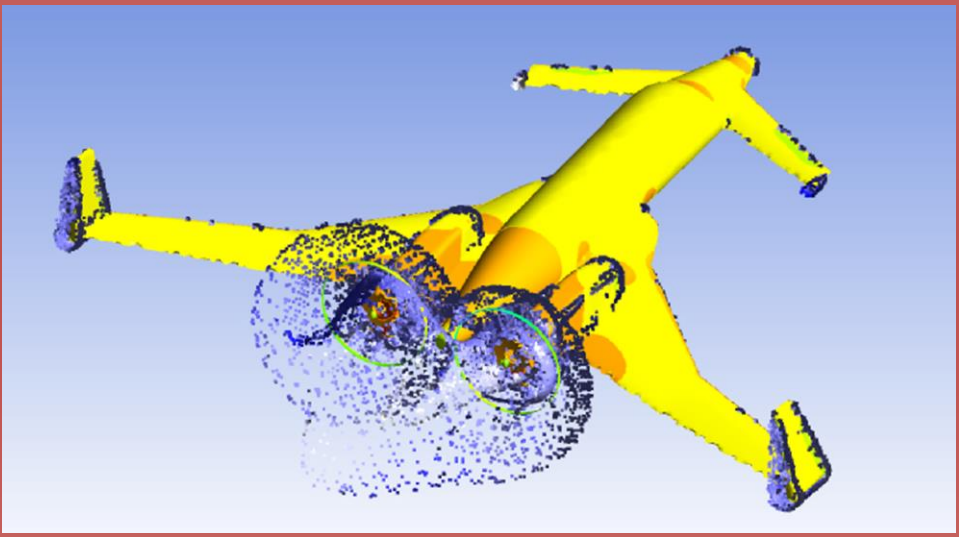
Coding Tutorials for Computational Fluid Dynamics

First Edition



Ahmed Al Makky

Table of Contents

List of Figures

Preface

The tutorials will be focusing on improving the skills of students working in the field of CFD. They will be split for beginners and intermediate students. The main goal of these tutorials is to be as a step by step guide to building coding skill. This is achieved by providing examples which are related to the application side of things in the field. The user will have to work on both FORTRAN, MATLAB and Excel. Based on the fact that during your work you need to validate your generated data, that can be accomplished through the visualization process. What you will find during the research process that large sums of data need to be analysed and the inspection process of cell by cell is tedious and not practical. This is why Excel, MATLAB and Tecplot come in hand. This Book is to be used with other references in parallel. After each chapter a recommended list of books to the student is presented. The book is intended to be as a quick reference to use during projects. Each chapter is independent from the other. Its most probably that the material will be made up of several volumes. This material is intended to pass all what I have learnt during my PhD to the next generation of students and researchers. To make them better and happier. The material is presented for scientists and engineers. I would be happy to get any feedback to improve the material. I will be regularly updating the material in this book (http://cfd2012.com/). How much the topics discussed seem to be easy with progress of time you will find that due to use of several methods the probability of a problem to occur increases, especially that we humans don’t work at a constant efficiency all the time.

Ahmed Al Makky

Coventry 25/01/2014

Dedication

Thank god for what I have been given in life. Thank god for the wonderful parents I have been given. They have supported me throughout the many ups and downs of life. To them I dedicate this book. I am grateful to Professor Peter Carpenter because if I hadn't read his book on Aerodynamics I wouldn't have found my way to the field of Computational Fluid Dynamics. I owe my gratitude to Michal Kopera for his valuable advice and patience on MATLAB. Praying that this book will be helpful to the readers and wishing them to become the best researchers in the field good luck.

Abstract

This section will be written.

List of Abbreviations and Symbols

Abbreviations

BC	Boundary Condition
CFD	Computational Fluid Dynamics
DNS	Direct Numerical Simulation
LES	Large Eddie Simulation
RANS	Reynolds-Averaged Navier-Stokes

Roman Symbols

A	Constant

Greek Symbols

α	Thermal diffusivity

Superscripts and Subscripts

$\tilde{a}$	Filtering operation
$\bar{a}$	Average operation

Chapter 1

What is Computational Fluid Dynamics?

1-1 What is CFD?

Computational Fluid Dynamics can be defined as the field that uses computer resources to simulate flow related problems. To simulate a flow problem you have to use mathematical physical and programming tools to solve the problem then data is generated and analysed. This field has been developing for the past 30 years. During the last 10 years this field has just made a big jump especially due to the introduction of new computer hardware and software. University academics write their own mini sized codes to solve the problems they are usually assigned. These codes are usually not that user friendly and not very well documented. This resulted in the emergence of commercial software packages that are user-friendly through the use of interface tools. Companies that market these packages provide user manuals with step by step examples and offer training programs. Before Computational Fluid Dynamics was used on a wide scale. Engineers had to make small scale models to validate how slender is their design. A design that is not aerodynamically efficient will have a negative impact on where the design will be used. So a bad car design will result in more fuel consumption. While a bad design for a plane might cause it to crash.

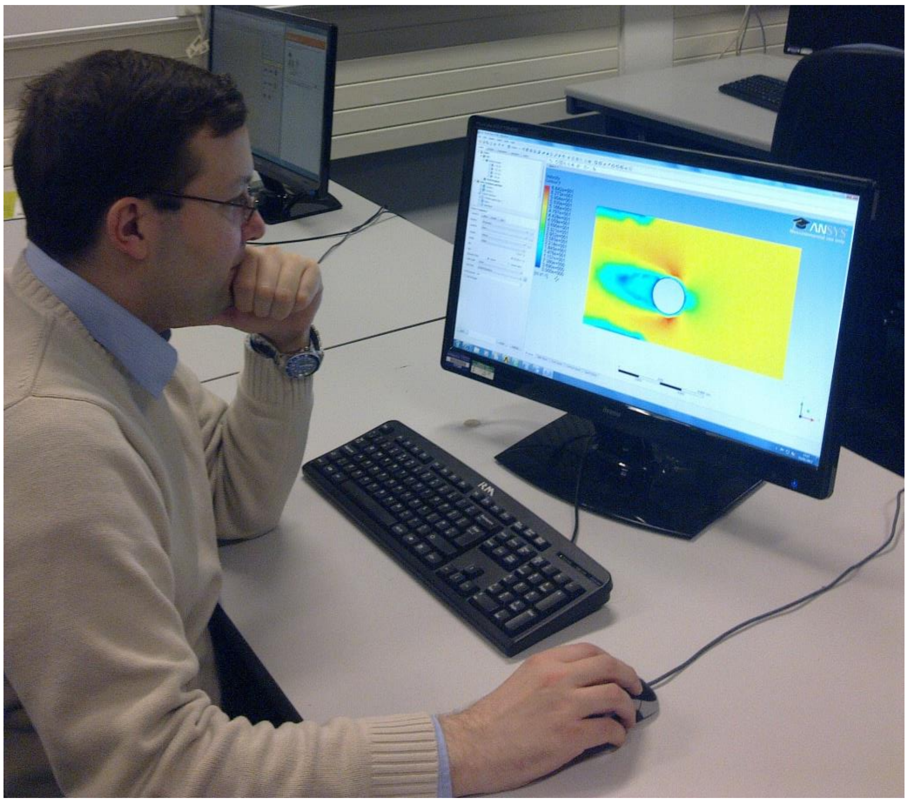


Figure 1 A researcher using ANSYS flow simulation software in the university computer lab.

Figure 2 : A researcher puts a lorry model inside a wind tunnel for tests.

These experiments where done by immersing the model in a transparent fluid/gas and then applying flow conditions similar to the conditions the model would encounter. Next came was applying a colour dye to the flow which resulted in showing the flow pattern resulting from the flow. The set back of using such methods is that after you have got your blue print drawings from the design team you have to model your design in a work shop. Once the model has been made you will need to have the experimental set up apparatus in an allocated space in the engineering lab. Just to note that Wind tunnels are huge in size some of them can contain are real scale model as illustrated on figure 3.While now you just need the flow modelling software , a good research team and lots of computer resources. The researchers advantage of CFD is that you can have a wind tunnel on your desktop without having to worry about choosing big fans or that your model will fit into the wind tunnel or not .Figure 4 shows the size extent of a wind tunnel blowing fan. These problems can be solved in the software by changing the scaling of the

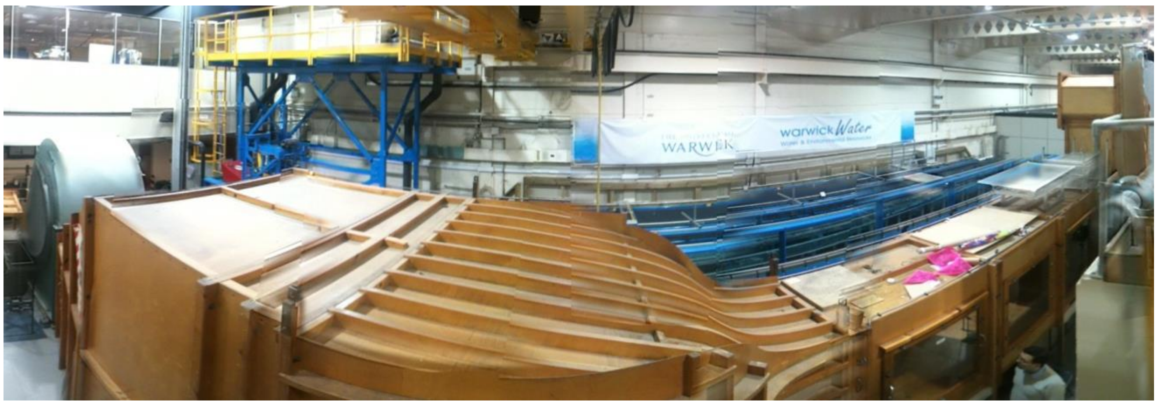


Figure 3: A panoramic view of the wind tunnel in the workshops, just to illustrate its tremendous size in comparison with a desktop computer. The researcher can be seen on the bottom right hand side of the picture.

