People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

UNIVERSITY ECHAHIDE HAMMA LAKHDAR-EL-OUED

FACULTY OF EXACT SCIENCES

Master's Thesis

ACADEMIC MASTER

Domain : Mathematics and Computer Science
Major : Mathematics
Speciality :Fundamental and Applied Mathematics

Project

Solving Partial Differential Equations with Matlab

Presented by : Hezbri Hafsa and Adjiba Kheira
Under the supervision of: Mohammed Moumen Bekkouche

Supported before the jury:

M: Mohammed Salah Mesai Aoun MA(B)	University El-Oued	President
M:Mohmmed Moumen Bekkouche.MC(A)	University El-Oued	Rapporteur
M Nadjet Doudi . MA(A)	University El-Oued	Examiner

Academic Year : 2023 – 2024

الإهداء

﴿ وَآخِرُ دَعْوَاهُمْ أَنِ الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ ﴾

الحمد لله حبا وشكرا وامتنانا على البدء والختام

لم تكن الرحلة قصيرة ولا الطريق محفوفا بالتسهلات، لكنني فعلتها، بالحمد لله الذي يسرالبدايات وبلغنا النهايات بفضله وكرمه .

إلى النور الذي أنار دربي والسراج الذي لاينطفي نوره ابدا والذي بذل جهد السنين من أجل ان اعتلي سلالم النجاح **"قدوتي أبي الغالي "** .

إلى من اخص الله الجنه تحت قدميها وغمرتني بالحب والحنان واشعرتني بالسعاده والامان هي حياتي وكل عمري **" قرة عيني أمي الغالية ".**

إلى من ساندني بكل حب عند ضعفي وازاح عن طريقي المتاعب

أخواتي **" الزهرة، منى، نور، سارة ".**

إلى أولاد أختي و والدهم **"أميمه و محمد، عبد العالي ".**

إلى روح جدي **"عبد الكريم رحمه الله ".**

إلى جميع أقاربي **"جداتي وأجدادي، أعمامي وعماتي، أخوالي وخالاتي".**

إلى الشهداء والأسرى والمجاهدين الفلسطينيين **" إلى أهلنا في غزة ".**

إلى أصدقاء والدي **"محمد البشير الزين، كمال مصطفاوي، سليماني محمد العربي، علي رزاق ".**

إلى من ألنَّ لي الطريق وكُنَّ كما الزهر الذي يعبق سعادةً و لُطفاً

" صديقاتي: روميصاء، هنية، منى، نجوى، نصيرة، شهيرة، أسيل ،رهف، رحيل، سارة، مرام، عفاف،رجاء،خيرة " .

إلى مشرفي **" الدكتور محمد مؤمن بكوش "** على وقته وجهده وتفانيه في تأدية أمانته .

إلى من تكبدَ أعباء تعليمي من الحرف وحتى ما أنا عليه اليوم وخاصة أساتذتي في قسم الرياضيات

" إلى هؤلاء جميعا أهدي هذا الجهد ".

هزبري حفصه

الإهداء

اللهم صل وسلم على نبينا محمد

لكل بداية نهاية ولكل رحلة طريق.الحمد لله على ما سلكناه وما عشناه وما مررنا به هي رحلة تعليمية بدايتها ابندائية ونهايتها جامعية وختامها مسك .

اهدي تخرجي اللا من ساندني في دراستي وحياتي وتمنى أن يراني ويكون معي في هذه اللحظات

"جدي رحمه الله " .

الى من أحمل اسمه بكل افتخار،رفيق دربي وحبيب قلبي

" والدي ".

الى ملاكي في الحياة من كان دعائها سر نجاحي و وجودها سبب استمراري

" والدتي ".

الى عضدي وسندي في الحياة.

"اخوتي واخواتي ".

الى رفيقات الدرب وأختاي **"نور الاسلام و أنوار ".**

الى كل من لهم مكانة في قلبي ومن كانوا معي داعمين وساندين في السراء والضراء دمتم لي

"ريحانة، هنادي، جمانة، سجود، محمد غميمة ".

الى كل أحباب قلبي من أعمام وأخوال حفظكم الله

إلى الدكتور المشرف **" محمد مؤمن بكوش " .**

إلى من تشاركت معها هذا العمل **" حفصه هزبري "**

عجيبه خيرة

شكر و تقدير

﴿الْحَمْدُ لِلّهِ الَّذِي هَدَانَا لِهَـذَا وَمَا كُنَّا لِنَهْتَدِيَ لَوْلا أَنْ هَدَانَا اللّهُ﴾

وصل اللهم وبارك على النعمة المسداة والرحمة المهداة والسراج المنير، حبيبنا وشفيعنا محمد بن عبد الله .

لا يسعنا ونحن ننهي هذا الجهد العلمي الا أن نتقدم بفائق الشكر والامتنان إلى من مد لنا العون وساعدنا في إنجاز هذا البحث،ونخص بالذكر المشرف الدكتور **محمد مؤمن بكوش** الذي أشرف بعناية فائقه على البحث،ولما بذله بإخلاص من صبروجهد، وما قدمه من توجيهات سديدة لإخراج البحث بالمستوى المطلوب.

جزاه الله عنا خير الجزاء وحفظه من كل مكروه.

كما نتقدم بأسمى عبارات الشكر والامتنان والتقدير والمحبة الى اللذين حملوا أقدس رسالة في الحياة الى اللذين مهدوا لنا طريق العلم والمعرفة الى جميع أساتذتنا الأفاضل، اساتذة قسم الرياضيات بجامعة الشهيد حمه لخضروعلى راسهم الدكتور**غنديرعون عبد اللطيف** والدكتورة **دودي نجاة** .

والشكر الخاص الى عميد الكلية الأستاذ الدكتور **منصور عبد الوهاب** و رئيس القسم الدكتور**سعيد بيلول**

Contents

List of Figures

List of Tables

Notations

We will use the following notations throughout the work:

$\mathbb{R}$	Set of real numbers.
$\mathbb{R}_+$	Set of positive real numbers.
$\mathbb{R}_+^*$	Set of non- zero positive real numbers.
$\mathbb{N}$	Set of natural numbers.
(a, b)	That is to say the domain $[a, b]$ or $]a, b]$ or $[a, b[$ or $]a, b[$.
$L^2([a, b])$	The space of lebesgue-measurable functions with 2 -nd power integrable over 2.
$H^1([a, b])$	The Sobolev space of order 1 of 2.
V	The Hilbert space equipped with the scalar product.
$\|\|.\|\|_V$	The norm of V.
$\frac{\partial u}{\partial x}$	Derivative of u by x.
$\frac{\partial^2 u}{\partial t}$	The second derivative of u by t.
Δu	Laplacian of u.
$< u, v >$	The scalar product of vectors u and v.
FD	Finite Difference.
FE	Finite Element.
FV	Finite Volume.

Introduction

Partial Differential Equations **PDEs** are a fundamental component of modern mathematics and science. In any system wherein the independent variables (e.g, space and time) take on a continuum of values, the laws governing the system usually result in a partial differential equation **PDE** for the quantity or quantities of interest.

Since models used to approximate many physical, biological, and economic systems are **PDE**-based, having some proficiency in **PDEs** can provide a means of addressing these complex systems.

Solving **PDEs** is a crucial task in many scientific and engineering fields, such as physics, fluid dynamics, heat transfer, and structural analysis. **MATLAB**, a popular numerical computing environment, provides powerful tools and libraries for solving **PDEs** efficiently.

One of the main reasons for choosing this topic is the lack of references that have addressed applied scientific issues about solving partial differential equations in **MATLAB**.

This research therefore aims to study the concepts of partial differential equations and familiarize itself with the concept of mathematical modelling.

The sources and references of this research are distinguished by their diversity, most notably the books: Li and Yi-Tung Chen, Computational Partial Differential Equations Using MATLAB, Seongjai Kim, Numerical Methods for Partial Differential Equations. Which contributed greatly to the excellence of the subject. In dealing with the topic, we followed an introduction, three chapters and a conclusion plan. In Chapter 1, we devoted this part to talking about Matlab's importance and how it is used, focused on defining and explaining his tools that we used in our research.

In Chapter 2, we discussed the definition, classification of partial differential equations and also studied ways of solving them, where we studied the method of FDM, FEM and FVM.

and Chapter 3 was to program issues with MATLAB and offer solutions.

This study culminated in a culmination of our findings.

At last, we wish that we would have succeeded even a small amount in our research.

Chapter 1

Introduction of MATLAB and Basic Operations

As the first chapter of the memoir, we give a brief introduction to **MATLAB** and a few basic operations in **MATLAB**.

1.1 Definition of MATLAB

MATLAB is a shortcut for wholesale **Matrix Laboratory**. Founded year 1984 from Cliff Muller and Jack Little, and is the language of the sulto code embraced by the American company Math Works. Used in solving scientific, technological and industrial liquid solution. Where we can by MATLAB a Mathematical calculations and writing programs and Drawing charts and Modeling.

1.2 Matlab Components

The calculated programs consist of five main parts and are:

- Programming language: It is a prepared Programming language and composed of sub files in which matrices, determinants and fever.

- Labor: It is a rang of means and facilities used to enable the user from work. This ocean contains means of organizing and managing variables as you bring and send information.

- Graphical organizer: It is a sedamal drawing system that contains orders to draw the two dimensions and three dimensions. It also contains orders to show and move them.

- Matlab sports functions library: Contains the following functions: Transforms fausse Fourier, Matrix eigenvalue, Matrice inverse, Cosine, Sine, Somme.

- It is a means of helping to connect the programs prepared in other languages such as (C and FOTRAN) with application program interface MATLAB.

Notation 1.1. *Some control functions in my window* ***command window, work space*** *and we are summarized in the following table:*

function	**operation**
clc	clear all contents command window just without scanning it from the work space
clear	clear all contents work space included all variables that were used in the program
clear a b c	Clear variables a b c from work space
clear a*	clear all variables in work space which begins with a letter a
who	View contents work space included all variables that were used in the program
whos	View contents work space in detail (name, dimension, size, type)
save	Save all contents work space in the default file MATLAB.mat

Table 1.1: Some instructions of command window and work space.

1.3 Program interface

The program interface is as shown in the picture

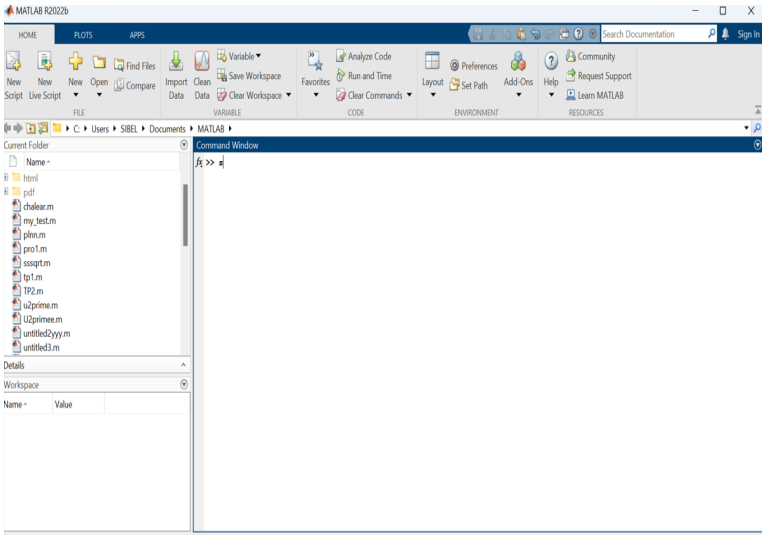


Figure 1.1: The default MATLAB Desktop.

The most important elements of the MATLAB screen are the following:

- **The Command Window**: This runs MATLAB functions.
- **The Command History**: This presents a history of the functions introduced in the Command Window and allows you to copy and execute them.
- **The Launch Pad**: This runs tools and gives you access to documentation for all MathWorks products currently installed on your computer.
- **The Current Directory**: This shows MATLAB files and execute files (such as opening and search for content operations).
- **Help (support)**: This allows you to search and read the documentation for the complete family of MATLAB products.
- **The Workspace**: This shows the present contents of the workspace and allows you to make changes to it.
- **The Array Editor**: This displays the contents of arrays in a tabular format and allows you to edit their values.
- **The Editor/Debugger**: This allows you to create, edit, and check M-files (files that contain MATLAB functions).

1.3.1 Command Window,Work Space,Command History,Help and The Editor for M-files in Matlab

Command window:

The Command Window (Figure 1-2) is the main way to communicate with MATLAB. It appears on the desktop when MATLAB starts and is used to execute all operations and functions. The entries are written to the right of the prompt >> and, once completed, they run after pressing Enter. It appears on the next screen.

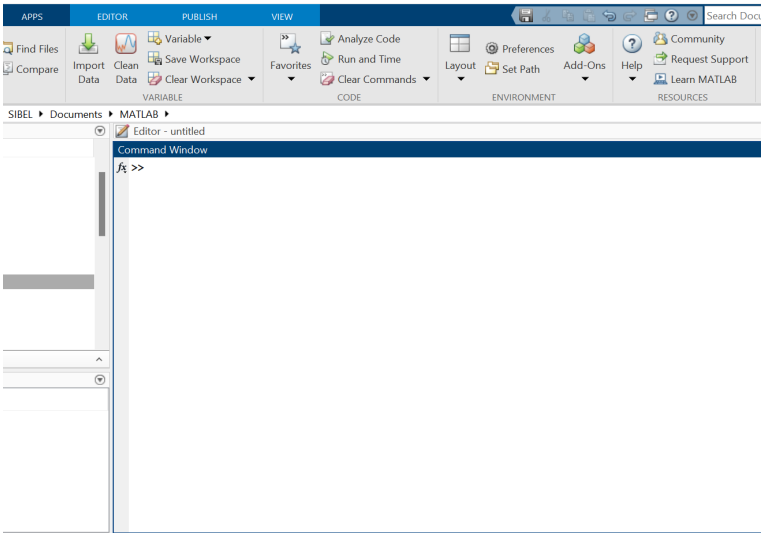


Figure 1.2: Command window

Work Space:

The Workspace window is located in the top left corner of the MATLAB desktop and is obtained by clicking on the label Work Space under it (Figure 1-3). Its function is to display the variables stored in memory. It shows the name, type, size and class of each variable, the following (Figure 1-3) shows a Work Space screen.

