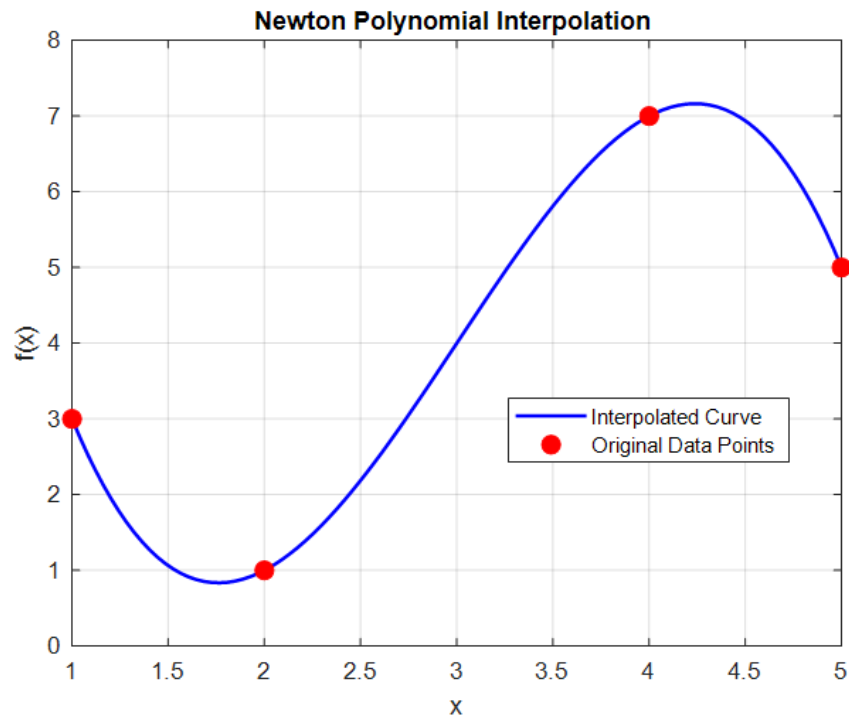
$$P_2(x)=7x^2 - 14x + 8$$

Code Matlab

```
clear all ,clc
x = [0 1 3 ];
y = [0 1 9] ;
n = length(x);

a(1) = y(1)
for i=1:n-1
    divDIF(i,1) = (y(i+1)-y(i))/(x(i+1)-x(i))
end
for j =2:n-1
    for i =1:n-j
        divDIF(i,j) = (divDIF(i+1,j-1)-divDIF(i,j-1))/(x(j+i)-x(i))
    end
end
for j =2:n
    a(j) = divDIF(1,j-1)
end

Xint= 2;
Yint =a(1);
xn = 1;
for k = 2:n
    xn = xn.*(Xint-x(k-1));
    Yint = Yint+a(k)*xn
end
plot(x,y,'o',Xint,Yint)
```



Références bibliographiques

A.Gilat,

Numerical Methods for Engineers and Scientists: An Introduction with Applications Using MATLAB. 3rd Edition, Wiley, Hoboken, New Jersey, 2014.

Chapra, S. C and Canale, R. P

Méthodes numériques pour ingénieurs (ou Numerical Methods for Engineers). McGraw-Hill, 7e édition, 2015.

Fallah, N.

Méthodes numérique (Polycopie de cours). Université des sciences et de la technologie Houari Boumediene(USTHB), Faculté de mathématiques. Alger, Algérie.

Cheney, W., & Kincaid, D

Numerical Mathematics and Computing.