Studied model or by modifying the value of the inflow velocity into the studied domain mesh.

Figure 4 : The Inlet duct of the blowing fan for the wind tunnel its reasonably large.

1-2 Examples of Results Produced by CFX Flow Simulation Software:

The Total Velocity is shown in the figure 5, this studied case was to check that the software can handle the produced mesh, the inflow speed was chosen to be low just to see if the combustor model achieves its modelled objectives. One of the objectives is the high speed in the core region to provide a swirl flow condition in the core region to achieve proper mixing of the fuel air mixture.

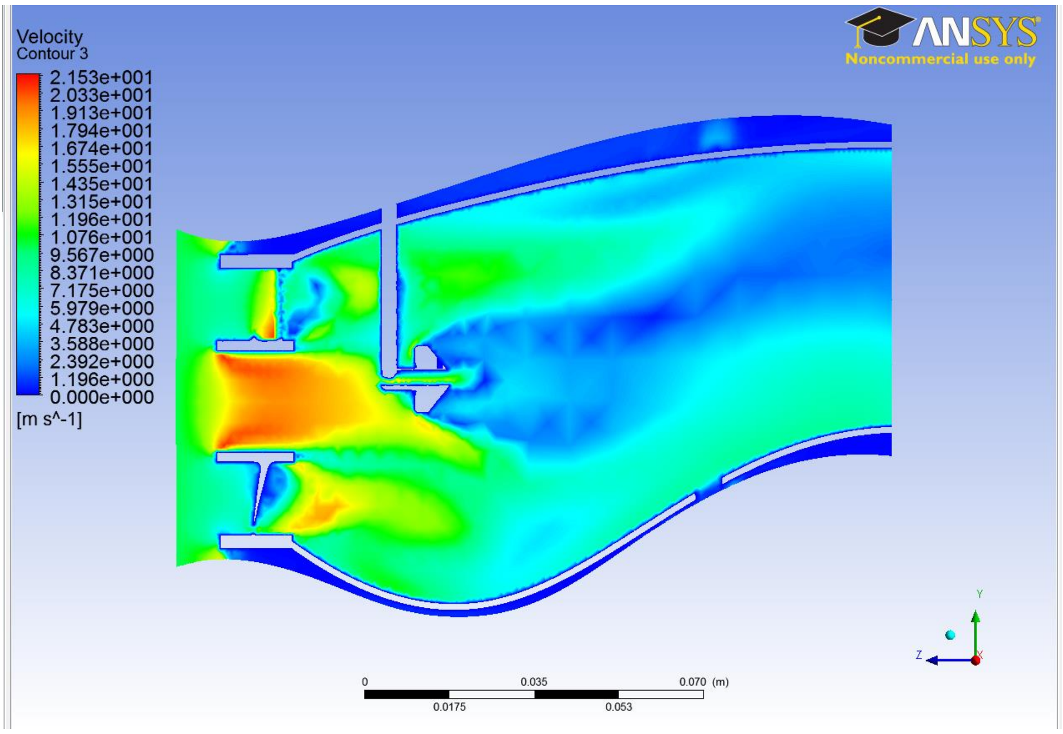


Figure 5: This expresses the total velocity in a gas turbine combustion chamber.

The simulation illustrates the design profile for where and where not should there high levels of shear stress. As can be visible on figure 6 the dark blue colour shows that the slender design is performing in an efficient way. There is some shear occurring on the guiding vanes which give the designer a hint that a fillet is needed to some edge disturbing the flow. In a real life situation it would be very difficult to test a gas turbine combustor due to the complexity of the test facility required not forgetting the required expertise and specialized personal and running costs and ...etc. The next phase for such a test case is to run a reactive case with combustion occurring and seeing when dose flame blow out happen. Also another feasible option is seeing whether the surface of the combustor needs more convective flow to its surface to cool it down due to the excesses heats its exposed to. The reaction rate constants are provided in a library with the software, so as a user what is required is just seeing what kind of fuel is used and what kind of species am i looking to study. An additional option is also adding a fuel spray to the studied case where droplet size and a spray cone can be specified. This can provide a good perception if the provided length in the combustor and swirl is substantial for droplet evaporation before the mass flow moves to the turbine section.

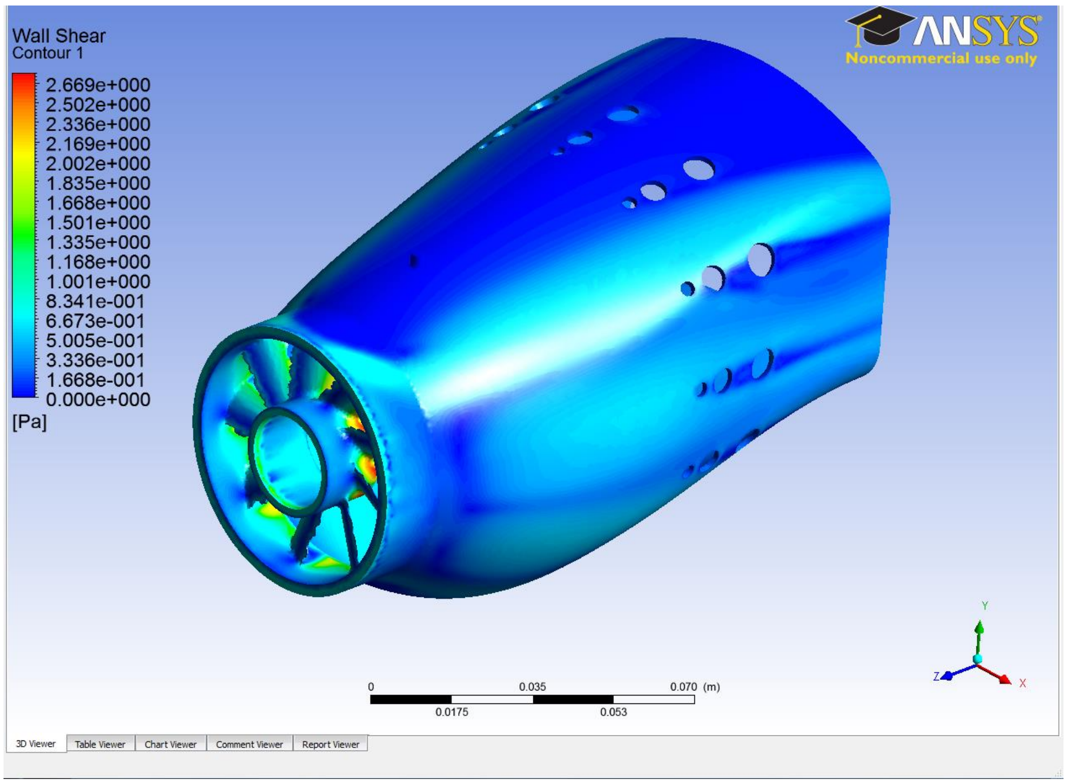


Figure 6: The dark blue colour shows that the slender design is performing in an efficient way.

Next comes figure 7 which shows the flow pattern inside the heat exchanger, where the first set of three tubes get a high portion of the heat which they have to extract (10 deg), then comes the next set of three which have to extract (6 deg), then the next set of three (4 deg) and finally the last set of three will extract (2 deg). Having a look at the simulation flow pattern proposes some design modifications. As we know these days heat exchangers are used everywhere and the condition for them to being compact light and efficient is some any manufacturer looks for. If any of the requirements is not achieved you can quickly remodel the design and test it on the software while in a real life situation you would need to go to the design team to have a new set of blue prints then the calculations are conduced then comes manufacturing your design in the work shop and finally comes the testing phase. All this is time consuming and competition between companies has led lots of companies to by these kinds of software due to their sever impact on cost reduction. The other positive point that due to the provided libraries that come with the software you can test the performance of the heat exchanger with different kinds of fluids such as refrigerants to combustion products. The libraries provided include the thermodynamic properties of the selected fluid from viscosity, kinematic viscosity, thermal conductivity, specific heat capacity at constant pressure.. etc. If for any reason the fluid is not in the provided libraries the software is provided with interface windows which allow you to input the formulas describing the studied thermodynamic property. It is obvious from figure 8 that the convective heat transfer is dominant on the first half of the heat exchange. One of the major reasons for this is due to the use of a separation plate that splits the heat exchanger into two halves. This is shown through the use of a volume rendering option of the flow velocity. One of the visible design modifications is in applying a dome cap to the bottom section instead of having a straight wall opposing the flow flux. The direct impact of such a modification is decreasing the pumping power required which results in less consumed power therefore that means less bills at the end of each month. As we know financial factors always attract attention.

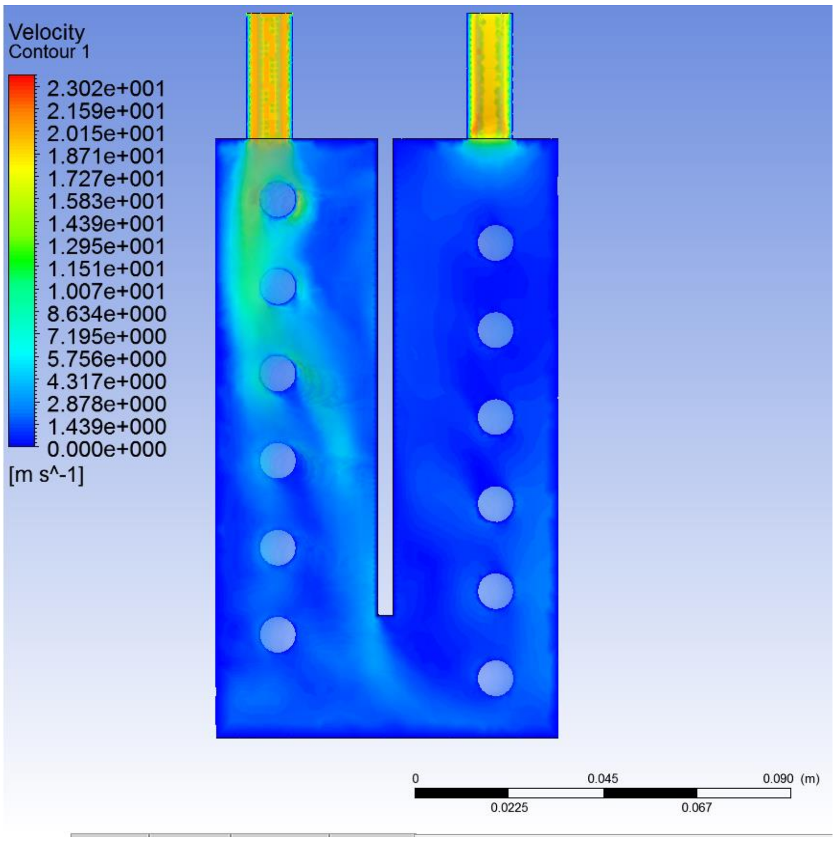


Figure 7: A cross sectional view of the flow pattern in the heat exchanger.