Figure 1.3: Work Space

Command History:

The Command History window (Figure 1-4) appears when you start MATLAB. It is located at the bottom right of the MATLAB desktop. The Everything is being recorded user on a MATLAB program in this window.

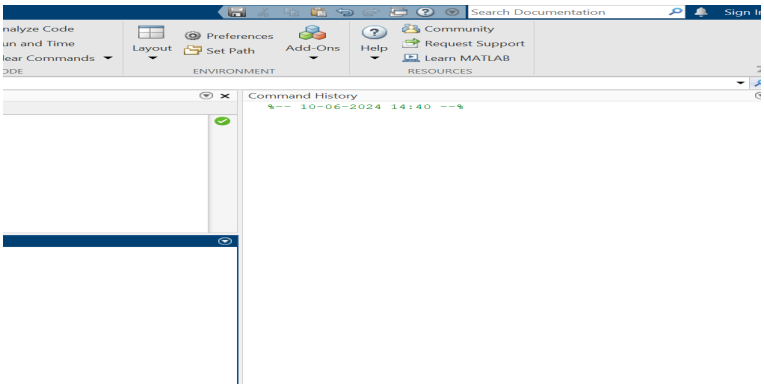


Figure 1.4: Command History

Help in Matlab:

MATLAB has a fairly efficient inline help system. The first tool to consider is browser support (Figure 1-5), which is accessed via the icon? or by typing help browser in the Command Window (the Help Browser option must be selected in the View menu). Selecting a theme in the pane on the left of the help browser will present help on the selected topic in the right pane, and you can navigate through the content via hyperlinks. The top bar of the left navigation pane features the options Content (support for content), Index (help by alphabetical index), Search (find help by subject) and Favorites (favorite help topics).

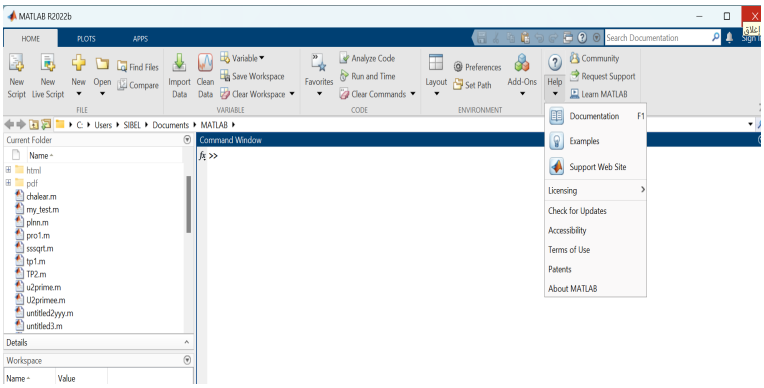


Figure 1.5: Help in matlab

The Editor for M-files

To create a new M-file in the Editor/Debugger simply click the button + in the MATLAB Tools toolbar or select File > New > M-file in the MATLAB desktop (Figure 1-6). The Editor/Debugger opens a file in which you create an M-file, i.e. a blank file for MATLAB programming code . The Edit command in the Command Window also opens the Editor/Debugger. To open an existing M-file use File Open in the MATLAB desktop. You can also use the command Open in the Command Window.

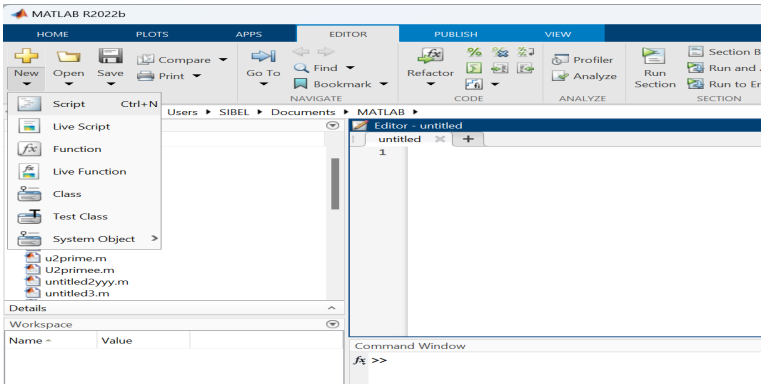


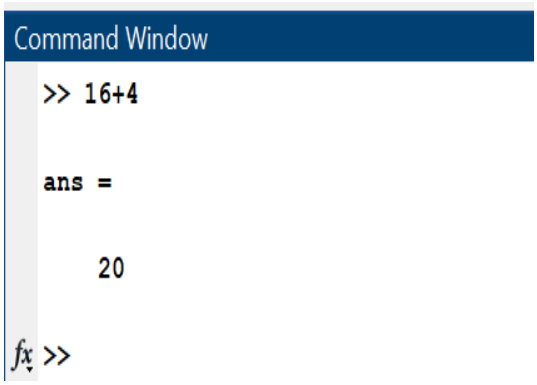
Figure 1.6: The Editor for M-files

1.4 Some processes and basic orders in the MATLAB

1.4.1 Some basic processes in the MATLAB

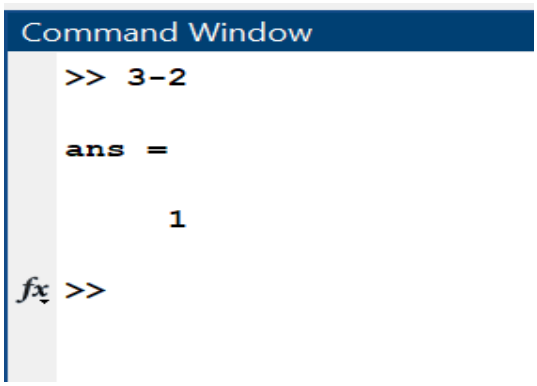
Adition:

Take combination mark in the muffler known as "+", as we collect $16 + 4$ the **MATLAB** will put the answer in the form of numbers is 20.



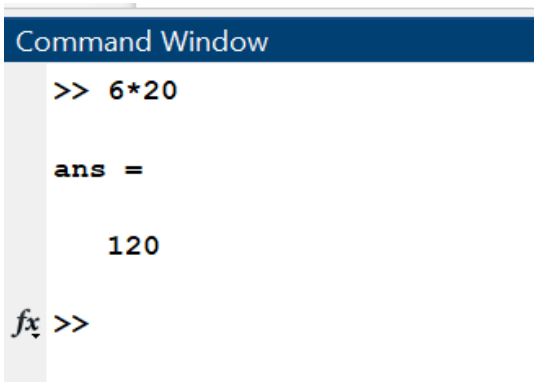
Subtraction operation:

Take the subtraction process $(-)$ in the **MATLAB**, for example: $3 - 2 = 1$.



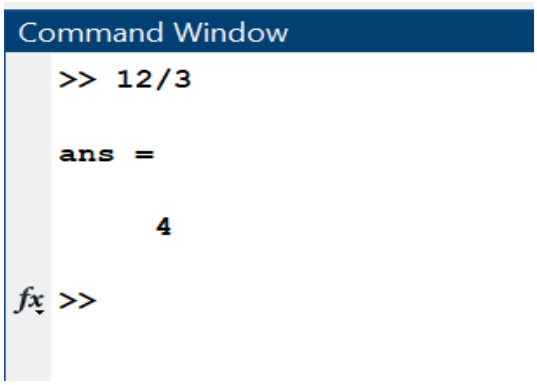
Multiplication:

Take a beating process $(*)$, for example: $20 * 6 = 120$.



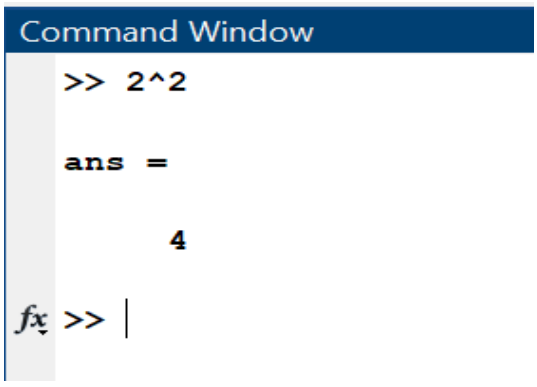
Division:

Take a division process (/), for example 12 on 3 equal 4.



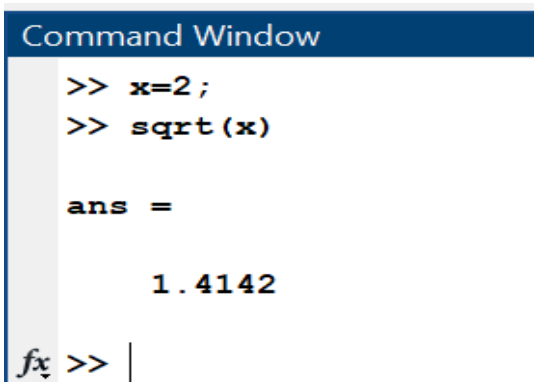
Cutting:

The icon ^ takes, this icon can be obtained by pressing **shift** +6 in the keyboard, for example: $2^2 = 4$.



The square root:

The square root is taken for any number by writing the command "sqrt".



1.4.2 Some basic orders in the MATLAB

Trigonometric Functions:

Trigonometric function	Built in function
Sine	sin
Cosine	cos
Tangent	tan
Secant	sec
Cosecant	csc
Cotangent	cot

Table 1.2: Trigonometric Functions.

Reverse triangular functions:

Reverse Trigonometric function	Built in function
Reverse Sine	asin
Reverse Cosine	acos
Reverse Tangent	atan
Reverse Secant	asec
Reverse Cosecant	acsc
Reverse Cotangent	acot

Table 1.3: Reverse triangular functions.

Hyperbolic function	Built in function
Hyperbolic Sine	sinh
Hyperbolic Cosine	cosh
Hyperbolic Tangent	tanh
Hyperbolic Secant	sech
Hyperbolic Cosecant	csch
Hyperbolic Cotangent	coth

Table 1.4: Hyperbolic Functions.

Hyperbolic Functions:

Reverse Hyperbolic Functions:

Reverse Hyperbolic function	Built in function
Reverse Hyperbolic Sine	asinh
Reverse Hyperbolic Cosine	acosh
Reverse Hyperbolic Tangent	atanh
Reverse Hyperbolic Secant	asec
Reverse Hyperbolic Cosecant	acsc
Reverse Hyperbolic Cotangent	acot

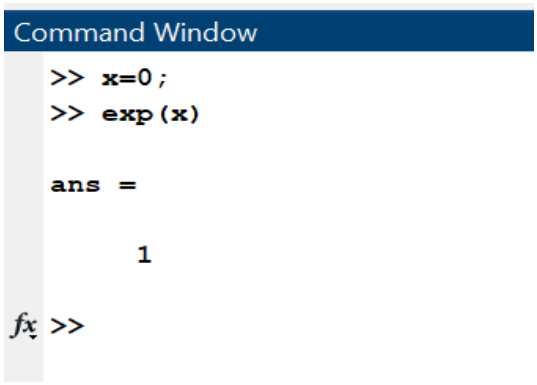
Table 1.5: Reverse Hyperbolic Functions.

The Exponential Functions:

The sore function takes the following sports formula

$$y = e^x.$$

In the MATLAB, they are short exp(x).



Composite Functions:

The complex numbers take a single formula and are the presence of a part of the real number and part of the imaginary number, and be in the following formula.

$$z = x + y * i.$$

The Matlab program is made in many processes that are made in the complex numbers, such as the real number selection only select the imaginary number.

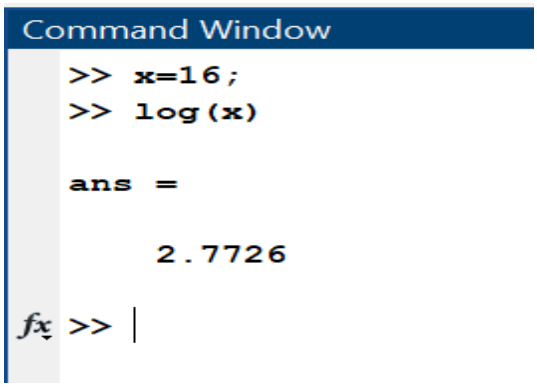
Find corner : it is obtained through the following relationship. $angel = \tan^{-1}(\frac{Imaginary \quad number}{Real \quad number})$

Collect tow compounds: this is done by collecting real numbers with each other, and collecting the complex numbers with some **Absolute value:** it is obtained through the following relationship.

Absolute value= $\sqrt{X^2 + Y^2}$.

Natural logarithms:

The muffler symbolizes natural o gars $\log(x)$.



Approximate operations:

Any decimal number that is a reality between two numbers, and the quantity has the ability to choose one of these two digit using the two thing **ceil** to choose the biggest number, and it **floor** to choose the smaller number.

Put the addresses during programming:

We also return to programs **Qbasic** and **C++** and many more programming programs, addresses are placed for what we do where to be like to know what we are doing in a part of the program.
in the calculation program to put an address, we must begin to mark a percentage, and then write what we want after.

1.4.3 Define some variables

Some defined variables in the MATLAB and know program:

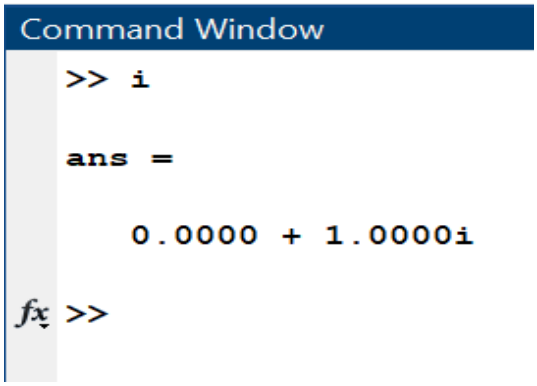
Stands for	**Predefined Variable**
$\pi = 3.1416$	pi
$\infty = infinity$	Inf
Not a Number	NaN
The complex variable $\sqrt{-1}$	i
The complex variable $\sqrt{-1}$	j

Table 1.6: Some defined variables in the MATLAB and know program.

We overwrite the value of the composite number:

We learned that if we write (i) in the command window appears:

$$0.0000 + 1.0000i$$



Canceling values and results:

The quantity can scan the entries and results (which are recorded in the results recording window), without scanning what you wrote, using it **Clear**.