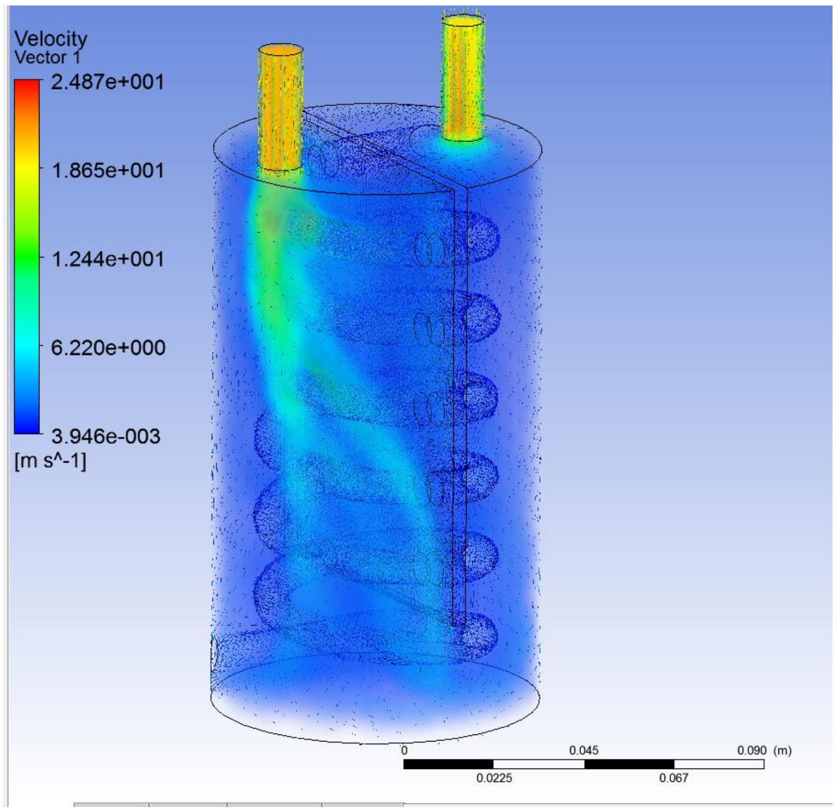


Figure 8: Using a volume rendering option in a CFD software gives the flow an effect as if the flow has a dye in it to visualize it.

1-3 Computational Fluid Dynamics Theoretical Foundation:

1-3-1 Linear Algebra, example TDMA, Gauss ,.....etc

For a researcher working in CFD linear algebra is a must and has to be covered or refreshed , because with progress with time you will find that all your work revolves around matrix operations, plotting vector fields , vector operations, etc. I would recommend this book for an introduction to the subject , its full of solved examples which are then linked to real life applications.

1-3-2 Numerical Analysis, example to know how to dicretetize first order, second order derivatives encountered in PDEs (Partial Differential Equations) and ODEs (Ordinary Differential Equations)

1-3-3 Tensor Calculus, example for applying the summation convention when tensor notation is used.

1-3-4 Fluid Mechanics, example to know how to use the Continuity Equation, Bernoulli Equation, Reynolds Number, Mach Number,….. etc

1-3-5 Programming:

Throughout your research you come across all sorts of codes, during this process you will need to quickly know how to approach the codes and to what details to look for and best ways to summaries the processes that take place in it. For an introduction to computer science field i would recommend this book:

1-3-5-1 Programming in General:

To know how to write a code through a step by step guide ,i would recommend this book it's in Pascal but you can use its outline depending on what language your using, it's a 1995 edition but it's very useful for self-study.

1-3-5-2 Fortran90.

You might be assigned during your PhD to generate data with a provided code, during the process of running the code you will sometimes have to go through the various parts of the code.

1-3-5-3 MATLAB.

MATLAB is a very user friendly software with lots of helpful online material and books not forgetting its excellent help. As a researcher you will have to learn this software due to its powerful tools in the data analysis part of your PhD.

1-3-6 Statistics and Probability.

If you're going to be working in the field of turbulence you will need statistical tools, while if you're going to work on reactive flows you will need initially statistical tools and later probability functions for later stages of your project once you get to the detailed side of chemistry.

1-3-7 Differential Equations and Computational PDEs.

1-3-8 Algorithm Theory.

1-3-9 Data Structures.

During the research process dealing with all sorts of matrices of different types is an issue that has to be accepted, depending on the studied problem.

1-3-10-Discrete Mathematics.

1-4 Steps to Set Up a Simulation

During the description of the following points might overlap every now and then. Planning for a simulation is advantageous it helps you to focus for longer hours keeps you focused on your target and confident that you will get to finish on target.

1- Choosing the problem like as case study or in most cases you are assigned a problem which might represent a air flow in a lecture room or air flow around a car, air flow around a building ...etc.

2- What parameters are of interest. That's if some hints had already been provided by the manufacturer or by the customer. If the researcher has been assigned the problem and has to start from scratch this leads him to run simulations for very simple cases , then to verify that the results if they are correct by comparing the data with experimental or available literature data. Once this objective is achieved the researcher would select the required parameters for study makes a list of them and then starts running more detailed simulations . What is meant if we are interested in a combustion case turbulence intensity is an attractive parameter to study which means it can illustrate the magnitude of mixing strength in the process. Another parameter to study and can give a good indication of the rate of mixing is through plotting the eddy viscosity frequency. Shear strain rate can give a good indication on flow break up leading to dissipation such an example is a jet blowing into a stagnant air in a room.

3-The user has to specify if it's an Internal flow case (looking at figure 9 which represents a throttle valve) or an external flow case (looking at figure10 which refers to a wind turbine).

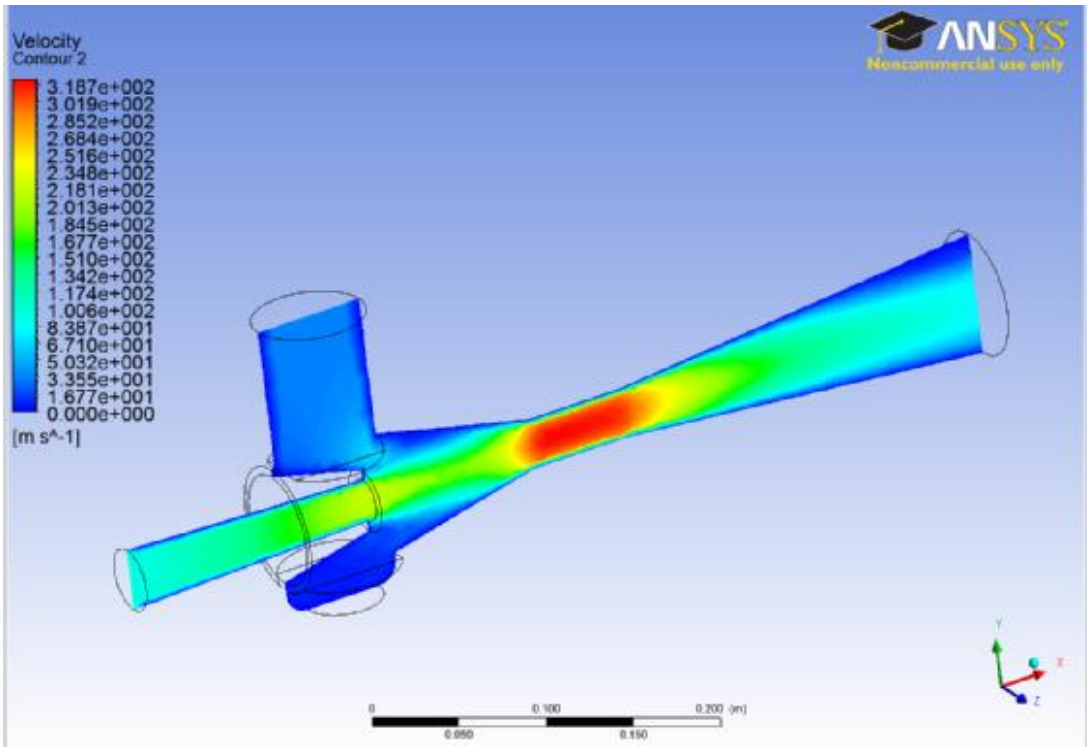


Figure 9: A throttle valve describes and internal flow example, where maximum velocity is achieved at the minimum area cross section (throat).

In some cases the researcher might have to break down the studied mesh into smaller domains due to that the available hardware resources are not sufficient for such a simulation unfortunately this is something that the user has to get used to as an obstacle that would occur every now and then. Options are provided with CFD software's in domain decomposition as an option which can be very helpful especially in study cases that have large meshes.