The sanctification process of the venues:

Not a program condition, it is possible to work for only one variable, we do a total survey of the tissue by writing order **clear** then the name of variable.

1.5 Matrices

1.5.1 What is Matrice ?

It is a set of data that is placed in the classes and columns, and take the following shape.

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

The matrices are used in the polynomials and in a set of equations.

1.5.2 How to write matrices in the MATLAB Program?

The matrix is entered by typing the first row items, then second and so on.
For example, write a matrix like the following.

$$\begin{pmatrix} 1 & 3 \\ 6 & 4 \end{pmatrix}$$

Example 1.1.

```
>> A=[1,3  ;  6,4]
A =
        1   3
        6   4
```

But before entering the following values, everyone should know that first grade elements are written and the first grade numbers are separated either by comma (,), or a space between the numbers, after entering the first row values, the first row items are separated from the second row items (which will enter values) either by pressing a key enter or using the separator (;).

1.5.3 Accounts in matrices

Addition:

The combination is collected with collection of the first element for the first line, for example, in the first matrix and its consideration is in the second matrix.
and condition for both have the same number of columns and lines.

Example 1.2.

```

>> A=[1  2  4;1  3  5];
>>  B=[2  1  3;6  8  2];
>>  A+B

ans =

      3      3      7
      7     11      7
```

Subtraction operation:

Where the process is made like the combination process and also has has the same requirement.

Example 1.3.

```
>> A=[1 2 4;1 3 5];
>>  B=[2 1 3;6 8 2];
>>  A-B

ans =

    -1     1     1
    -5    -5     3
```

Multiplication:

The requirement hit any matrices is to have the number of first matrix columns equal to the number of second matrix lines.

Example 1.4.

```
>> A=[1 2 4;1 3 5];
>>  B=[2 1 3;6 8 2];
>> A.*B

ans =

     2     2    12
     6    24    10
```

Special orders for matrices:

In this figure we applied some examples of orders in the table you follow.

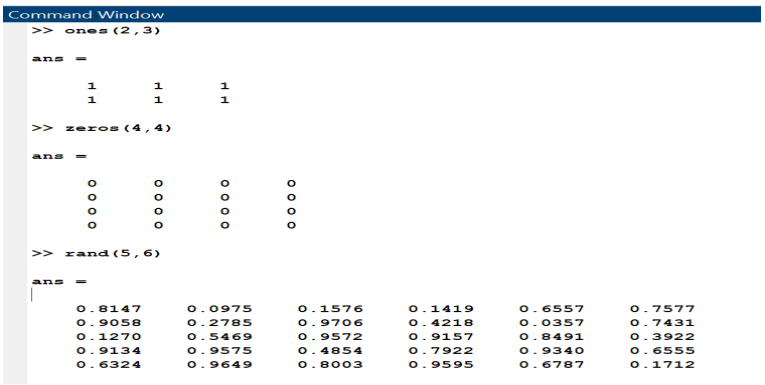


Figure 1.7: An illustrative example of some commands specific to an array.

size	Returns the number of rows and columns of A.
A'	Transported of matrix A.
Inv(A) or A^-1	Invert matrix A.
det(A)	Determinant matrix A.
diag(A)	Diagonal matrix A.
zeros(a,b)	Creates an $a * b$ matrix with all elements equal to 0.
ones(a,b)	Creates an $a * b$ matrix with all elements equal to 1.
rand(a,b)	Creates an $a * b$ random matrix (normal distribution).
eye(a,b)	Creates an $a * b$ identity matrix.
trace(A)	He finds the collection of country elements.

Table 1.7: Special orders for matrices.

1.6 Vectors

1.6.1 Operations on vectors

There are two forms of vector are horizontal and vertical.

Example 1.5.

```
>> A[1  2  3  4  5]
A =
 1  2  3  4  5
>>B[1;2;3;4;5]
B =
 1
 2
 3
 4
 5
```

length	Find the number of vector items.
max	Determines the biggest element in the vector.
min	Determines the smaller element in the vector.
prod	Calculates the efficiency of the vector elements.
sum	The total elements of the vector.

Table 1.8: Special orders for vectors.

1.6.2 Special orders for vectors

1.7 Script M-files

1.7.1 Defined

It is a kind of Script files which is used by the MATLAB program as a means of entering commands and code, where program commands are edited in a Script file (this file is called "M-files").

1.7.2 Create a new M-file:

There is third way to Create a new file to write the Matlab program:

Method 1: From the file menu, choose the New command where a subfolder appears, select them either the script or M-file.

Method 2: Press the New M-file command icon that has a white sheet shape and exist in the toolbar.

Method 3: Write the edit command inside the command window as follows:
>> edit.

- The previous three methods will appear a new window.

The MATLAB program will be determined by default name for this file is untitled and when you save this file, the MATLAB program is added to add extension (*.m) to this file name.

1.7.3 Conditions for saving M-file:

1- The file name must begin with a letter and not a number or symbol. For example,the file name cannot be written in the format **1test.m** alternatively, we can name the file in the format. **test1.m**.

2- The file name must not be named after a known command or function built within a program MATLAB, for example, can not name the file with the word if because this name represents one of the functions internal program.

3- The file name must not contain some special characters, such as: $(*,\ /!;)$ except for the under score sign.

1.7.4 Save M-file

Save the MATLAB program file written on an revised editor software page by following the following method:
Go to the file menu, choose the save command, or press the keys **Ctrl+S** from the keys board, or choose the **save As** command also from the file menu to save another copy of the file, or choose the **save All** command under the file menu to **save All** currently open files.

1.7.5 Run the program

The MATLAB program is run inside M-File followed the following method:
Press **the RUN** play button icon in the toolbar in a revised window of Editor.

1.7.6 Open M-file has already been saved

Click on the open file icon in the toolbar on the desktop program or from a revised window program Editor.

1.8 Control flow

In the following section the Control flow of loops (for, while - end) and conditional statements (if - else, switch - case) will be introduced.

1.8.1 For loops

The most common way of repeating a sequence of statements for a specific number of times is to use a **for loop**. Consider the following example:

```
>>for  x=1:2
disp('this statement is repeated')
end
```

the statement between **for** and **end** will be repeated two times. The ouput on the screen will be:
this statement is repeated.
this statement is repeated.

1.8.2 While loops

Often we wish to repeat a sequence of statements while a given condition remains true. And public formula is:

```
while condition
statement(s)
end
```

Example 1.6. *The following program collects numbers from 1 to 10 and an increase of 0.5 (total numbers 1,1.5,2,2.5,....,10) using while solution:*
>> file_name
s=104.500.

Stop words (Break)

This phrase stops the implementation of For or While loop when placed within the ring.

1.8.3 If - Else - End

To execute different sequences of statements according to some conditions we can use **if-else-end**. The general syntax is as follows:

```
if expression1
    (commands evaluated if expression1 is true )
elseif expression2
    (commands evaluated if expression2 is true )
elseif expression3
    (commands evaluated if expression2 is true )
.
.
.
else
    (commands evaluated if no other expression is true )
end
```

1.8.4 Switch -Case - Otherwise - End

Another method of executing statements according to some conditions is to use **Switch - case - otherwise - end** The general syntax is as follows:

```
switch expression
case test-expression1
(commands1)
case test-expression2
(commands2)
otherwise
(commands3)
end
```

1.9 Input and output sentences

1.9.1 Input sentences

Input instruction :

Example 1.7.

```
n=input('enter n:')
m=input('enter m:')
for i=1:n
    for j=1:m
        result(i,j)=i j
    end
end
```

Formula to inser (Shaped dialog box):

```
prompt=('enter x:')
def=('20')
dlgtitle='Input for my program'
lineNo=1
answer=inputdl(prompt,dlgtitle,lineNo,def)
x=str2num(answer{1})
```

1.9.2 Pending output

1-**Disp instruction:**

Example 1.8.

```
>>sum=9.8
>>disp(['sum=',num2str(sum)])
sum=9.8
```

2-**Msgbox instruction:**

```
>>msgbox('ok','result')
```

1.10 MATLAB Plotting and Graphs

Two-dimensional Plotting

2 D Plotting Commands

Plot symbols and colors may be obtained.

>>plot(X,Y,S)

Which plot variable X against variable Y with S format.

S is a character string made 'color marker style'.

Colors and Markers:

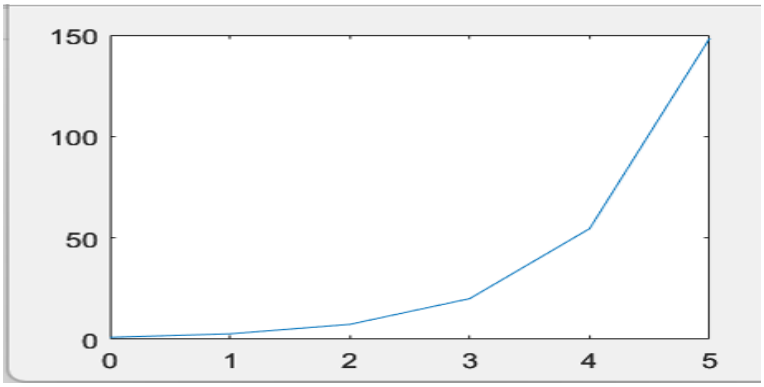
Symbol	Color
b	Blue
g	Green
r	Red
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

Table 1.9: Colors and Markers.

Example 1.9.

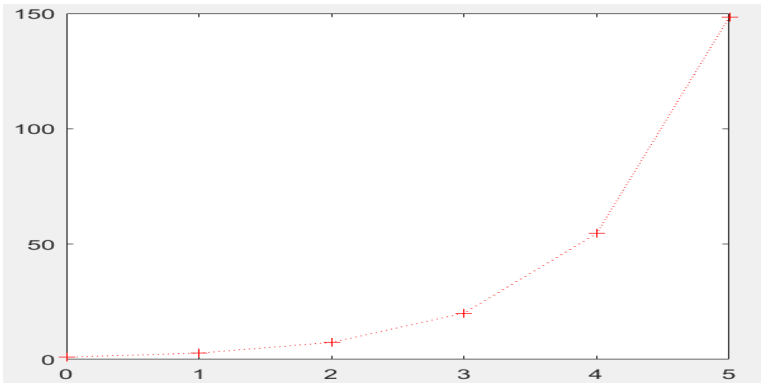
```
>>X=0:1:5;
>> Y=exp(X)

Y =

    1.0000    2.7183    7.3891   20.0855   54.5982  148.4132
```

>>plot(X,Y)



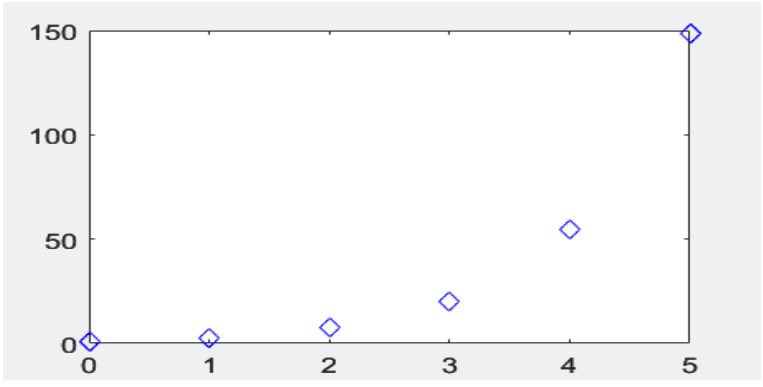
Default plots a continuous blue line

>>***plot(X, Y, 'r+:')***



plots a red dotted line with plus at each data point.

>>***plot(X, Y, 'bd')***



plots blue diamond at each data point but not draw any line.

Adding New Curves:

Using the hold command to add lines to an existing plot.

Command and Description:

hold on : Retain existing axes, add new curves to current axes when new plot commands are issued.

hold off : Releases the current figure window for new plots.

i hold : Logical command that returns 1 (true) if hold is on and 0 (false) if hold is off.

Grids, Axes box, and Labels:

There are several commands to control the appearance of the plot.

Command	Description
grid	Toggles grid status (off to on, or on to off).
title ('text')	Labels top of plot with text in quotes.
text(x,y,'text')	Adds text in quotes to location(x,y)on the current axes in units from the current plot.
Subplot	Figure command allows the creation of multiple.

Table 1.10: commands to control the appearance of the plot.

Chapter 2

Partial Differential Equations

2.1 Introduction of PDEs

Partial Differential Equation **(PDEs)** is similar to an Ordinary Differential Equation **(ODEs)**, except that the dependent variable is a function of not just one, but of several independent variables. Let's be more precise. Given a function $u = u(x_1, x_2, ..., x_n)$, a partial differential equation. **(PDEs)** in u is an equation which relates any of the partial derivatives. of u to each other and/or to any of the variables $x_1, x_2, ..., x_n$ and u. Before doing some examples, we introduce a bit of notation: Instead of the somewhat unwieldy $\frac{\partial u}{\partial x}$, $\frac{\partial^3 u}{\partial^2 x \partial y}$ and the like, we will use subscripts whenever possible. We write

$$u_x = \frac{\partial u}{\partial x}.$$

For higher order derivatives, we read the subscripts from left to right. So, for example:

$$u_{xy} = \frac{\partial}{\partial y}(\frac{\partial u}{\partial x}) = \frac{\partial^2 u}{\partial x \partial y}.$$

However, for all practical purposes, the order of differentiation will not matter to us.
So, for example, we'll have $u_{xzyx} = u_{zxxy} = u_{yxzx}$, etc.

Example 1.

$$\textbf{\textit{transport equation :}} \quad u_x + u_y = 0 \tag{2.1}$$

$$\textbf{\textit{inviscid Burger's equation :}} \quad u_t + uu_x = 0 \tag{2.2}$$

$$\textbf{\textit{Laplace's equation :}} \quad u_{xx} + u_{yy} = 0 \tag{2.3}$$

$$\textbf{\textit{wave equation:}} \quad u_{tt} - u_{xx} = 0 \tag{2.4}$$

$$\textbf{\textit{heat equation:}} \quad u_t - u_{xx} = 0 \tag{2.5}$$

$$\textbf{\textit{KdV equation :}} \quad u_t + uu_x + u_{xxx} = 0 \tag{2.6}$$

$$\textbf{\textit{Shr¨odinger's equation:}} \quad iu_t - u_{xx} = 0 \tag{2.7}$$

$$au_{xx} + 2bu_{xy} + cu_{yy} + u_x + eu_y + fu = g \quad \textit{or a,b,c,e,f and g are a function of x and y} \tag{2.8}$$

Frequently, we will seek functions which are solutions of a given **PDEs** in some restricted region. Also, in order to ensure that there is never a problem with the order of differentiation, we will require any solution u of an $nth - order$ **PDEs** to have the property that all of thenth partial derivatives of u exist and are continuous.

2.2 Classification of Partial Differential Equations

There are a number of properties by which **PDEs** can be separated into families of similar equations. The two main properties are order and linearity.

Order:

The order of a partial differential equation is the order of the highest derivative entering the equation. In examples above (2.2) are of first order; (2.3), (2.4), (2.5), (2.7) and (2,8) are of second order; (2.6) is of third order.

Linearity:

Linearity means that all instances of the unknown and its derivatives enter the equation linearly. To define this property, rewrite the equation as

$$Lu = 0. \tag{2.9}$$

Where L is an operator, which assigns u a new function Lu. For example $L = \frac{\partial^2}{\partial x^2} + 1$, then $Lu = u_{xx} + u$. The operator L is called linear if

$$L(u+v) = Lu + Lv, and\, L(cu) = cLu. \tag{2.10}$$

for any functions u, v and constant c. The equation (2.9) is called linear, if L is a linear operator. In our examples above (2.1), (2.3), (2.4), (2.5), (2.7) are linear, while (2.2) and (2.6) are nonlinear (i.e. not linear). To see this, let us check, e.g. (2.5) for linearity:

$$L(u+v) = (u+v)_t - (u+v)_{xx} = u_t + v_t - u_{xx} - v_{xx} = (u_t - u_{xx}) + (v_t - v_{xx}) = Lu + Lv.$$

And

$$L(cu) = (cu)_t - (cu)_{xx} = cu_t - cu_{xx} = c(u_t - u_{xx}) = cLu.$$

So, indeed, (2.5) is a linear equation, since it is given by a linear operator. To understand how linearity can fail, let us see what goes wrong for equation (2.2):

$$L(u+v) = (u+v)_t + (u+v)(u+v)_x = u_t + v_t + (u+v)(u_x + v_x) = (u_t + uu_x) + (v_t + vv_x) + uv_x + vu_x \neq Lu + Lv.$$

You can check that the second condition of linearity fails as well. This happens precisely due to the non linearity of the uu_x term, which is quadratic in "u and its derivatives. Notice that for a linear equation, if u is a solution, then so is cu, and if v is another solution, then $u + v$ is also a solution. In general any linear combination of solutions.

$$c_1u_1(x,y) + c_2u_2(x,y) + \cdots + c_nu_n(x,y) = \sum_{i=1}^{n} c_iu_i(x,y).$$

Will also solve the equation. The linear equation (2.9) is called homogeneous linear **PDEs**, while the equation .

$$Lu = g(x,y). \tag{2.11}$$

is called in-homogeneous linear equation. Notice that if u^h is a solution to the homogeneous equation (2.9), and u^p is a particular solution to the in-homogeneous equation (2.11), then $u^h + u^p$ is also a solution to the in-homogeneous equation (2.11). Indeed

$$L(u^h + u^p) = Lu^h + Lu^p = 0 + g = g.$$

Thus, in order to find the general solution of the in-homogeneous equation (2.11), it is enough to find the general solution of the homogeneous equation (2.9), and add to this a particular solution of the in-homogeneous equation (check that the difference of any two solutions of the in-homogeneous equation is a solution of the homogeneous equation). In this sense, there is a similarity between **ODEs** and **PDEs**, since this principle relies only on the linearity of the operator L.

2.2.1 First Order Partial Differential Equations

The first-order partial differential equation x, y, u.
$\mathrm{F}(x, y, u, \dfrac{\partial u}{\partial x}, \dfrac{\partial u}{\partial y}) = f(x, y)$.
if u and the variables x and y are unknown, then the equation is: $\mathrm{P}\dfrac{\partial u}{\partial x} + \mathrm{Q}\ \dfrac{\partial u}{\partial y} = R$.
It is a partial differential equation of the first order, where R, Q qnd P are functions of the variables x, y, u.

Example 2. *1- (IVP) of First Order PDEs:*

$$\begin{cases} au_t + bu_x = 0, \\ u(x, 0) = f(x). \end{cases}$$

2.2.2 Second-Order Partial Differential Equations

There is a classification of second order linear partial differential equations. Consider for example a **PDEs** written in the from (2,8). Where a, b and c may be functions of x and y. Based on the local value of the coefficients the equations are classified as follows:

1 - If $b^2 - 4ac > 0$ in a domain D the equation is said to be **Hyperbolic**.

2 - If $b^2 - 4ac = 0$ the equation is called **Parabolic**.

3 - If $b^2 - 4ac < 0$ the equation is called **Elliptical**.

Example 3. *1- Equation Elliptical, Stationary Model, Laplace Equation:*

$$\begin{cases} -\Delta u = f, on \quad \Omega. \\ +boundary \quad conditions. \end{cases}$$

2- Equation Parabolic, In stationary Model, Heat Equation:

$$\begin{cases} \dfrac{\partial u}{\partial t} - \Delta u = f, \, on \quad \Omega \times \mathbb{R}_*^+. \\ +boundary \quad conditions + initial \quad conditions. \end{cases}$$

3- Equation Hyperbolic, Wave Equation:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} - \Delta u = f, \, on \quad \Omega \times \mathbb{R}_*^+. \\ +boundary \quad conditions + initial \quad conditions. \end{cases}$$

2.3 Finite Difference Approximations for PDEs

The Finite Difference Method **(FDM)** consists of replacing the derivatives by divided differences or by combinations of point values of the function at a finite number of discrete points.

2.3.1 Discretization

Let I=[a,b] be an interval of $\mathbb{R}$, we say that$(x_k)_{0 \leq k \leq N+1}$ a subdivision of the interval [a,b] if $x_k \in [a,b], \forall\ 0 \leq k \leq N+1$ and verified: $x_0 = a < x_1 < x_2 < ..... < x_N < x_{N+1} = b$, for$i = 0, ..., N$, We notice $h_{i+\frac{1}{2}} = x_{i+1} - x_i$ and we define the pitch of the mesh by $h = max h_{i+\frac{1}{2}}$.

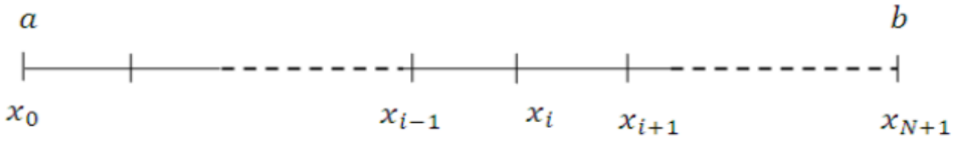


Figure 2.1: Discretization of the domain dimension 1.

2.3.2 Principle of the Difference Method

The principle of the **FDM** consists of giving ourselves a certain number of points in the domain, which we will note$(x_1, x_2, ..., x_N) \in \mathbb{R}^N$ we approach the differential operator by a quotiont in such a way as to deduce a system of equations based on discrete unknowns representing approximations of unknown u at the points discretization first we carry out a development of Taylor in the neighborhood of x_i to order 3.

$$u(x_{i+1}) = u(x_i + h) = u(x_i) + h\frac{\partial u}{\partial x}(x_i) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x_i) + o(h^3).$$
$$u(x_{i-1}) = u(x_i - h) = u(x_i) - h\frac{\partial u}{\partial x}(x_i) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x_i) + o(h^3).$$

Subtracting these two relationships gives.

$$u(x_{i+1}) - u(x_{i-1}) = 2h\frac{\partial u}{\partial x} + o(h^3). \tag{2.12}$$

We notice $u_i = u(x_i), \forall i = 1, ..., N$. The relation (2,12) becomes

$$u_{i+1} - u_{i-1} = 2h\frac{\partial u}{\partial x} + o(h^3).$$

Which allows us to obtain the order 2 scheme **(centered)** to approximate the first derivative of $uinx_i$.

$$\frac{\partial u}{\partial x}(x_i) = \frac{u_{i+1} - u_{i-1}}{2h} + o(h^2).$$

Next we carry out a development of Taylor in the neighborhood of x_i to order 4.

$$u(x_{i+1}) = u(x_i + h) = u(x_i) + h\frac{\partial u}{\partial x}(x_i) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x_i) + \frac{h^3}{6}\frac{\partial^3 u}{\partial x^3} + o(h^4).$$
$$u(x_{i-1}) = u(x_i - h) = u(x_i) - h\frac{\partial u}{\partial x}(x_i) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x_i) - \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3} + o(h^4).$$

By adding the sum of two equalities, we arrive at

$$u_{i+1} + u_{i-1} - 2u_i = h^2\frac{\partial^2 u}{\partial x^2}(x_i) + o(h^4).$$

Which makes it possible to obtain the second order scheme to approximate the second derivative of u.

$$\frac{\partial^2 u}{\partial x^2}(x_i) = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + o(h^2).$$

Other Schemes:

Acoording to the Taylor expansion in the neighborhood of x_i, we have

$$\begin{aligned} u(x_{i+1}) &= u(x_i) + h\frac{\partial u}{\partial x}(x_i) + o(h^2). \\ \Rightarrow \frac{\partial u}{\partial x}(x_i) &= \frac{u(x_{i+1}) - u(x_i)}{h} + o(h). \end{aligned}$$

Scheme off-center front. If we use the point x_{i-1}:

$$\begin{aligned} u(x_{i-1}) &= u(x_i) - h\frac{\partial u}{\partial x}(x_i) + o(h^2). \\ \Rightarrow \frac{\partial u}{\partial x}(x_i) &= \frac{u(x_i) - u(x_{i-1})}{h} + o(h). \end{aligned}$$

Scheme off-center rear.

2.3.3 Simple example in dimension 1 with conditions of Dirichlet

Consider the following differential equation.

$$\begin{cases} -u'' = f(x), x \in [0,1]. \\ u(0) = \gamma;\ u(1) = \delta. \end{cases}$$

(Dirichlet conditions not homogeneous, f).
Using the approximation of u'' by means of a centered scheme of order 2, is thus

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f_i\ ;\ for\ i = 1, ..., N. or\ f_i = f(x_i).$$

The last expression represents a system of N equations and N unknowns can be written as a system as follows.

$$\begin{cases} i=1: & \frac{1}{h^2}\left(-u_0+2u_1-u_2\right)=f_1. \\ i=2: & \frac{1}{h^2}\left(-u_1+2u_2-u_3\right)=f_2. \\ i=3: & \frac{1}{h^2}\left(-u_2+2u_3-u_4\right)=f_3. \\ \vdots & \\ i=N-1: & \frac{1}{h^2}\left(-u_{N-2}+2u_{N-1}-u_N\right)=f_{N-1}. \\ i=N: & \frac{1}{h^2}\left(-u_{N-1}+u_N-u_{N+1}\right)=f_N. \end{cases}$$

This linear system can be written in the following matrix from

$$\frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \cdots & \cdots & \cdots & 0 \\ -1 & 2 & -1 & \ddots & & & \vdots \\ 0 & -1 & 2 & -1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & \cdots & 0 & -1 & 2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ \vdots \\ u_N \end{bmatrix}=\begin{bmatrix} f_1+\frac{\gamma}{h^2} \\ f_2 \\ f_3 \\ f_4 \\ \vdots \\ \vdots \\ f_N+\frac{\delta}{h^2} \end{bmatrix}$$

2.3.4 Problems of Evolution

Consider the one-dimensional problem of heat in a bar length of 1 put. The temperature field $U(x,t)$ verifies the heat equation.

$$\begin{cases} \dfrac{\partial U}{\partial t}-\alpha\dfrac{\partial^2 U}{\partial x^2}=0, x\in\,]0,1[\,,t\in\,]0,T[\,, \\ U(0,t)=U_g \\ U(1,t)=U_d \\ U(x,0)=U_0. \end{cases}$$

The interval $[0,1]$ is discretise in $N+1$ nodes of coordinates $x_i\,, i=0,...,N+1$, h: the discetization step (uniform) of space. Times is discreet in intervals of constant step Δt, such as $\Delta t=\frac{T}{M}$. Let U_i^n denote the temperature at node $x_i=i\Delta x$ and at time $t^n=n\Delta t$. we can use two methods to separate this equation:

2.3.5 An explicit scheme

Here, we take the approximations of $\frac{\partial U}{\partial t}$ by **forward finite differences** and the approximations of $\frac{\partial^2 U}{\partial x^2}$, by **centered finite differences**.

$$\frac{\partial U}{\partial t}(x_i, t_n) = \alpha \frac{\partial^2 U}{\partial x^2}(x_i, t_n).$$
$$\frac{\partial U}{\partial t}(x_i, t_n) = \frac{U_i^{n+1} - U_i^n}{\Delta t}.$$
$$\frac{\partial^2 U}{\partial x^2}(x_i, t_n) = \frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}.$$

Then, the discretized equation is given by

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} - \alpha \frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2} = 0 \ ; \ \ i = 1, ..., N; \ \ n = 0, ..., M-1 \ or$$
$$U_i^{n+1} = \alpha \frac{\Delta t}{\Delta x^2} U_{i+1}^n + (1 - 2\alpha \frac{\Delta t}{\Delta x^2}) U_i^n + \alpha \frac{\Delta t}{\Delta x^2} U_{i-1}^n \ ; i = 1, ..., N \ ; \ n = 0, ..., M-1.$$

We set $\beta = \alpha \frac{\Delta t}{\Delta x^2}$ with $U_0^n = u_g \, , U_{N+1}^n = U_d$. Let it be in matrix from:

$$\begin{bmatrix} 1-2\beta & \beta & 0 & \cdots & \cdots & \cdots & 0 \\ \beta & 1-2\beta & \beta & \ddots & & & \vdots \\ 0 & \beta & 1-2\beta & \beta & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \beta & 1-2\beta & \beta \\ 0 & \cdots & \cdots & \cdots & 0 & \beta & 1-2\beta \end{bmatrix} \begin{bmatrix} U_1^n \\ U_2^n \\ U_3^n \\ U_4^n \\ \vdots \\ \vdots \\ U_N^n \end{bmatrix} + \beta \begin{bmatrix} U_g \\ 0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ U_d \end{bmatrix} = \begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \\ U_3^{n+1} \\ U_4^{n+1} \\ \vdots \\ \vdots \\ U_N^{n+1} \end{bmatrix}$$

Hence

$$\begin{bmatrix} U_1^0 \\ U_2^0 \\ U_3^0 \\ U_4^0 \\ \vdots \\ \vdots \\ U_N^0 \end{bmatrix} = \begin{bmatrix} U_0 \\ U_0 \\ U_0 \\ U_0 \\ \vdots \\ \vdots \\ U_0 \end{bmatrix}$$

Because $U(x;0) = U_0$ or $U_i^0 = U_0; i = 1, ..., N$.