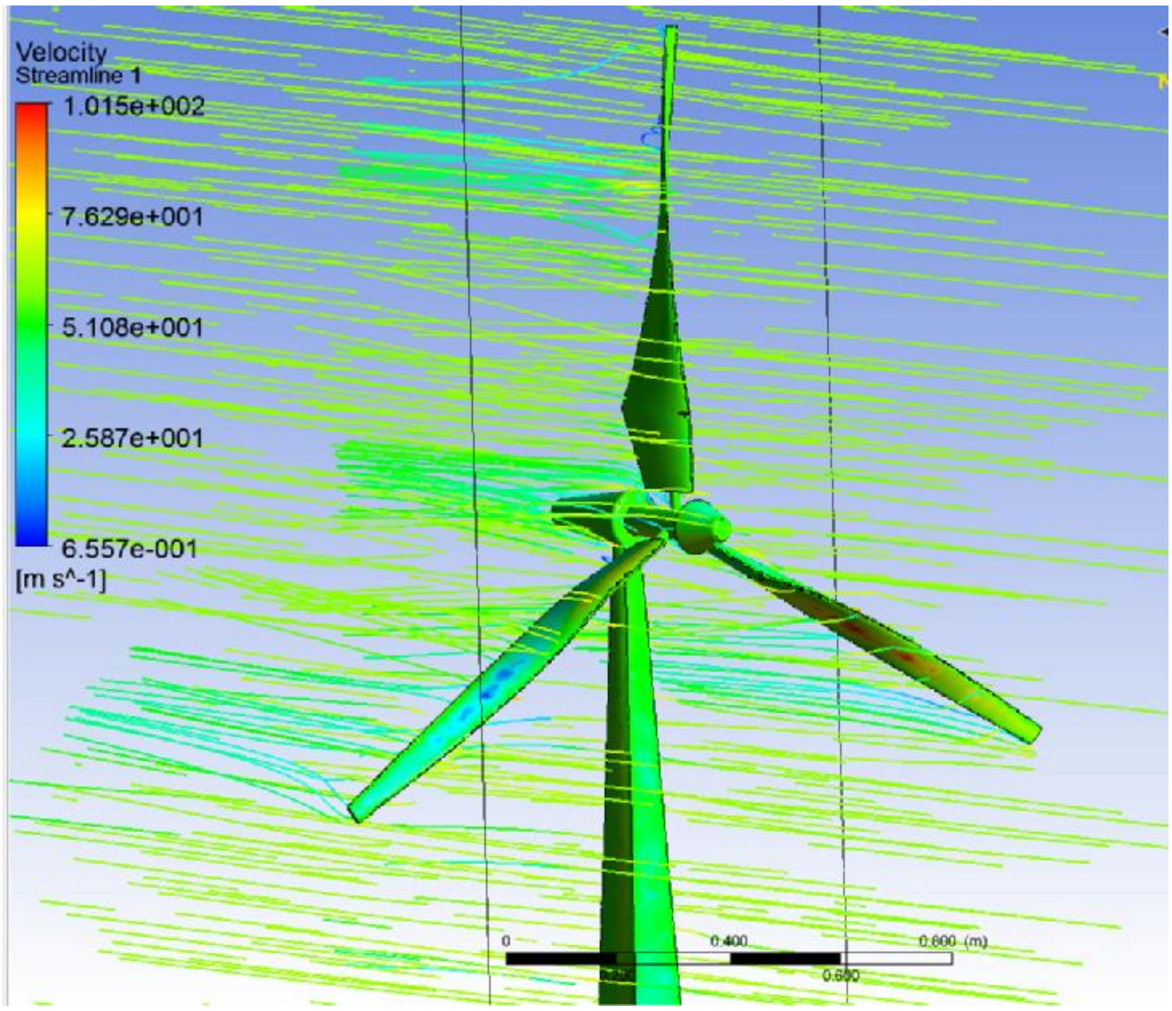


Figure 10: A wind turbine describes an external flow case.

4- Choosing a simulation model. These models all depend on what is wanted out of the simulation and what is being looked for. These models start from Direct Numerical Simulations to Reynolds Averaged Methods to .....etc.

5- Assessing the required hardware and software resources and what is available. Due to these challenges the user has to take into account these issues as an example simulation run time, calculation time on the desktop, data storage and simulation grid resolution, how many cores are required ....etc.

6- Specifing the boundary conditions in the simulation. These can refer to solid walls or inflow and outflow boundaries. At some cases moving boundaries are taken into account as cases in simulating flows in pumps, compressors ,wind turbines ....etc. At certain cases the smoothness of the boundaries have to be specified as smooth or rough boundaries.

7- Selecting the physical continuum which might be Air, Argon, Nitrogen, ...etc.

8- Selecting a Mesh type for the simulation. Will it be hexahedral or tridiagonal ...etc. Usually CFD software have several meshing options to choose from.

1-5 Mathematical Flow Modelling

On the assumption that the problem is a flow problem:

1- Choosing the coordinates system, which specifies what space the student, will work on. Metric ,Soblove or Hillbert space..etc. At later stages this will lead the researcher to adopt specified mathematical methods that can only be used in these spaces.

2- Choosing the reference coordinate system, this is automatically taken at point (0,0,0) . The equations that you are going to be solved are related to the coordinate system for the space relation. Once motion is going to be studied that means we will have a generated mesh velocity and acceleration to be taken in account. The use of velocity acceleration leads to the conclusion that derivatives are going to be used. Once derivatives come into account that means at later stages we will need to use some kind of discretization method.

3-The studied domain geometry and dimensions. Is it a box , rectangular , sphere or cylinder or a complex domain?. This relates to the required scales to be captured by the simulation. Another factor is the studied object. As an example the Navier-Stokes equations are applicable at certain length ranges at small scales Lattice-Boltzman might be better at a smaller scale schrodinger equation that all depends on using the Knudsen number to verify the scale of the studied case.

4- Then comes how many species will there be? in most case air is taken as the studied fluid. That will ease the studied case. While if several species are taken into account will complicate the study more. Several species will lead us to two cases either a reactive or a non-reactive case.

5- Compressible or non-compressible fluid case. Each case has a different approach to solve the major difference is in how to solve the pressure problem. In compressible flows you just use the ideal gas equation while in incompressible flows you will need to use a more complex approach for it.

6-Chossing the physical governing laws of importance. Will it be a Conservation of momentum or Conservation of mass or Conservation of energy or all at the same time?

7- Choosing thermodynamic reference parameters temperature, viscosity, pressure, this is essential, because at certain occasions due to that some parameters change in relation to other parameters, as an example viscosity is related to temperature using the Sutherland formula.

8- Now we are ready to choose a Numerical Model. A quick survey on the available models their pros and cons. Usually you can find a researcher who had already done this in a recently published paper if your studied case is not that recent then look for a book that has summarised the models. At really advanced stages you might have to make your own model which is very hard not forgetting it’s not easy to convince people to use your model except if it's something really outstanding.

9- None dimensionlizing process of the used equations where required, this always gives the written code a more dynamic characteristic to use the code on a wider spectrum of problems.

1-6 Equations Used (Numerical Method)

All the equations used in the study of flows are derived from Reynolds Transport Theory, it is recommended to memories this equation and from it derive all the forms you want to study. From it you can derive the continuity equation, the momentum equations, the species equation and finally the energy equation (in the case of working with compressible gases new quantities will be added to the study such as the adiabatic equation). You can add as many quantities to your studied case by using the Reynolds Transport Theory keeping in mind the computer resource that are available for the researcher.

The Navier-Stokes Equations are used, that’s in general term, in detail the researcher can think of these equations as representing the motion of a particle in space. Using CFD the splitting of a predefined space into small cells and using the Navier-Stokes Equations for each studied cell. People usual feel scared from the term equations and once derivatives and lots of them are used that quickly lead to a quick decision i can’t understand it i won’t read it. Actually these set of equations represent the change of a vector quantity in 3D space, the vector quantity in the Stokes equation is the velocity vector. The Stokes equations represent the transfer of momentum in a studied flow.

$$\int \frac{\partial \rho \emptyset}{\partial t} dtdV + \int div(\rho \emptyset V) dtdV = \int Ddiv(grad\emptyset) dtdV + \int S_{\emptyset} dtdV \tag{1.1}$$

Simplifying the equation to a form that is related to a regular uniform mesh gives

$$\frac{\partial \rho \emptyset}{\partial t} + div(\rho \emptyset V) = Ddiv(grad\emptyset) + S_{\emptyset} \tag{1.2}$$

Applying $\emptyset = 1$ would lead to the continuity equation

$$\frac{\partial \rho}{\partial t} + div(\rho V) = S \tag{1.3}$$

Once you write down the set of equations you will be using for your study, the next step is to see what forces are wanted to be taken into account and what factors will be neglected in the study. This will lead the researcher to the available models for his study which starts with DNS (Direct Numerical Analysis and finishes with simple models) the selection of the model depends on the studied case from one perspective and available recourses from another perspective.

so for a case of studying the transport of the scalar quantity of velocity in the y axis direction we would assign the value $\emptyset = v$ which results in the following formula:

$$\frac{\partial \rho v}{\partial t} + div(\rho v V) = Ddiv(gradv) + S_v \tag{1.4}$$

while for a case of studying the transport of the scalar quantity of temperature in the studied domain we would assign the value $\emptyset = T$ which results in the following formula:

$$\frac{\partial \rho T}{\partial t} + div(\rho T V) = D_T div(gradT) + S_T \tag{1.5}$$

so for a case of studying the transport of the scalar quantity of species in the studied domain we would assign the value $\emptyset = Y_n$ which is called a volume fraction which results in the following formula.

$$\frac{\partial \rho Y_n}{\partial t} + div(\rho Y_n V) = D_n div(grad Y_n) + S_n \tag{1.6}$$

1-7 A CFD code consist of the following parts:

Just as a way to make things clear for the reader you can think of a CFD parts as a Human body where input parameters are food to be eaten, the time discretization section is the humans head, the space discretization section is the human body. While the computer compiler represents the heart.