We can calculate the vector U^{n+1} from U^n explicitly.

2.3.6 An Implicit Scheme

In this method, We take the approximation of $\frac{\partial U}{\partial t}$ by **DF backward** and the approximation of $\frac{\partial^2 U}{\partial x^2}$ by **DF centered**.

$$\frac{\partial U}{\partial t}(x_i, t_{n+1}) = \frac{U_i^{n+1} - U_i^n}{\Delta t}.$$
$$\frac{\partial^2 U}{\partial x^2}(x_i, t_{n+1}) = \frac{U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}}{\Delta x^2}.$$

The discretized equation is then given by

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} - \alpha \frac{U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}}{\Delta x^2} = 0.$$

We set $\beta = \alpha \frac{\Delta t}{\Delta x^2}$, with

$$(1 + 2\beta)U_i^{n+1} - \beta U_{i+1}^{n+1} - \beta U_{i-1}^{n+1} = U_i^n.$$

Let it be in matrix from:

$$\begin{bmatrix} 1+2\beta & -\beta & 0 & \cdots & \cdots & \cdots & 0 \\ -\beta & 1+2\beta & -\beta & \ddots & & & \vdots \\ 0 & -\beta & 1+2\beta & -\beta & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & -\beta & 1+2\beta & -\beta \\ 0 & \cdots & \cdots & \cdots & 0 & -\beta & 1+2\beta \end{bmatrix} \begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \\ U_3^{n+1} \\ U_4^{n+1} \\ \vdots \\ \vdots \\ U_N^{n+1} \end{bmatrix} = \begin{bmatrix} U_1^n \\ U_2^n \\ U_3^n \\ U_4^n \\ \vdots \\ \vdots \\ U_N^n \end{bmatrix} + \beta \begin{bmatrix} U_g \\ 0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ U_d \end{bmatrix}$$

Finite Difference Toolkit For Partial Derivatives

This table represents form **FD** approximation toolkit. **FD** approximations to partial derivatives with respect to t and x arederived in a similar and are included in Table.

Partial Derivatives	Finite Difference Approximation	Type	Order
$\frac{\partial U}{\partial x} = U_x$	$\frac{U_{i+1}^n - U_i^n}{\Delta x}$	forward	first in x
$\frac{\partial U}{\partial x} = U_x$	$\frac{U_i^n - U_{i-1}^n}{\Delta x}$	backward	first in x
$\frac{\partial U}{\partial x} = U_x$	$\frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x}$	central	second in x
$\frac{\partial^2 U}{\partial x^2} = U_{xx}$	$\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}$	symmetric	second in x
$\frac{\partial U}{\partial t} = U_t$	$\frac{U_i^{n+1} - U_i^n}{\Delta t}$	forward	first in t
$\frac{\partial U}{\partial t} = U_t$	$\frac{U_i^n - U_i^{n-1}}{\Delta t}$	backward	first in t
$\frac{\partial U}{\partial t} = U_x$	$\frac{U_i^{n+1} - U_i^{n-1}}{2\Delta t}$	central	second in t
$\frac{\partial^2 U}{\partial x^2} = U_{tt}$	$\frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{\Delta t^2}$	symmetric	second in t

Table 2.1: Finite Difference Toolkit For Partial Derivatives.

2.4 The Finite Element Method

The Finite Element Method **(FEM)** is a numerical technique used to solve differential equations arising from physical problems in engineering and applied sciences. It is particularly useful for problems involving complex geometries, material properties, and boundary conditions LaTeX" [7, Sec 2.4].

2.4.1 Variational Formulation

Consider the Poisson problem in one dimension with homogeneous Dirichlet boundary conditions: Find a function u defined on $I = [0, 1]$, such that

$$\begin{cases} -u''(x) = f(x)\,; \quad x \in (0,1) \\ u(0) = u(1) = 0 \end{cases}$$

Through this example we aim to provide basic ideas about **FEM** and this by adopting **the Galerkin Method**. Let's be the $v(x) \in C^1([0,1])$. We can write:

$$-\int_0^1 u''(x)v(x)dx = \int_0^1 f(x)v(x)dx.$$

The function v is called test function. If we make an integration by parts in the left-hand side of this equality, We get:

$$\int_0^1 u'(x)v'(x) - [u'(x)v(x)]_0^1 dx = \int_0^1 f(x)v(x)dx.$$

And if v is a function such as $v(0) = v(1) = 0$, Then:

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 f(x)v(x)dx$$

.

For this, We consider spaces of the following functions:

$$L^2(\Omega) = \left\{ u \in C^1(\Omega), \int_\Omega |u|^2 < \infty \right\}, \quad (\Omega \subset \mathbb{R}).$$
$$H_0^1 = \left\{ u \in L^2(\Omega)\,,\, u' \in L^2(\Omega)\,,\, u = 0 \text{ to } \partial\Omega \right\}.$$

Definition 2.1. *We call integral formulation or variational formulation of the problem, the problem is which consists of*
Find a function u belonging to the space H_0^1, such as

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 f(x)v(x)dx \quad for\, any\, function\, v \in H_0^1 \tag{2.13}$$

Remarque 2.1. *A solution of the variational form (2,13) is called a weak solution of the initial differential problem.*

2.4.2 Conditions at the linits of Neumann

Now we consider the following problem where boundary conditions are given by the value of the derivative of the unknown u.
Find a function u set to $[0, 1]$ such as

$$\begin{cases} u''(x) + u(x) = f(x) & x \in (0,1) \\ u'(0) = \beta \quad , u'(1) = \eta. \end{cases}$$

The integral formulation for this type of problem differs by taking into account boundary conditions.We take the approach that allows us to write a variational formulation. We assume that u is a solution and we multiply the equation by a fontion regular test v then we integrate on $[0,1]$.

$$-\int_0^1 (u''(x)+u(x))v(x)dx = \int_0^1 f(x)v(x)dx.$$

After integration by part, we obtain

$$\int_0^1 u'(x)v'(x)dx - u'(x)v(x)|_0^1 + \int_0^1 u(x)v(x)dx = \int_0^1 f(x)v(x)dx.$$

That is to say

$$\int_0^1 u'(x)v'(x)dx + \int_0^1 u(x)v(x)dx = \int_0^1 f(x)v(x)dx + \eta v(1) - \beta v(0). \tag{2.14}$$

Since $u'(0)=\beta$ and $u'(1)=\eta$. Thus, the Neumann boundary condition is directly taken in comnpte in the variational formulation, and it is not necessary to include it in the choice of the resolution vector space. For (2,14) to be defined, it is sufficient that u and v are continuous and C^1 in pieces on $I=[0,1]$. So we introduce the space

$$H^1 = \left\{u \in L^2(I), u' \in L^2(I)\right\}.$$

So,The variational formulation of this problem becomes.
Find $u \in H^1(I)$, such that

$$\int_0^1 u'(x)v'(x)dx + \int_0^1 u(x)v(x)dx = \int_0^1 f(x)v(x)dx + \eta v(1) - \beta v(0), \forall v \in H^1(I).$$

In this case

$$a(u,v) = \int_0^1 u'(x)v'(x)dx + \int_0^1 u(x)v(x)dx,$$
$$l(v) = \int_0^1 f(x)v(x)dx + \eta v(1) - \beta v(0).$$

2.4.3 The Lax-Milgram Theorem

Definition 2.2. ***Coercivity***

Let H be a Hilbert space.

Let a: $H \times H \longrightarrow \mathbb{R}$ be a bilinear form, we say that a is ***coercive*** *(or H-elliptic) if*

$$\exists \alpha > 0,\ \forall u \in H, a(u,u) \geq \alpha ||u||^2.$$

Remarque 2.2. ***Continuity of a bilinear form***

Let H be a Hilbert space. The bilinear form $a : H \times H \longrightarrow \mathbb{R}$ is continuous if

$$\exists C > 0,\ \forall u, v \in H, a(u,v) \leq C||u||\,||v||.$$

Note that linearity implies continuity in finite-dimensional spaces. It is not the case in infinite-dimensional spaces.

Theorem 2.1. ***Lax-Milgram***

Let H be a Hilbert space.

Let a be a continuous and coercive bilinear form. Let $l \in H'$

The equation

$$\forall v \in H, a(u,v) = l(v).$$

- *Has one and only one solution u.*
- *The application $l \mapsto u$ is linear and continuous from $H'toH$.*

2.4.4 Galerkin's Method

We now want to find a method to calculate approximate solutions to a Poisson Problem. As we also want to be able to deal with cases where the solution is not regular, we will in fact

solve the variational formulation in an approximate manner. The variational formulation that we have seen can be put in the form.
Find a function u in V, such that

$$a(u,v) =< f, v > \ for\ any\ u \in V. \tag{2.15}$$

V a Hilbert space equipped with the scalar product $(,)$ and its assciated norm $||\ ||_V$

- $(u,v) \longmapsto a(u,v)$ is a bilinear form on $V \times V$.
- $v \longmapsto < f, v >$ is linear form on V.

In the case of the of the Poisson problem with homogeneous Dirichlet boundary conditions, the vector space where we seek the solution is $V = H_0^1(0,1)$ and the shape a, and $(f,.)$ are defined by

$$a(u,v) = \int_0^1 u'(x)v'(x)dx \quad < f, v >= \int_0^1 f(x)v(x)dx.$$

The resolution space V being of on finite dimension, the principle of Galerkin's method consists of replacing it by a vector space V of finite dimension and solving the approximate problem.
Finding a function u, in V a such that

$$a(u_h, v_h) =< f, v_h >, \quad for\ any\ function\ v \in V_h. \tag{2.16}$$

In this course, we will always take spaces $V_h \subset V$, we then speak of con formal approximation. As V_h is of finite dimension, the resolution of (2,16) actually amounts to the resolution of a linear system. Indeed, if we give ourselves a base $(\varphi_1, \varphi_2, ..., \varphi_N)$ of V_h, the solution sought can be decomposed on this basis according to

$$u_h(x) = \sum_{j=1}^{N} u_j \varphi_j(x).$$

Solving(2,16)is then equivalent to finding a vector $U = (u_1, u_2, ..., u_N) \in \mathbb{R}^N$

$$a\left(\sum_{j=1}^{N} u_j \varphi_j(x), v_h\right) =< f, v_h >, \quad \forall\, v_h \in V_h.$$

$$\iff \sum_{j=1}^{N} a(\varphi_j(x), v_h) u_j =< f, v_h >, \quad \forall\, v_h \in V_h.$$

$$\iff \sum_{j=1}^{N} a(\varphi_j(x), \varphi_i(x)) u_j =< f, \varphi_i(x) >, \quad \forall i = 1, ..., N.$$

Such that since $(\varphi_1, \varphi_2, ..., \varphi_N)$ is a base of V_h. By laying the matrix

$$A = (A_{i,j})_{1 \leqslant i,j \leqslant N} \in \mathbb{R}^{N \times N},\ A_{i,j} = a(\varphi_j(x), \varphi_i(x))$$

and the vector

$$F = (F_i)_{1 \leqslant i \leqslant N} \in \mathbb{R}^N,\ F_i =< f, \varphi_i(x) > .$$

That is solve the linear system

$$\begin{pmatrix} a(\varphi_1, \varphi_1) & \cdots & \cdots & a(\varphi_N, \varphi_1) \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ a(\varphi_1, \varphi_N) & \cdots & \cdots & a(\varphi_N, \varphi_N) \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} \langle f, \varphi_1 \rangle \\ \vdots \\ \vdots \\ \langle f, \varphi_N \rangle \end{pmatrix}$$

In matrix form we write this

$$AU = F.$$

Matrix A is called stiffness matrix, referring to the mechanical problems where it was first introduced. this matrix has properties that come directly form the properties satisfied by the bilinear form a and F as the loading vector.

2.4.5 Estimate of the error

Lemma 2.1. ***(Galarkin orthogonality)***

Be $u, \epsilon V$ the exact solution of the continuous problem (2,15) and u_h the exact solution of the

discrete problem (2,16). Then the error $e = u - u_h$ satisfies

$$a(e, \upsilon_h) = 0 \quad \forall\, \upsilon_h \epsilon\, V_h. \tag{2.17}$$

Proof

The exact solution u satisfied $a(u, v) = l(v)$ for any υ, ϵ and V. Since $V_h \subset V$, we can take a test function of V_h and eduire that $a(u, \upsilon_h) = l(\upsilon_h)$ for any $\upsilon_h \epsilon\, V_h$. Using $a(u_h, \upsilon_h) = l(\upsilon_h)$ for any $\upsilon_h \epsilon\, V_h$, we get the result immediately.

Lemma 2.2. ***Céa's lemma:***
Under the hypotheses of the ***Lax-Milgram*** *theorem, be u the solution of (2,15) and u_h the solution of (2,16). Then*

$$||u - u_h|| \leq \frac{M}{\alpha}||u - \upsilon_h||, \forall \upsilon_h \epsilon\, V_h \quad ie \quad ||u - u_h|| \leq \frac{M}{\alpha} d(u, V_h).$$

Where d is the distance induced by the standard $||.||$.

Proof:

we got

$$\begin{aligned} a(u - u_h, u - u_h) &= a(u - u_h, u - \upsilon_h + \upsilon_h - u_h) \quad \forall\, \upsilon_h \epsilon\, V_h. \\ &= a(u - u_h, u - \upsilon_h) + a(u - u_h, \upsilon_h - u_h). \end{aligned}$$

but $\upsilon_h - u_h \in V_h$. Hence, according (2,17) $a(u - u_h, \upsilon_h - u_h) = 0$.
So, we have

$$a(u - u_h, u - \upsilon_h) = a(u - u_h, u - \upsilon_h) \quad \forall\, \upsilon_h \epsilon\, V_h, \tag{2.18}$$

a being coercive, there is $\alpha > 0$ such that $a(u - u_h, u - u_h) \geqslant \alpha||u - u_h||^2$ or $\|.\|$ is a standard on V. In addition, a being continuous, there are $a(u - u_h, u - \upsilon_h) \leqslant M||u - u_h||||u - \upsilon_h||$.
By re-injection these two inequalities on both sides of (2.18) and simplifying by $||u - u_h||$, we obtain

$$||u - u_h|| \leqslant \frac{M}{\alpha}||u - u_h|| \quad \forall\, \upsilon_h \epsilon\, V_h.$$

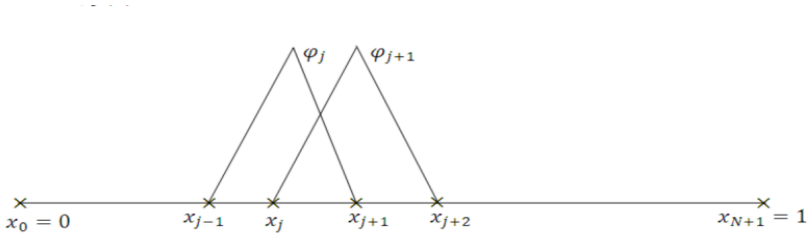


Figure 2.2: Functions The base φ_j.

2.4.6 The Finite Element Method in 1D

We will consider the model problem of the introduction:
Find a function u belonging to the space $H_0^1(0,1)$ such that as

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 f(x)v(x)dx,\ for\ any\ function\ v \in H_0^1(0,1)$$

To construct the space V_h, a discretization is introduced, as in the case of different finite, with a uniform grid

$$x_j = j\Delta x\,;\ j = 0, ..., N$$

where $\Delta x = \frac{1}{N+1}$, $weposeh = \Delta x$. Then, for $1 \leq j \leq N$, We introduce the fuctions φ_j defined by

$$\varphi_i(x_j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{else.} \end{cases} \forall i = 1, \ldots, N; j = 1, \ldots, N.$$

Hence

$$\varphi_j(x) = \begin{cases} \frac{x - x_{j-1}}{x_j - x_{j-1}} & \text{if } x_{j-1} \leq x \leq x_j \\ \frac{x - x_{j+1}}{x_j - xj+1} & \text{if } x_j \leq x \leq x_{j+1} \\ 0 & \text{else.} \end{cases}$$

Functions φ_j are polynomials of degree 1. These functions are called Finite Elements of degree 1. With these finite elements, the matrix A is tridiagonal. It is also possible to choose functions of degree 2 or more for **FE**.

The calculation of the matrix A involves the derivatives $\varphi'_j(x)$ simple to calculate

$$\varphi'_j(x) = \begin{cases} \frac{1}{x_j - x_{j-1}} & \text{if } x_{j-1} \leq x \leq x_j \\ \frac{1}{x_j - xj+1} & \text{if } x_j \leq x \leq x_{j+1} \\ 0 & \text{else.} \end{cases}$$

Let us now calculate the coefficients $A_{i,j}$ summing the contributions of the various elements according to:

$$A_{i,j} = a(\varphi_j(x), \varphi_i(x)) = \int_0^1 \varphi'_i(x)\varphi'_j(x)dx = \sum_{k=1}^{N} \int_{x_k}^{x_{k+1}} \varphi'_i(x)\varphi'_j(x)dx.$$

Consider for example the element $T_i = [x_i, x_{i+1}]$. On this element, there are only two non-zero basis functions:φ_i and φ_{i+1}. The element T will therefore effectively produce a non-zero contribution to the four coefficients $A_{i,i}, A_{i+1,i}, A_{i+1,i+1}\, and\, A_{i,i+1}$ of the global matrix A. Let us calculate the elementary contributions of T_i and put them in the form of a 2×2 elementary matrix

$$ElemT_i = \begin{pmatrix} e^i_{1,1} & e^i_{1,2} \\ e^i_{2,1} & e^i_{2,2} \end{pmatrix}$$

With

$$e^i_{1,1} = a(\varphi_i(x), \varphi_i(x)) = \int_{x_i}^{x_{i+1}} \varphi'_i(x)\varphi'_i(x)dx = \frac{1}{x_{i+1} - x_i}$$
$$e^i_{1,2} = e^i_{2,1} = \int_{x_i}^{x_{i+1}} \varphi'_i(x)\varphi'_{i+1}(x)dx = -\frac{1}{x_{i+1} - x_i}$$
$$e^i_{2,2} = \int_{x_i}^{x_{i+1}} \varphi'_i(x)\varphi'_{i+1}(x)dx = \frac{1}{x_{i+1} - x_i}$$

So

$$ElemT_i = \frac{1}{x_{i+1} - x_i} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

for the $F_j =< f, \varphi_j(x) >$, we must use an approximate integral calculation method for example, the trapezoid method.

$$F_j = \int_0^1 f(x)\varphi_j(x)dx = (\frac{x_{i+1} - x_{i-1}}{2})f_j$$

2.4.7 Finite Elements For a Non-Linear Problem

Consider the following nonlinear problem:

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + u\dfrac{\partial u}{\partial x} - u = e^{2x}, \quad x \in (0,1) \\ u(0) = 1 \quad u(1) = e \end{cases}$$

It is clear that $u = e$ is the exact solution of that's problem.

We will solve this problem numerically using the finite element method. We will explain all the steps in detail to facilitate the understanding of the algorithm on which we relied in programming.

Domain Discretization

We subdivide the interval [0,1] on 5 under intevalles (N=5), where the length of each sub intevalle is $h_e = \frac{1}{N} = 0,2$.

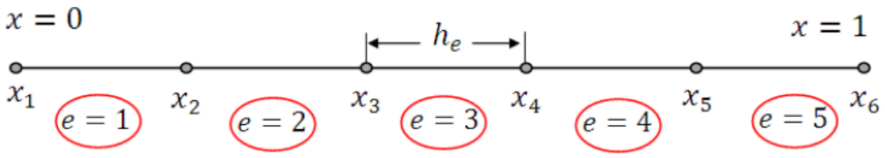


Figure 2.3: Mesh consisting of five elements.

Variational Formulation

$$\int_{x_e}^{x_{e+1}} w\frac{\partial^2 u}{\partial x^2} + \bar{u}\frac{\partial u}{\partial x} - u - e^{2x}dx = 0$$

By partial integration we obtain

$$\int_{x_e}^{x_{e+1}} \frac{\partial w}{\partial x}\frac{\partial u}{\partial x} + wu\frac{\partial u}{\partial x} - w\bar{u}dx = \left(w\frac{\partial u}{\partial x}\right)_{x_e}^{x_{e+1}} + \int_{x_e}^{x_{e+1}} we^{2x}dx$$

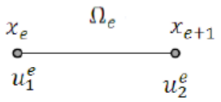
So, The variational formulation of the problem

$$\int_{x_e}^{x_{e+1}} \frac{\partial w}{\partial x}\frac{\partial u}{\partial x} + wu\frac{\partial u}{\partial x} - w\bar{u}dx = \int_{x_e}^{x_{e+1}} we^{2x}dx + w(x_e)Q_1 + w(x_{e+1})Q_2. \tag{2.19}$$

Or

$$Q_1 = -\left(\frac{\partial u}{\partial x}\right)_{x_e} \quad and \quad Q_2 = \left(\frac{\partial u}{\partial x}\right)_{x_{e+1}}$$

2.4.8 Linear approximation of $u(x)$



$$u^e(x) = c_1 + c_2 x$$

By applying the conditions

$$\left.\begin{aligned} u^e(x_e) &= c_1 + c_2 x_e = u_1^e \\ u^e(x_{e+1}) &= c_1 + c_2 x_{e+1} = u_2^e \end{aligned}\right\} \Rightarrow u^e(x) = u_1^e \varphi_1^e(x) + u_2^e \varphi_2^e(x) = \sum_{j=1}^{j=2} u_j^e \varphi_j^e(x)$$

Or

$$\varphi_1^e(x) = \frac{x_{e+1} - x}{x_{e+1} - x_e} \ , \ \varphi_2^e(x) = \frac{x - x_e}{x_{e+1} - x_e}.$$

Are know as basic functions.

Proprities of basic functions

1 $\varphi_i^e(x_j^e) = \begin{cases} 1 & \text{if} i = j, \\ 0 & \text{if} i \neq j \end{cases}$

2 $\sum_{j=1}^{j=2} \varphi_j^e(x) = 1$

The finite element model can be obtained from equation (2,19) by substituting the linear finite element approximation of the form

$$u^e(x) = \sum_{j=1}^{j=2} a_j^e \varphi_j^e(x).$$

Which

$$w = \varphi_i^e(x), i = 1, 2, ...$$

Where $\varphi_i^e(x)$ are the basic functions on the element (x_e, x_{e+1}). substitution of the values of u and w in equation (2,19)

$$\int_{x_e}^{x_{e+1}} \left[\frac{\partial \varphi_1^e}{\partial x} \frac{\partial}{\partial x}(u_1^e \varphi_1^e + u_2^e \varphi_2^e) + \varphi_1^e \bar{u} \frac{\partial}{\partial x}(u_1^e \varphi_1^e + u_2^e \varphi_2^e) - \varphi_1^e (u_1^e \varphi_1^e + u_2^e \varphi_2^e) \right] dx$$
$$= \varphi_1^e(x_e) Q_1 + \varphi_1^e(x_{e+1}) Q_2 + \int_{x_e}^{x_{e+1}} \varphi_1^e e^{2x} dx.$$

$$\int_{x_e}^{x_{e+1}} \left[\frac{\partial \varphi_2^e}{\partial x} \frac{\partial}{\partial x}(u_1^e \varphi_1^e + u_2^e \varphi_2^e) + \varphi_2^e \bar{u} \frac{\partial}{\partial x}(u_1^e \varphi_1^e + u_2^e \varphi_2^e) - \varphi_2^e (u_1^e \varphi_1^e + u_2^e \varphi_2^e) \right] dx$$
$$= \varphi_2^e(x_e) Q_1 + \varphi_2^e(x_{e+1}) Q_2 + \int_{x_e}^{x_{e+1}} \varphi_2^e e^{2x} dx.$$

In matrix notation, algebric equation can be written as

$$K_{11}^e u_1^e + K_{12}^e u_2^e = f_1^e + Q_1^e$$
$$K_{21}^e u_1^e + K_{22}^e u_2^e = f_2^e + Q_2^e$$

Or in other words

$$[K^e] \{u^e\} = \{f^e\} + \{Q^e\}.$$

Such that

$$K_{ij}^e = \int_{x_e}^{x_{e+1}} \left[\frac{\partial \varphi_i^e}{\partial x} \frac{\partial \varphi_j^e}{\partial x} + \varphi_i^e \bar{u} \frac{\partial \varphi_j^e}{\partial x} - \varphi_i^e \varphi_j^e \right] dx \;\; ; \;\; f_i^e = \int_{x_e}^{x_{e+1}} \varphi_i^e e^{2x} dx.$$

Suppose we get

$$\bar{u} = \sum_{j=1}^{j=2} \bar{u} \varphi_j^e(x) = \bar{u}_1 \varphi_1^e(x) + \bar{u}_2 \varphi_2^e(x)$$

Then

$$K_{ij}^e = \int_{x_e}^{x_{e+1}} \left(\frac{\partial \varphi_i^e}{\partial x} \frac{\partial \varphi_j^e}{\partial x} + \left[\varphi_i^e \varphi_1^e \frac{\partial \varphi_j^e}{\partial x} \right] \bar{u}_1 + \left[\varphi_i^e \varphi_2^e \frac{\partial \varphi_j^e}{\partial x} \right] \bar{u}_2 - \varphi_i^e \varphi_j^e \right) dx$$
$$= A_{ij}^e + B_{ij}^e \bar{u}_1 + C_{ij}^e \bar{u}_2 + D_{ij}^e$$

Such that

$$A_{ij}^e = \int_{x_e}^{x_{e+1}} \left(\frac{\partial \varphi_i^e}{\partial x} \frac{\partial \varphi_j^e}{\partial x} \right) dx \quad ; \quad B_{ij}^e = \int_{x_e}^{x_{e+1}} \left(\varphi_i^e \varphi_1^e \frac{\partial \varphi_j^e}{\partial x} \right) dx$$
$$C_{ij}^e = \int_{x_e}^{x_{e+1}} \left(\varphi_i^e \varphi_2^e \frac{\partial \varphi_j^e}{\partial x} \right) dx \quad ; \quad D_{ij}^e = - \int_{x_e}^{x_{e+1}} \left(\varphi_i^e \varphi_j^e \right) dx$$

By calculating these integrals, we find

$$A_{ij}^e = \int_{x_e}^{x_{e+1}} \left(\frac{\partial \varphi_i^e}{\partial x} \frac{\partial \varphi_j^e}{\partial x} \right) dx = \frac{1}{h_e} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$B_{ij}^e = \int_{x_e}^{x_{e+1}} \left(\varphi_i^e \varphi_1^e \frac{\partial \varphi_j^e}{\partial x} \right) dx = \frac{1}{6} \begin{pmatrix} 2 & -2 \\ 1 & -1 \end{pmatrix}$$

$$C_{ij}^e = \int_{x_e}^{x_{e+1}} \left(\varphi_i^e \varphi_2^e \frac{\partial \varphi_j^e}{\partial x} \right) dx = \frac{1}{6} \begin{pmatrix} 1 & -1 \\ 2 & -2 \end{pmatrix}$$

$$D_{ij}^e = - \int_{x_e}^{x_{e+1}} \left(\varphi_i^e \varphi_j^e \right) dx = \frac{h_e}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

$$f_i^e = - \int_{x_e}^{x_{e+1}} \varphi_i^e e^{2x} dx = \begin{pmatrix} \frac{1}{4h_e}(e^{2x_e} - e^{2x_{e+1}}) + \frac{e^{2x_e}}{2} \\ \frac{1}{4h_e}(e^{2x_{e+1}} - e_{2x_e}) + \frac{e^{2x_{e+1}}}{2} \end{pmatrix}$$