1- Input Parameters:

This section holds the number of grid points in the domain, domain dimensions, initialization profile selection, and simulation time length. As a researcher you would be working most of the time on the input file changing different variables and then running a simulation. At more advanced stages of PhD you would have to write your own subroutines and add them to the code.

2- Time Discretization Section:

The two mostly liked methods that are used in CFD, are the Runge-Kutta and Crank Nicholson Method. If the code uses the Runge-Kutta method the time stepping section forms a mostly independent section of the code while using the crank Nicholson Method you will find there is a more strong coupling between the time and space discretization parts of the code. Codes usually have criteria to run a simulation where after each calculated time step the code checks that the simulation stays stable. In such cases there are some formulas that are used to satisfy the condition in having the best discrete time to achieve a stable simulation.

3- Space Discretization Section:

Space discretization consist of the next four parts, these four parts are essential and cannot be neglect, because each one is related to the other and has a direct impact to the other**.**

a- Grid Generation.

A grid has to be generated and depending to the type of fluid it will be chosen. as an example the use of staggered grid in incompressible gases. Some people would say why the grid is so important. Once you start the analysis process you will need to see the values of the studied scalar at the generated points. This is done through plotting the vector fields or plotting contour plot ....etc. Some time you will need to see the value of velocity at a certain part of the domain. In such a case you will just give the coordinates of the point to extract its value

b- Initialization Profiles.

You can run a simulation without having an initialization profile but its side effects that you will need several thousand time steps till the flow develops into its predicted form depending on the simulation stop criteria. Explicit or implicit methods are also related to initialization for space discretization and time.

c- Boundary Conditions.

During a fluid flow simulation you always need to assign a certain surface some inflow parameters and another one at least to be outflow surface. You can specify that be assigning a source term a value of flow velocity .The choice of periodic condition or none periodic depends on your requirements. Periodic satisfies a condition where at each time step of the simulation the amount of flow into the domain is the same as the amount of going out of the domain. Choosing surface roughness is also a requirement.

d- Discretization Methods.

There are various discretization schemes they start with very simple once such as central differencing scheme which requires two points and can end with schemes that use more than 15 grid points.

4- Output Section:

This is the section usually the researcher is assigned to. The researcher has to read the output data into a visualization software such as Tecplot, MATLAB, Excel....etc. Usually the code generates a set of output files these files have the calculated sets of data like the grid reference number its coordinates the value of the velocity at the point its temperature its pressure ...etc.

1-8 Skills for Building a Code:

1-To build a working numerical model that gives you data output which is not necessary correct is done using random number generators. This can also be accomplished using straight forward numbers such as 1.

2-Checking the physical units of the formula that they are homogenous.

2-Once you get the code working without bugs, next comes validation of data output. This is done by seeing experimental plots our published tables of the studied quantity.

1- 9 What's an Equation?

An equation is a mathematical expression that represents a region of points in a certain space. The researcher has to choose the range of study to specify his data matrix dimensions. Once the known's are specified for the studied case then comes writing the mathematical formula into a way the programming language interprets it to the computer. This is done by taking the formula apart depending on the number of variables and assigning them suitable names. Then comes specifying the number of points required. This is mostly done based on computer resources and how much is the required accuracy.

1-10 Recommended Reading List

Helpful References in the Field of CFD for the Finite Difference and Volume Approach: The classification is based on the level of the researcher**.**

For Beginners:

1-Anderson, D.A. 1995 Computational Fluid Dynamics, McGraw-Hill Higher Education.

2-Versteeg, H. Malassekra, W. 2007 An Introduction to Computational Fluid Dynamics: The Finite Volume Method, Prentice Hall.

3-Abbot, M.A & Basco, D.R. 1990 Computational; Fluid Dynamics An Introduction for Engineers, Longman.

4- Croft, D.R., Lilley, D.G & Stone, J.A.R. 1986 Heat Transfer Calculations Using Finite Difference Equations, Sheffield Hallam University Press.

For Intermediates:

1-Ferziger, J.H. Perci, M. 2001 Computational Methods for Fluid Dynamics, Springer.

2-Chung, T.J. 2002 Computational Fluid Dynamics, Cambridge University Press.

3- Date, A.W. 2009 Introduction to Computational Fluid Dynamics, Cambridge University Press

4-Roache, P.J. 1972 Computational Fluid Dynamics, Hermosa Publishers.

5- Hoffmann, K.A. Chiang, S.T. 1993 Computational Fluid Dynamics for Engineers 001,Engineering Education System.

6- Hoffmann, K.A. Chiang, S.T. 1993 Computational Fluid Dynamics for Engineers 002,Engineering Education System.

7- Pletcher, R. Tannehill, J.C. Anderson, D. 2012 Computational Fluid Mechanics and Heat Transfer ,Taylor & Francis.

For Professionals:

This is where the researcher’s part comes in finding the required journal.

Chapter 2

Introductory Examples

2-1 Introduction:

This section covers several important examples that programmers are exposed to on a regular basis while their writing a CFD code. After reading several examples the reader will start to see a certain trend that has been adopted for the coding examples. Where the code at its beginning has a memory clearing section, so that the user wouldn't have variables run from a previous code which have a similar name to the run code to give errors or contradicting results. Clearing the command window is beneficial. Working on a clear white command window is less disturbing to the eyes. What comes next is that several variables are assigned constants. We then proceed to a matrix construction section and the use of several built in functions of MATLAB. The final part then is the plotting part which is a visual aid to validate the output generated data. The main purpose of these examples is that in each example is to provide the researcher with new skills. The researcher has to put in mind that programming is an accumulation process of lots of skills hence you can't write a big code from the first month of coding.

2-2 Getting Started

The researcher can copy the codes directly from the notes and paste them directly into MATLAB. The examples assume that you already have some back ground knowledge on using MATLAB. These examples are for calculating the basic parameters of your code. The basic parameter like fluid density, viscosity, pressure, temperature, thermal conductivity ...etc. Once these examples are mastered and understood the researcher can proceed to much complex stuff.

2-3 Modelling Fluctuating Velocity

Modelling the turbulent velocity v in a one dimensional as a function of time. This is done by using the random number generator. Using the formula

```
t=(1:1:N)
```

Generates a one dimensional matrix that starts at 1 and finishes at N with a step of 1.This matrix represents the time matrix. This number generator produces some positive and negative numbers from a Gaussian distribution function. The for loop used here is used to store the generated data into a one dimensional matrix, note that the left hand side of :

```
v(i)=randn(1,1)
```

is the unknown while the right hand side represents the known . Then what follows is the plotting section of the code. You need to plot two axis, where the x axis represents the time axis while the y axis represents the velocity fluctuations.

```
clc
clear
N=100;
t=(1:1:N)
for i=1:N;
v(i)=randn(1,1)
end
plot(t,v)
title('Fluctuating Velocity')
xlabel('Time')
ylabel('Velocity v')
grid on
```

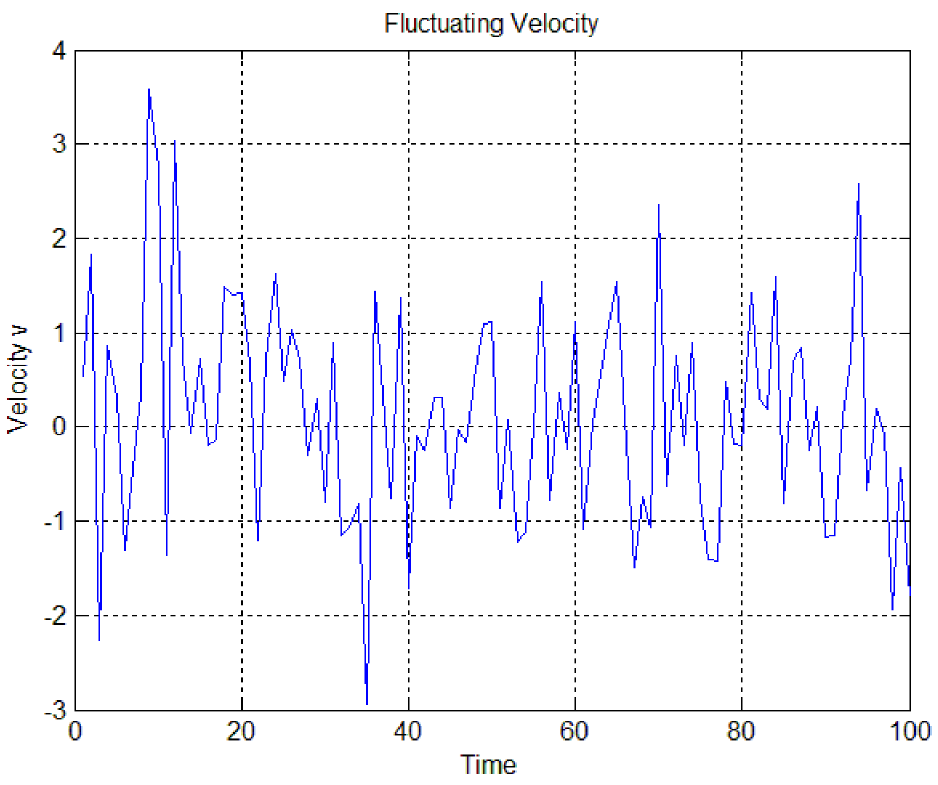


Figure 11 A plot showing velocity fluctuations against time, which is something you will always encounter in Turbulence.