2.4.9 Assembly Technique

Initiation of the linear structure

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

1^{st} **Element** $e = 1$**:**

$$\begin{pmatrix} K_{11}^1 & K_{12}^1 & 0 & 0 & 0 & 0 \\ K_{21}^1 & K_{22}^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} f_1^1 \\ f_2^1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} Q_1^1 \\ Q_2^1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

2^{nd} **Element** $e = 2$**:**

$$\begin{pmatrix} K_{11}^1 & K_{12}^1 & 0 & 0 & 0 & 0 \\ K_{21}^1 & (K_{22}^1 + K_{11}^2) & K_{12}^2 & 0 & 0 & 0 \\ 0 & K_{21}^2 & K_{22}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} f_1^1 \\ f_2^1 + f_1^2 \\ f_2^2 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} Q_1^1 \\ Q_2^1 + Q_1^2 \\ Q_2^2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3^{rd} **Element** $e = 3$**:**

$$\begin{pmatrix} K_{11}^1 & K_{12}^1 & 0 & 0 & 0 & 0 \\ K_{21}^1 & (K_{22}^1 + K_{11}^2) & K_{12}^2 & 0 & 0 & 0 \\ 0 & K_{21}^2 & (K_{22}^2 + K_{11}^3) & 0 & 0 & 0 \\ 0 & 0 & K21^3 & K_{22}^3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} f_1^1 \\ f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ f_2^3 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} Q_1^1 \\ Q_2^1 + Q_1^2 \\ Q_2^2 + Q_1^3 \\ Q_2^3 \\ 0 \\ 0 \end{pmatrix}$$

4^{th} **Element** $e = 4$**:**

$$\begin{pmatrix} K_{11}^1 & K_{12}^1 & 0 & 0 & 0 & 0 \\ K_{21}^1 & (K_{22}^1 + K_{11}^2) & K_{12}^2 & 0 & 0 & 0 \\ 0 & K_{21}^2 & (K_{22}^2 + K_{11}^3) & 0 & 0 & 0 \\ 0 & 0 & K21^3 & (K_{22}^3 + K_{11}^4) & K_{12}^4 & 0 \\ 0 & 0 & 0 & K_{21}^4 & K_{22}^4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} f_1^1 \\ f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ f_2^3 + f_1^4 \\ f_2^4 \\ 0 \end{pmatrix} + \begin{pmatrix} Q_1^1 \\ Q_2^1 + Q_1^2 \\ Q_2^2 + Q_1^3 \\ Q_2^3 + Q_1^4 \\ Q_2^4 \\ 0 \end{pmatrix}$$

5^{th} **Element** $e = 5$**:**

$$\begin{pmatrix} K_{11}^1 & K_{12}^1 & 0 & 0 & 0 & 0 \\ K_{21}^1 & (K_{22}^1 + K_{11}^2) & K_{12}^2 & 0 & 0 & 0 \\ 0 & K_{21}^2 & (K_{22}^2 + K_{11}^3) & 0 & 0 & 0 \\ 0 & 0 & K21^3 & (K_{22}^3 + K_{11}^4) & K_{12}^4 & 0 \\ 0 & 0 & 0 & K_{21}^4 & (K_{22}^4 + K_{11}^4) & K_{12}^5 \\ 0 & 0 & 0 & 0 & K_{21}^5 & K_{22}^5 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} f_1^1 \\ f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ f_2^3 + f_1^4 \\ f_2^4 + f_1^5 \\ f_2^5 \end{pmatrix} + \begin{pmatrix} Q_1^1 \\ Q_2^1 + Q_1^2 \\ Q_2^2 + Q_1^3 \\ Q_2^3 + Q_1^4 \\ Q_2^4 + Q_1^5 \\ Q_2^5 \end{pmatrix}$$

The second member of the linear system is evaluted as follows

$$Q_2^e + Q_1^{e+1} = \begin{cases} 0 & \text{if no external point source isapplied} \\ Q* & \text{if an external point source of magnitude Q* is applied} \end{cases}$$

where $(Q* = Q_o.)$

Imposition of conditions on limits

$$\begin{pmatrix} K_{11}^1 & K_{12}^1 & 0 & 0 & 0 & 0 \\ K_{21}^1 & (K_{22}^1 + K_{11}^2) & K_{12}^2 & 0 & 0 & 0 \\ 0 & K_{21}^2 & (K_{22}^2 + K_{11}^3) & 0 & 0 & 0 \\ 0 & 0 & K21^3 & (K_{22}^3 + K_{11}^4) & K_{12}^4 & 0 \\ 0 & 0 & 0 & K_{21}^4 & (K_{22}^4 + K_{11}^4) & K_{12}^5 \\ 0 & 0 & 0 & 0 & K_{21}^5 & K_{22}^5 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} f_1^1 \\ f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ f_2^3 + f_1^4 \\ f_2^4 + f_1^5 \\ f_2^5 \end{pmatrix} + \begin{pmatrix} Q_1^1 \\ 0 \\ 0 \\ 0 \\ 0 \\ Q_2^5 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ K_{21}^1 & (K_{22}^1 + K_{11}^2) & K_{12}^2 & 0 & 0 & 0 \\ 0 & K_{21}^2 & (K_{22}^2 + K_{11}^3) & 0 & 0 & 0 \\ 0 & 0 & K21^3 & (K_{22}^3 + K_{11}^4) & K_{12}^4 & 0 \\ 0 & 0 & 0 & K_{21}^4 & (K_{22}^4 + K_{11}^4) & K_{12}^5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} u(0) \\ f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ f_2^3 + f_1^4 \\ f_2^4 + f_1^5 \\ u(1) \end{pmatrix}$$

After assembling the equation of the elements, a system of nonlinear equations is obtained, therefore an iterative scheme was applied to solve it. This system is linearized by incorporating the function $\bar{u}$, which is assumed to be known from the previous iteration and the calculations for u are then carried out for the current iteration. This process is repeated until the desired accuracy is achieved.

2.5 Finite Volume Method

The Finite Volume Method (**FVM**) is a discretization technique for Partial Differential Equations, especially those that arise from physical conservation laws. **FVM** uses a volume integral formilation of the problem with a finite partitioning set of volumes to discretize the equations. **FVM** is in common use for discretizing computational fluid dynamics equations LaTeX"[16, Sec 2.5].

2.5.1 Finite volume diagram for an elliptic problem in 1 dimension of space

Origin of the schema

We will study the discretization by finite volumes of the problem

$$\begin{cases} -u_{xx} = f, \ \ x \in]0,1[\,, \\ u(0) = u(1) = 0. \end{cases} \tag{2.20}$$

2.5.2 Finite Volume mesh

We call finite volume mesh of the interval $[0,1]$, a set of N meshes $(K_i) = 1, ..., N$, such that $K_i = \left]x_{i-1/2}, x_{i+1/2}\right[$, with $x_{\frac{1}{2}} = 0 < x_{\frac{3}{2}} < ... < x_{\frac{i-1}{2}} < x_{\frac{i+1}{2}} < ... < x_{\frac{N+1}{2}} = 1$, and we note $K_i = x_{i+1/2} - x_{i-1/2}$. We also give ourselves N points $(x_i)_{i=1,...,N}$
located in the cells K_i. So, we have

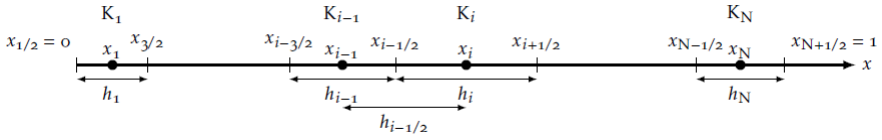


Figure 2.4: One-dimensional mesh in finite volumes.

$$0 = x_{\frac{1}{2}} < x_1 < x_{\frac{3}{2}} < ... < x_{\frac{i-1}{2}} < x_{\frac{i+1}{2}} < ... < x_{\frac{N+1}{2}} = 1$$

We will note $h_{i+1/2} = x_{i+1} - x_i$, and $h = max_{i=1,...,N}$, and for notational questions, we will also ask $x_0 = 0$ and $x_{N+1} = 1$.
We integrate (2,20) over $K_i = x_{i+1/2} - x_{i-1/2}$, and we obtain:

$$-u_x(x_i + 1/2) + u_x(x_i - 1/2) = \int_{K_i} f(x)dx. \tag{2.21}$$

We set: $f_i = \frac{1}{h_i}\int_{(K_i)} f(x)dx$, and we introduce the unknowns $(u_i) = 1, ..., N$ and the discrete equations of the numerical scheme:

$$F_{i+1/2} - F_{i-1/2} = h_i f_i \ \ ; \ \ \ i = 1, ..., N$$

Where $F_{i+1/2}$ is the digital flow in $x_{i+1/2}$ which should be a reasonable approximation of $-u_x(x_i+1/2)$, we ask then:

$$F_{i+1/2} = -\frac{u_{i+1} - u_i}{h_{i+1/2}}\,;\;\; i = 1, ..., N$$
$$F_{1/2} = -\frac{u_1}{h_{1/2}};\;\; F_{N+1/2} = -\frac{u_N}{h_{N+1/2}}$$

To take into account the homogeneous Dirichlet boundary conditions $u(0) = u(1) = 0$. We can also

$$F_{i+1/2} = -\frac{u_{i+1} - u_i}{h_{i+1/2}}\,;\;\; i = 0, ..., N$$
$$by\, setting\, u_0 = u_{N+1} = 0$$

We can write the linear system obtained on $(u_1, u_2, ..., u_N)^t$ in the from

$$A_h u_h = b_h.$$

With

$$(A_h) = \frac{1}{h_i}\left[\frac{-1}{h_{i+1/2}}(u_{i+1} - u_i) + \frac{1}{h_{i-1/2}}(u_i - u_{i-1})\right] \; and \; (b_h) = f_i$$

Remarque 2.3. ***(Non-consistency in the sense of finite differences)***
The approximation of $-u''(x_i)$ *by*

$$\frac{1}{h_i}\left[\frac{-1}{h_{i+1/2}}(u(x_{i+1}) - u(x_i)) + \frac{1}{h_{i-1/2}}(u(x_i) - u(x_{i-1}))\right]$$

Chapter 3

Solving PDEs with MATLAB

In this chapter we wanted to present some problems with how to write them in MATLAB and show the solutions.

3.1 How to write the code MATLAB?

The first chapter discussed the definition of Matlab and its properties, and in the second chapter we studied numerical solution methods for Partial Differential Equations. In this chapter, we will combine what we studied previously to obtain a program that enables us to solve the required equation.

3.1.1 The code MATLAB for the First Order Partial Differential Equations

we chose to study a First Order Partial Differential Equations in the Euler Explicit Method for this problem:

$$\begin{cases} u_t - u_x = 0 \\ u(x,0) = f(x) \end{cases}$$

Here is the Matlab code for it:

```
%the Euler Explicit Method to the First Order Partial Differential Equations.
N=20;
dx=0.1;
dt=0.1;
x=0:dx:5;
t=0:dt:1;
f = @(x)exp(-x.^2);
U = zeros(length(x),length(t));
U(:,1) = f(x) ;
for n = 1:(length(t)-1)
    for i = 2:(length(x)-1)
        U(i,n+1) = U(i,n)+(dt/(2*dx))*(U(i+1,n)-U(i-1,n));
    end
end
plot(x,U(:,end))
```

3.1.2 The code MATLAB for the Second-Order Partial Differential Equations

Since we also touched in a previous chapter on studying second-order partial differential equations and solving them using three numerical methods, we wanted to solve an example of each method.

First:

Example 3.1. *An explicit scheme of Finite Difference method to solve the problem:*

$$\begin{cases} \frac{\partial U}{\partial t} - \alpha \frac{\partial^2 U}{\partial x^2} = 0, x \in]0,1[, t \in]0,T[, \\ U(0,t) = U_g \\ U(1,t) = U_d \\ U(x,0) = U_0. \end{cases}$$

Here is the Matlab code for it:

```
%Finite Difference method to An Explicit Scheme .
```

```
clc
clear
a=0;
b=1;
N=60;
M=30;
h=(b-a)/(N+1);
T=1;
k=T/M;
r=k/h^2;
syms t x
f(t,x)=(1+pi^2*t)*sin(pi*x);
g(x)=0*x;
for i=1:N
    X(i)=a+i*h;
    U(i)=g(X(i));
end
for n=1:10
  V(1)=r*U(2)+(1-2*r)*U(1)+k*f(n*k,X(1));
for i=2:N-1
    V(i)=r*U(i+1)+(1-2*r)*U(i)+r*U(i-1)+k*f(n*k,X(i));
end
   V(N)=(1-2*r)*U(N)+r*U(N-1)+k*f(n*k,X(N));
U=V;
end
plot(X,U,'*')
hold on
fplot*(10*k*sin(pi*x),[a b])
```

Second:

Example 3.2. *Finite Difference method to solve the problem:*

$$\begin{cases} -u''(x) = \pi^2 \sin(\pi x); \quad x \in (0,1) \\ u(0) = \alpha \ , \ u(1) = \beta \end{cases}$$

Here is the Matlab code for it:

```
%Finite Difference Method to Laplace Equation.
clc
clear
a=0;
b=1;
N=40;
h=(b-a)/(N+1);
syms t
f(t)=pi^2*sin(pi*t);
S(t)=sin(pi*t);
A=zeros(N,N);
for i=2:N-1
    A(i,i)=2;
    A(i-1,i)=-1;
    A(i+1,i)=-1;
end
A(N,N)=2;
A(1,1)=2;
A(2,1)=-1;
A(N-1,N)=-1;
for i=1:N
    x(i)=a+i*h;
B(i)=h^2*f(x(i));
end
u=A^-1*B';
for i=1:N
E(i)=abs(S(x(i))-u(i))
end
subplot(2,1,1)
plot(x,u,'*')
hold on
fplot(S,[a b])
subplot(2,1,2)
plot(x,E)
```

Next:

Finite Element Method to solve the problem:

Example 3.3.

$$\begin{cases} -u''(x) = \pi^2 \sin(\pi x)\,; \quad x \in (0,1) \\ u(0) = \alpha\ ,\ u(1) = \beta \end{cases}$$