2-4 Reynolds Decomposition of Velocity

While modelling turbulence the following tool is used where the turbulent velocity is decomposed into two components using the Reynolds decomposition. The decomposition of velocity looks like as the following:

$$u = \bar{u} + \grave{u} \tag{2.1}$$

This example is similar to the previous one with the addition of using the mean command the calculates the average value of the total generated velocity data. So as an additional task for the reader is to write an additional code to extract the fluctuating velocity matrix.

```
clc
clear
N=100;
t=(1:1:N);
for i=1:N;
v(i)=randn(1,1);
end
V(1:N)=mean(v)
plot(t,v,'-*')
hold on
plot(t,V)
title('Reynolds Decomposition of Velocity')
xlabel('Time')
ylabel('Mean+Fluctuating Velocity v')
grid on
```

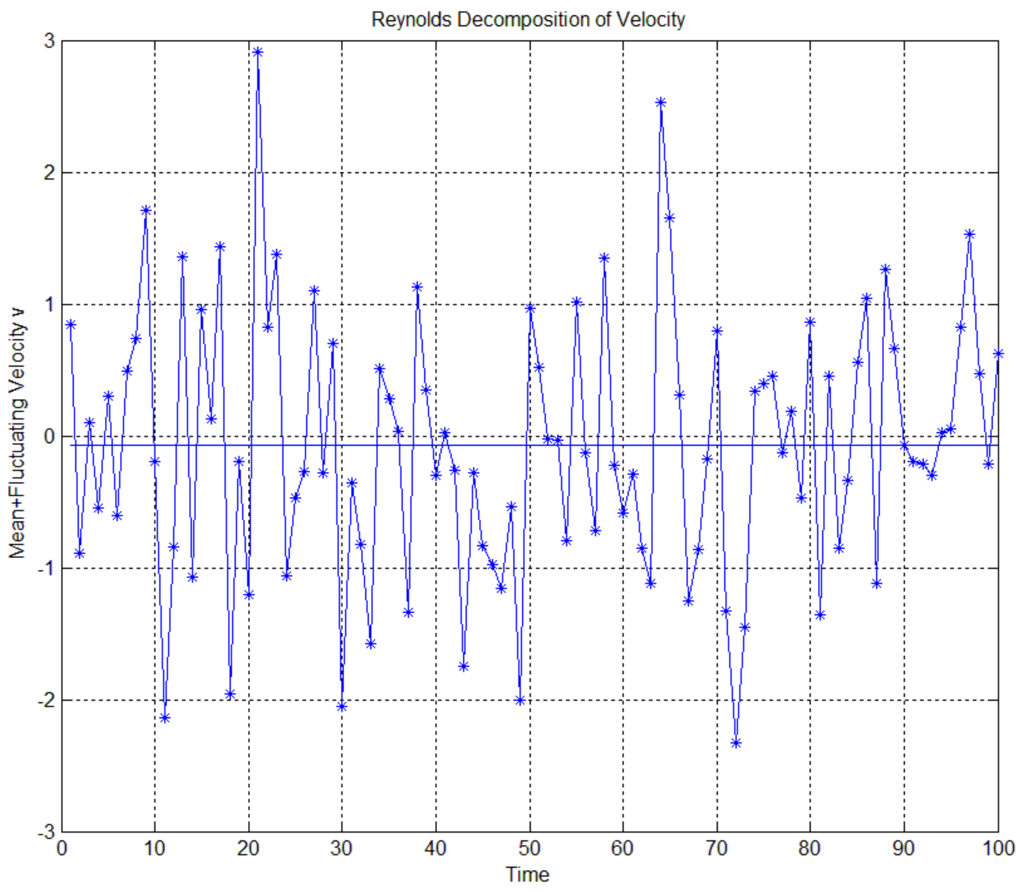


Figure 12 This plot shows the relationship between the main and fluctuating velocity with relation to time.

2-5 Finding the Norm of a velocity vector in a 2D plane made up of 10 Points:

Imagine we have a 2D box, this example breaks down the box into 100 tiny cubs. At the moment we don't need any data therefore we are using the random number generator. The random number generator assigns values for the u and v velocity components at each cell. Each cell has a coordinate of (i , j) and where i for the x direction and j in the y direction. Just as a reminder that the formula for calculating the norm of a vector is as follows:

$$V = \sqrt{u^2 + v^2} \tag{2.2}$$

In our example we have written the code that calculates the norm while in MATLAB there is a ready command that calculates for you the norm directly. The four used loops are used to represent the functions that matrix v relates to. In our case the matrix is a function of distance x (First cell),distance y (Second cell) ,time t (Third cell) and Scalar S (Fourth cell). The next three for loops represent a calculation of the norm value of the0while v(i,j,t,2) represents the v velocity component.

```
clc
clear
N=10;
t=(1:1:N);
for s=1:2;
for t=1:2;
for j=1:N;
for i=1:N;
v(i,j,t,s)=randn(1,1);
end
end
end
end
for t=1:2;
for j=1:N;
for i=1:N;
V(i,j,t)=(v(i,j,t,1)^2+v(i,j,t,2)^2)^0.5;
end
end
end
surf(V(:,:,2))
```

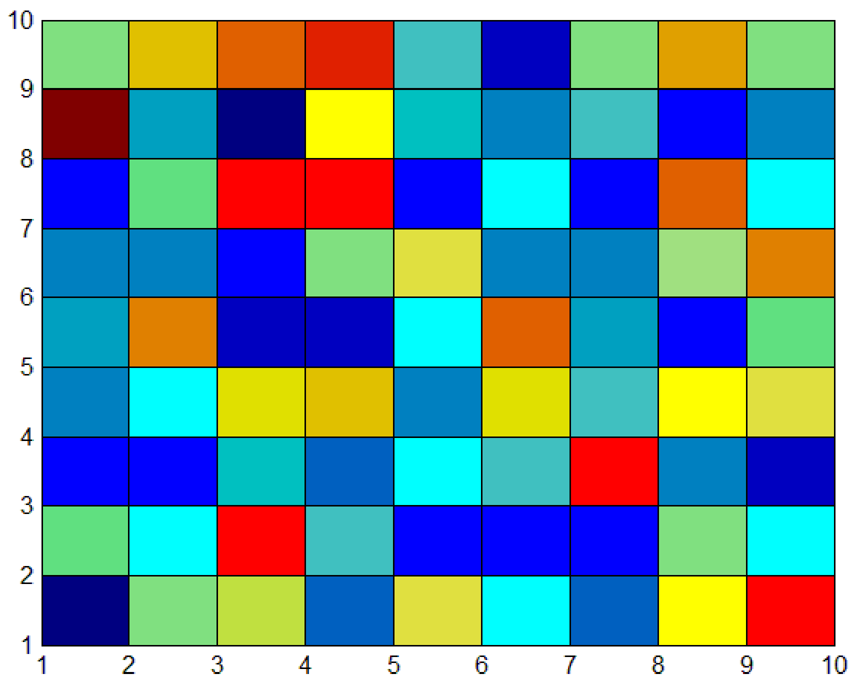


Figure 13 This plot doesn't look that fancy at the moment but for later stages of coding we will get to know how important this is an example.

2-6 Coding the Mass Diffusivity formula

The diffusion term is important for the researcher once he starts running simulations of several used species leading to the need of calculating the diffusion term of one species into another then substituting the value in the transport equation of the studied species,

$$\frac{\partial Y}{\partial t} + u\frac{\partial Y}{\partial x} + v\frac{\partial Y}{\partial y} + w\frac{\partial Y}{\partial z} = D\left(\frac{\partial^2 Y}{\partial x^2} + \frac{\partial^2 Y}{\partial y^2} + \frac{\partial^2 Y}{\partial z^2}\right) \tag{2.3}$$

there are lots of forms representing mass diffusion, for our example we will take this equation

$$D = D_0 \exp(-\frac{E}{RT}) \tag{2.4}$$

```
clc
clear
%Studying the diffusion case for a CO2
%Selecting 100 points
```

```
N=100;
%The discrete temperature difference
delat_t=0.1;
%Referance Temperature
T(1)=300;
%Data Generation Loop
for i=1:N-1;
T(i+1)=T(i)+delat_t;
R(i)=8.314; %j/mole.kelvin
EA(i)=160000;  %j/mole
b(i)=R(i)*T(i+1);
a(i)=EA(i)/b(i);
D0(i)=16;%*(0.001)^2;
D(i)=D0(i)*exp(-a(i));
end
%Data Plotting Section
plot(D(1:N-1),T(1:N-1))
figure(1)
title('Mass Diffusion vs Temperature')
xlabel('Diffusion mm^2/sec')
ylabel('Temeprature kelvin')
grid on
```

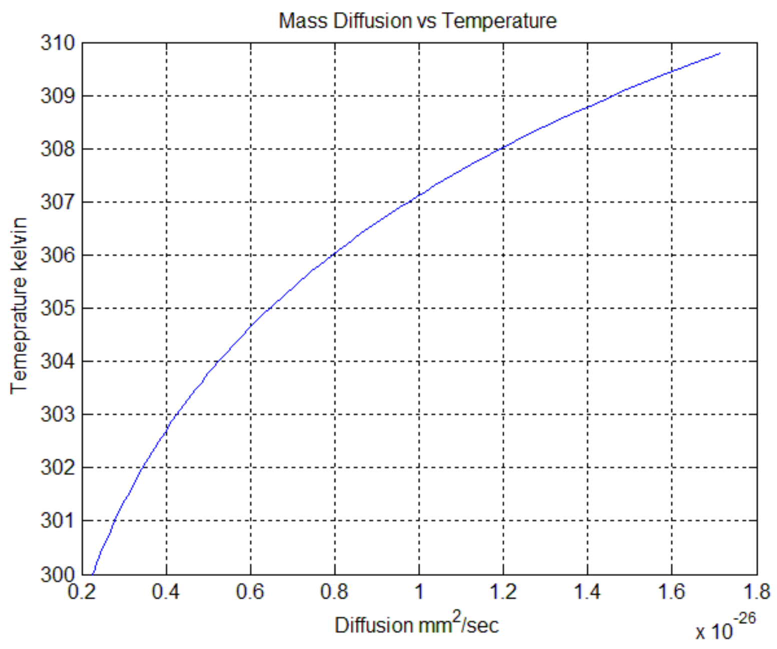


Figure 14 This plot represents the relationship between mass diffusion and temperature.