Here is the Matlab code for it:

```
%Finite Element Method to Laplace Equation.
clc
clear
a=0
b=1
N=40
h=(b-a)/(N+1);
syms t s
f(t)=pi^2*sin(pi*t);
alpha=0;
beta=0;
S(t)=sin(pi*t);
A=zeros(N,N);
for i=2:N-1
    A(i,i)=2;
    A(i-1,i)=-1;
    A(i+1,i)=-1;
end
A(N,N)=2;
A(1,1)=2;
A(2,1)=-1;
A(N-1,N)=-1;
for i=1:N
    x(i)=a+i*h;
end
B(1)=vpa(int(f(s)*s,a,x(1))+int(f(s)*(-s+h*2),x(1),x(2))+alpha);
for i=2:N-1
B(i)=vpa(int(f(s)*(s-h*(i-1)),x(i-1),x(i))+int(f(s)*(-s+h*(i+1)),x(i),x(i+1)));
end
B(N)=vpa(int(f(s)*(s-h*(N-1)),x(N-1),x(N))+int(f(s)*(-s+h*(N+1)),x(N),b)+beta);
u=A^-1*B';
X(1)=a;
```

```
U(1)=alpha;
for i=2:N+1
X(i)=x(i-1);
U(i)=u(i-1);
end
X(N+2)=b;
U(N+2)=beta;
for i=1:N+2
E(i)=abs(S(X(i))-U(i));
end
subplot(2,1,1)
plot(X,U,'*')
hold on
fplot(S,[a b])
subplot(2,1,2)
plot(X,E)
```

Finally:

Finite Volume Method to solve the problem:

Example 3.4.

$$\begin{cases} -u''(x) = \pi^2 \sin(\pi x); \quad x \in (0,1) \\ u(0) = 0 \ , \ u(1) = 0 \end{cases}$$

Here is the Matlab code for it:

```
%Finite Volume Method to Laplace Equation.
clc
clear
N=50;
a=0;
b=1;
h=(b-a)/(N+1);
syms t
f(t)= pi^2*sin(pi*t);
A=zeros(N,N);
for i=2:N-1
```

```
A(i,i)=2/h;
A(i-1,i)=-1/h;
A(i+1,i)=-1/h;
end
A(N,N)=2/h;
A(1,1)=2/h;
A(2,1)=-1/h;
A(N-1,N)=-1/h;
for i=1:N
x(i)=a+i*h;
B(i)=h*f(x(i));
end
U=A^-1*B';
ss(t)=sin(pi*t);
for i=1:N
E(i)=abs(ss(x(i))-U(i));
end
subplot(2,1,1)
plot(x,U,'+m')
hold on
fplot(ss,[a b])
subplot(2,1,2)
plot(x,E,'+')
```

3.2 The solutions

In this section, we will present the results obtained for each of the problems programmed in this chapter.

3.2.1 For the First - Order Partial Differential Equations

The following image shows the approximate solution to the previously studied problem using Euler Explicite Method.

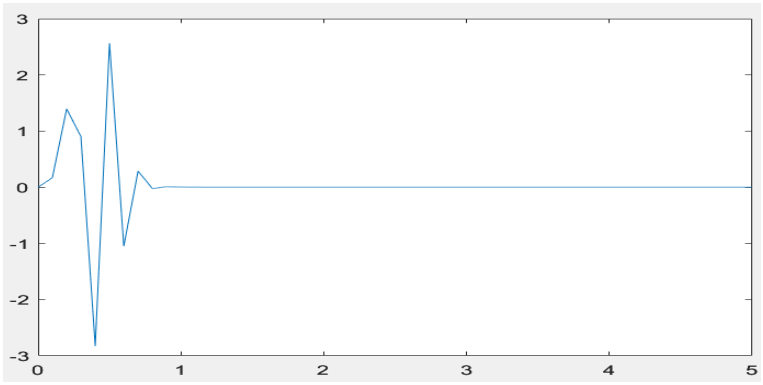


Figure 3.1: Approximate solution of Euler Explicite Method.

3.2.2 For the Second - Order Partial Differential Equations

The following pictures show solutions for each of the methods studied in Chapter Tow . This solution is specific to An Explicit scheme studied with Finite Differences.

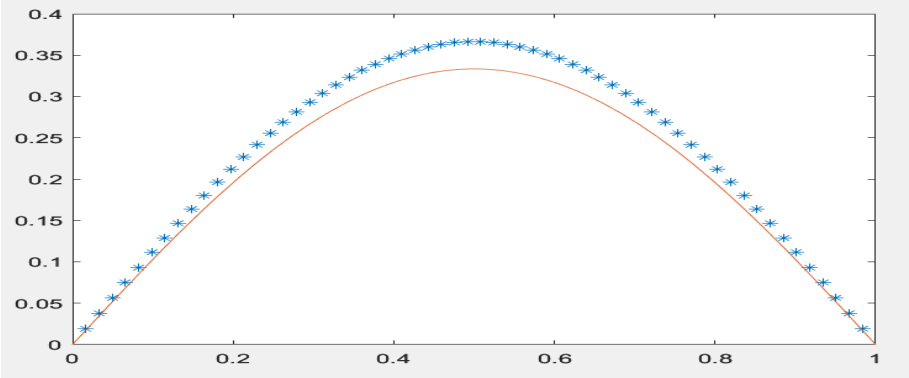


Figure 3.2: Exact, Approximate solutions of Finite Difference Method.

The following pictures show the solution to the same problem,but in three different methods.

Finite Difference Method:

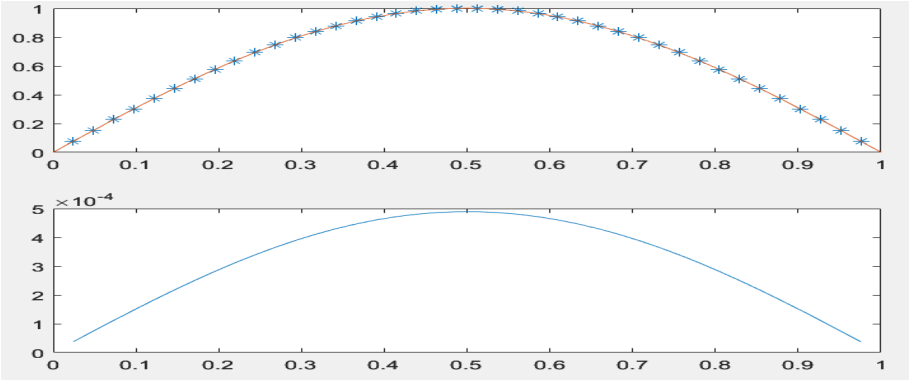


Figure 3.3: Exact, Approximate solutions and the error of Finite Difference Method.

Finite Element Method:

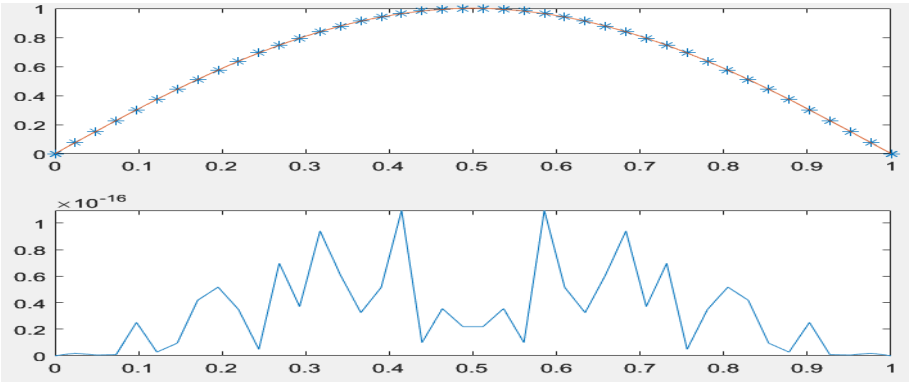


Figure 3.4: Exact, Approximate solutions and the error of Finite Element Method.

Finite Volume Method:

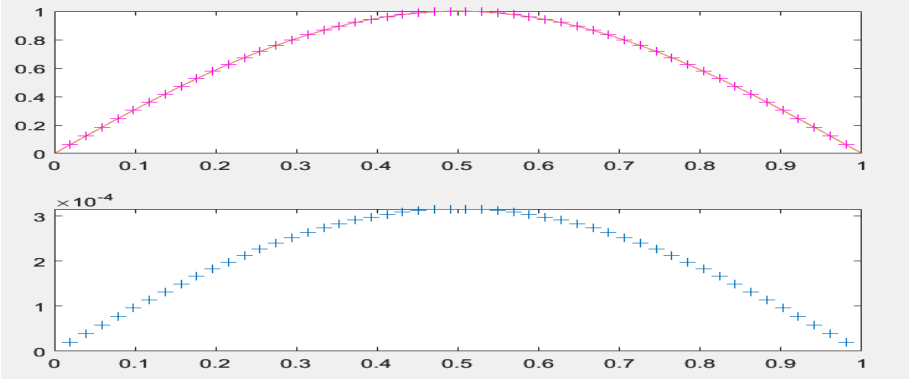


Figure 3.5: Exact, Approximate solutions and The Error of Finite Volume Method.

Notation 3.1. *We note the last problem solved in three different ways and each gave a different result than the other.*

Conclusion

In conclusion, **MATLAB** provides a powerful and flexible environment for solving partial differential equations **(PDEs)** numerically.With its built-in **PDE** solvers, such as pdepe, parabolic, and elliptic, researchers and engineers can efficiently solve a wide range of **PDEs** arising in various fields, including physics, engineering, and finance. **MATLAB's PDE** solvers offer several advantages, such as user-friendly problem setup, adaptive mesh refinement, and support for complex geometries and boundary conditions.

Additionally, **MATLAB's** powerful visualization tools enable users to effectively analyze and interpret the numerical solutions, aiding in the understanding of the underlying physical phenomena.

However, it is important to note that the choice of the appropriate numerical method and the proper implementation of boundary conditions and initial conditions are crucial for obtaining accurate and reliable solutions.

Users should have a solid understanding of the mathematical formulation of the problem and the numerical methods employed by **MATLAB's PDE** solvers.

Furthermore, **MATLAB**'s extensibility through user-defined functions and toolboxes allows for the development of customized **PDE** solvers tailored to specific applications or research needs.

This flexibility makes **MATLAB** an attractive choice for researchers and practitioners working on cutting-edge problems in the field of **PDEs**.

Overall, **MATLAB's** robust **PDE** solving capabilities, combined with its user-friendly interface and powerful visualization tools, make it an excellent choice for researchers, engineers, and students working on problems involving partial differential equations across various domains.

Bibliography

[1] M. Beggas, Méthode de différences finies pour les équations aux dérivées partielles.

[2] D. M. Causon & C. G. Mingham, Introductory finite difference methods for PDEs, Bookboon, (2010).

[3] Y. T. Chen, Computational Partial Differential Equations Using MATLAB®. CRC Press, (2019).

[4] D. Choı, Méthode des éléments-finis par l'exemple, Université de Caen, Bld Maréchal Juin, 14032, 7-8.

[5] M. P. Coleman, An introduction to partial differential equations with MATLAB, CRC Press, (2016).

[6] D. Christian, METHODES D'APPROXIMATION DES EQUATIONS AUX DERIVEES PARTIELLES, Université de Cergy-Pontoise, Cours Master Madocs M2, (2009/2010).

[7] N. Doudi, Méthodes numériques pour EDO et EDP, Cours pour 3ième année licence math, université d'E-oued, (2021/2022).

[8] J. F. Epperson, An introduction to numerical methods and analysis, John Wiley & Sons. (2013).

[9] S. L. Eshkabilov, Practical MATLAB Modeling with Simulink: Programming and Simulating Ordinary and Partial Differential Equations. Apress, (2020).

[10] D. Houcque, Introduction to Matlab for engineering students, Northwestern University, 1. (2005).

[11] S. Kim, Numerical Methods for Partial Differential Equations. Department of Mathematics and Statistics, Mississippi University, USA, (2017).

[12] C. Lopez, MATLAB programming for numerical analysis, Apress, (2014).

[13] S. Matlab, Matlab. The MathWorks, Natick, MA, 9, (2012).

[14] S. R. Otto & J. P. Denier,An introduction to programming and numerical methods in MATLAB, Springer Science & Business Media, (2005).

[15] Y. Pinchover & J. Rubinstein, An introduction to partial differential equations, (Vol. 10). Cambridge university press, (2005).

[16] J. F. Scheid, Volumes Finis, (2017).

[17] R. L. Spencer & M. Ware, Introduction to MATLAB. Brigham Young University, available online, accessed, 7, (2008).

[18] S. M. Toolbox, Matlab. Mathworks Inc, 20, (1993).

Résumé

Dans cette thèse, nous nous intéressons à la façon de résoudre des équations aux dérivées partielles dans MATLAB. Nous nous sommes d'abord présentés à MATLAB et à ses propriétés. Ensuite, nous avons étudié les équations aux dérivées partielles d'ordre 1 et 2 et les avons résolues avec trois méthodes numériques : différences finies, éléments finis et volumes finis. Finalement, nous avons trouvé quelques solutions Pour les équations étudiées dans MATLAB.

Mots clés: EPDs,Différences Finies,éléments Finis,Volumes Finis et MATLAB.

ملخص

في هذه الأطروحة نحن مهتمون بكيفية حل المعادلات التفاضلية الجزئية بالماتلاب. حيث قمنا أولا بالتعريف على الماتلاب وخصائصه.من ثم تطرقنا في دراسة المعادلات التفاضلية الجزئية من الرتبة 1و2 وحلها بطرق العددية الثلاثة: الفروق المنتهية,العناصر المنتهية والحجوم المنتهية. والأخير أنجزنا بعض الحلول للمعادلات التي تمت دراستها بالماتلاب.

الكلمات المفتاحية: المعادلات التفاضلية الجزئية,الفروق المنتهية,العناصرالمنتهية,الحجوم المنتهية والماتلاب.

Abstract

In this thesis, we are interested in how to solve partial differential equations using MATLAB. We first introduced ourselves to MATLAB and its properties.Then, we studied partial differential equations of order 1 and 2 and solved them with three numerical methods: finite differences,finite elements and finite volumes. Finally, we achieved some solutions For the equations studied in MATLAB.

Keyword: PDEs, Finite Differences, Finite Elements, Finite Volumes and MATLAB.