2-7 Coding the Sutherland formula:

The Sutherland formula is required to be used if you're going to code the Navier-Stokes equations . The Sutherland is used to calculate the viscosity term shown in red :

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}=\frac{\mu}{\rho}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)-\frac{\partial p}{\partial x} \tag{2.5}$$

This case represents an air example. The coded formula is as follows:

$$\mu=\mu_0\frac{T_0+C}{T+C}\left(\frac{T}{T_0}\right)^{\frac{3}{2}} \tag{2.6}$$

The generated data out of the code represents a one dimensional case. Some algebraic Operations and simplifications where used to ease the debugging process for the code. The code at the beginning clears any stored data from a previous run. One hundred points are take for our studied case the researcher can choice the number of points which seems practical for his case. Then comes the section that assigns constant values to named matrices to be retrieved and used at later stages of the calculations process. Now comes the part where we generate a discrete set of temperature data through a step of 10 degrees noting that the initial value is taken at 291.15k. Algebraic Operations and simplifications are used in the next section where they are represented in red:

$$\mu=\mu_0\frac{T_0+C}{T+C}\left(\frac{T}{T_0}\right)^{\frac{3}{2}} \tag{2.7}$$

The addition process is first used for term T_0+C and then for $T+C$ this is done through assigning them to matrices. Then the next step is assigning the divided terms to a matrix and finally multiplying all the matrices that represents the cut down forms.

Then comes the part at which views the generated data by using the display command that assigns the text for the heading of the table .what follows is the list of the generated data.

The last section is the plotting part as represented on figure15. As you progress in coding you start developing your own skills. The first couple of codes you write you would find yourself writing lots of notes as a primary process with lots of thinking taking your time. Then at later stages you don't even write notes you just write a code relating to the idea your studying.

```
clc
clear
N=100;  % For an air studied problem
              %Assigning values for the Constants
for i=1:N;
T0(i)=291.15;
C(i)=120;
myu0(i)=18.27;
end
% Generating a set of temperature Data
T(1)=291.15;
for i=1:N-1;
T(i+1)=T(i)+10;
end
%Algebraic Operations and simplifications
for i=1:N;
 %addition Process
 a(i)=T0(i)+C(i);
 b(i)=T(i)+C(i);
 %division Process
 c(i)=a(i)/b(i);
 d(i)=(T(i)/T0(i))^(1.5);
end
%output section
for i=1:N;
myu(i)=myu0(i)*c(i)*d(i);
end
%storing the produced data into a 2D matrix
for i=1:2;
for j=1:N;
if (i==1)
e(i,j)=T(i);
elseif (i==2)
```

```
e(i,j)=myu(i);
 end
 end
end
display('Temperature Table')
T'
display('viscosity Table')
myu'
display('Table');
e'
plot(myu,T,'r');
grid on
figure(1)
xlabel('myu')
ylabel('T')
Title('Viscosity vs Temperature')
```

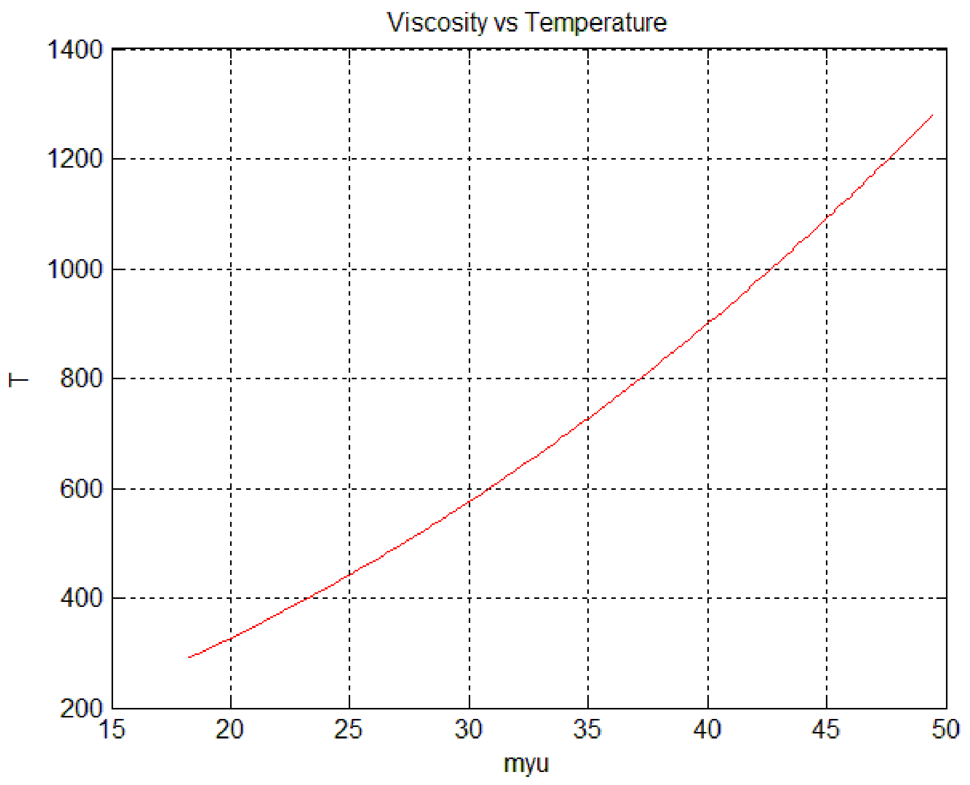


Figure 15 The relationship between temperature and viscosity.

2-8 Coding The Redlich-Kwong Equation of State:

The Redlich-Kwong equations is used to calculate thermodynamic properties for pure compound and mixtures with satisfactory accuracy. Our studied example will be for Argon

$$P = \frac{RT}{V_m - b} - \frac{a}{\sqrt{T}V_m(V_m + b)} \tag{2.8}$$

Where a is a constant that corrects for attractive potential of molecules, T_c is the temperature at the critical point so for our studied case which is Argon $T_c = 150.7$, P_c is the pressure at the critical point for Argon its $P_c = 4870$.

$$a = \frac{0.4275R^2T_c^{5/2}}{P_c} \tag{2.9}$$

while b is a constant that corrects for volume.

$$b = \frac{0.08664RT_c}{P_c} \tag{2.10}$$

The Redlich–Kwong equation is adequate for calculation of gas phase properties when the ratio of the pressure to the critical pressure (reduced pressure) is less than about one-half of the ratio of the temperature to the critical temperature (reduced temperature):

$$\frac{P}{P_c} < \frac{T}{2T_c} \tag{2.11}$$

The example here doesn't cover any plotting commands due to them being covered in previous examples. The first step starts by deleting any stored data from a previous run. We assign 10 points for our study. We use a one dimensional for loop for the constants assigning section. Then comes the discrete data generation part for temperature using a one dimensional for loop. We conduct algebraic operations on the first two equations for a and b through assigning the variables having a power value to a matrix then after all the complex parts have been assigned to the simple matrix form. The final step is the multiplication process between a for loop.

A conditional statement has been used to verify what kind of condition is achieved.

```
% Redlich-Kwong Equation of State for Argon
clc
clear
N=10;
%constants section
for i=1:N;
R(i)=8314;
TC(i)=150.7;
pc(i)=4870;
v(i)=0.001;
n(i)=1;
Vm(i)=v(i)/n(i);
end
%Generating Temperature Data
T(1)=273.15;
for i=1:N-1;
T(i+1)=T(i)+1;
end
%Algebraic operations for the two equations
for i=1:N;
aa(i)=(R(i))^2;
aaa(i)=(TC(i))^(5/2);
aaaa(i)=(aa(i)*aaa(i))/pc(i);
a(i)=0.4275*aaaa(i);
bb(i)=R(i)*TC(i);
bbb(i)=bb(i)/pc(i);
b(i)=0.08664*bbb(i);
end
%Algebraic operations for the Redlich formula
for i=1:N;
ccc(i)=Vm(i)-b(i);
cc(i)=R(i)*T(i);
c(i)=cc(i)/ccc(i);
dddd(i)=(T(i))^(0.5);
ddd(i)=Vm(i)*(Vm(i)+b(i));
dd(i)=ddd(i)*dddd(i);
d(i)=a(i)/dd(i);
p(i)=c(i)-d(i);
e(i)=0.5*(T(i)/TC(i));
ee(i)=p(i)/pc(i);
if (ee(i)<e(i))
display('It Satisfies the adequate Condition ')
else
```

```
display('It does not Satisfy the adequate Condition ')
end
end
%Storing the data into a matrix
xx=0;
for i=1:N;
for j=1:3;
xx=xx+1;
if (j==1)
A(i,j)=xx;
elseif (j==2)
A(i,j)=T(i);
elseif (j==3)
A(i,j)=p(i);
end
end
end
display('Stored Data Matrix')
A
display('Stored Data Matrix colum 1')
A(1:10,1)
display('Stored Data Matrix colum 2')
A(1:10,2)
display('Stored Data Matrix colum 3')
A(1:10,3)
```

Chapter 3

Initial Coding Skills

3-1 Introduction:

This chapter is condensed with coding skills. Remember you might not get the point of the example from the first time. Read the example and see what it gives as an output. Then try looking at the section that has been assigned values for the constants try changing some of their values and see the outcome of such a change. The covered skills give the researcher a substantial push forward to work independently on writing small and medium sized codes.

3-2 Working with for loops:

The number of used for loops depends on how many variables are of interest. The trend usually favoured by CFD researchers is in using a five cell matrix. One of the regular mistakes that students make is assigning a real number or a negative number to the cell reference which causes the compilation process comes to a halt. Integer numbers have to be used as a cell reference numbers because the student has to think of the used matrix as a storage bank, this storage bank has lots of storage boxes and the only way to access these boxes is by knowing their reference number. Something that is very similar in life that we encounter if you don't have the phone number of your friend you cannot contact him to get daily life info.

3-2-1 Working with a one dimensional matrix:

This is the regarded is the first step towards complex codes, plotting the data of the matrix usually comes against the number of used stored data cells or against another matrix that has the same dimension as the first. This code example generates a matrix referring to pressure fluctuations using rand function.

```
clc
clear
M=10;
for i=1:M;
p(i)=rand(1,1);
end
plot(p)
grid on
figure(1)
xlabel('Matrix Cell Number')
ylabel('Pressure Fluctuations')
Title('Pressure Fluctuations vs Matrix Cell Number')
legend('Pressure')
```

The intention of applying an i integer to the statement p(i)=rand(1,1) so that at each value of the I matrix p stores the value of each random number that is generated. Then comes the plotting case with the use of the plot command due to that we only used on matrix MATLAB assumes that matrix p represents the y axis while the number of cells represents the y axis. The generated plot might look similar to this depending on the generated data by the random generator.

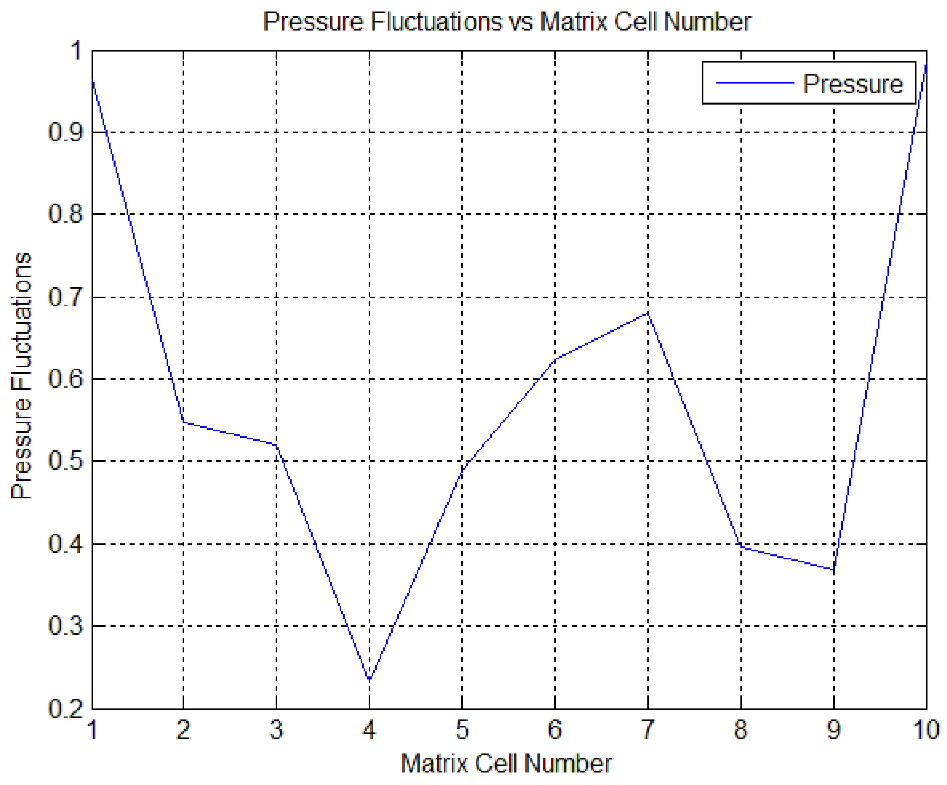


Figure 16 A plot showing the relationship between pressure fluctuations and matrix cell number.

The next example relies on a random number generator to create a one dimensional matrix inside the loop a counter was added . The main task of the counter is to create a volume matrix that has 10 cells, where the first cell starts with the value of 1 where (vvv) was given an initial value of zero. Once the number is counted in (vvv=vvv+1) it is then stored in matrix v(i) at the specified cell number i. The researcher can offset the counter value by giving (vvv) an initial value which would lead to a relative value in respect to the previous point. Students find it difficult to use counters. The main problem is that they say how can (i) have the same variable on two sides. The answer is that you have to think of the statement that it has two sides a left side and a right side. The left hand side as an unknown and the right hand side as known. So as a condition the variable has to be assigned a value for the right hand side to work. The initial value for a counter has to be outside the for loop to work. So if we have a loop that has to be run two times the compiler reads the initial value then starts the first loop and counts the first value. Then when the second loop starts the compiler takes the value that was calculated from the previous step. The same applies for an n number of loops.

```
clc
clear
M=10;
vvv=0;
for i=1:M;
p(i)=rand(1,1);
vvv=vvv+1;
v(i)=vvv;
end
plot(v,p)
grid on
figure(1)
xlabel('Volume Axis')
ylabel('Pressure Axis')
Title('Pressure vs Volume')
```

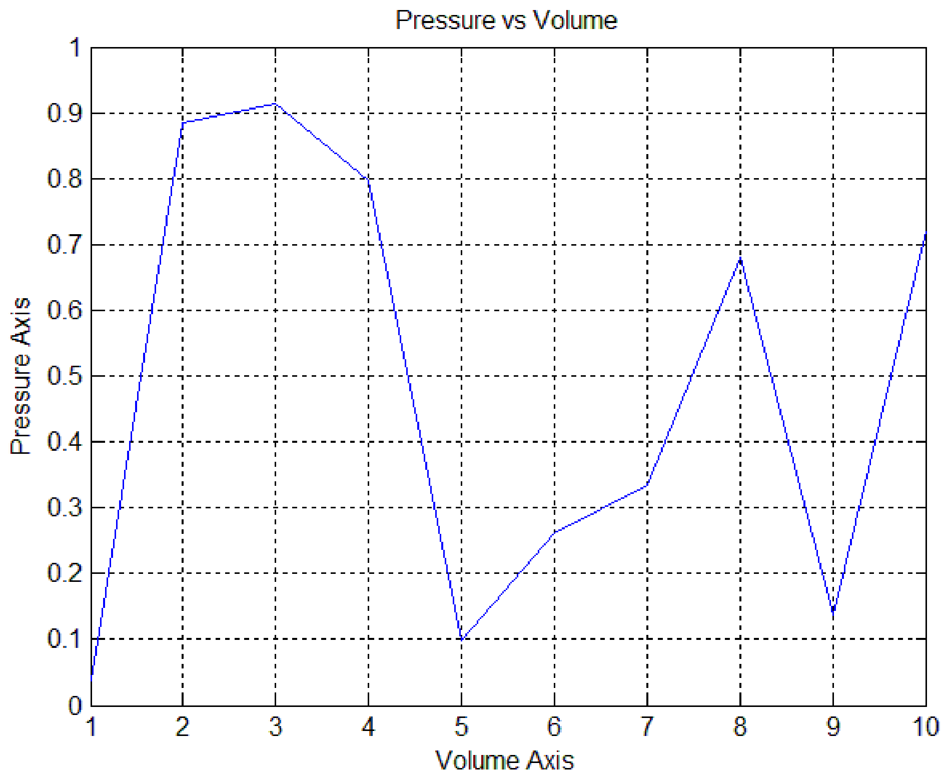


Figure 17 A plot showing the relationship between pressure and volume.

3-2-2 Working with a two dimensional matrix:

This example is conducted on a two dimensional XY plane case for the ideal gas equation. That's why two loops are used. So if we split the XY plane into 10 points in height and 10 points in width we can assign a pressure value at each point . The Ideal gas equation

$$P = M\frac{RT}{V} \tag{3.1}$$

Our example is conducted on a mass of Air that is 1 kg. Using the random number generator that creates positive numbers an Air Volume matrix is created.

```
clc
clear
N=10;
R=8.314;
for j=1:N;
for i=1:N;
v(i,j)=100000*rand(1,1);
t(i,j)=randn+20;
T(i,j)=t(i,j)+273;
M(i,j)=1;
p(i,j)=(M(i,j)*R*T(i,j))/v(i,j);
end
end
plot(p(:,3))
grid on
figure(1)
xlabel('Volume Axis')
ylabel('Pressure Axis')
Title('Pressure at Plane Y=3')
figure(2)
surf(p)
xlabel('X Axis')
ylabel('Y Axis')
Zlabel('Pressure Axis')
Title('Pressure Visualization in a XY Plane')
```

Then comes creating a temperature matrix first we generate a changing temperature that has a mean value of 20 C and a fluctuating value. Once this matrix is calculated the temperature is calculated in Kelvin's. A mass of 1 kg is assigned to each cell.

Once these values are created pressure is calculated. Finally the calculated data is plotted.

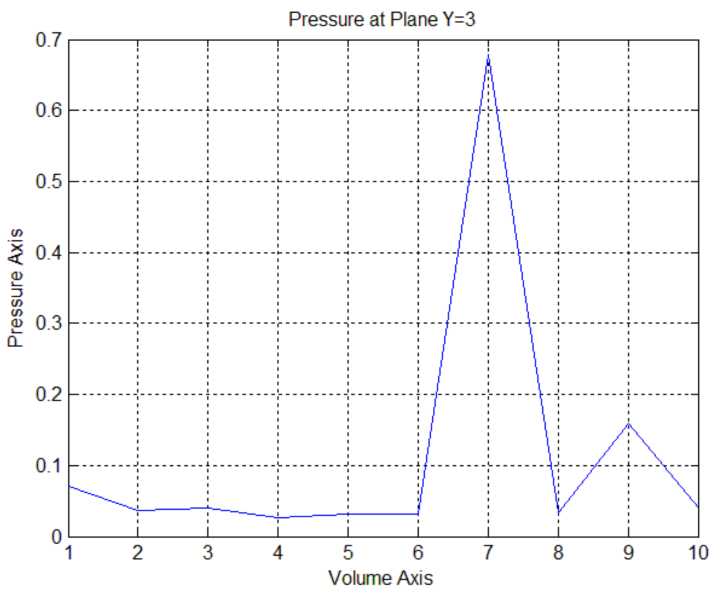


Figure 18 This plot shows a cross section of XZ plane at y=3, this plot can be used in comparison with the figure19 to see the location of the cross section plane.

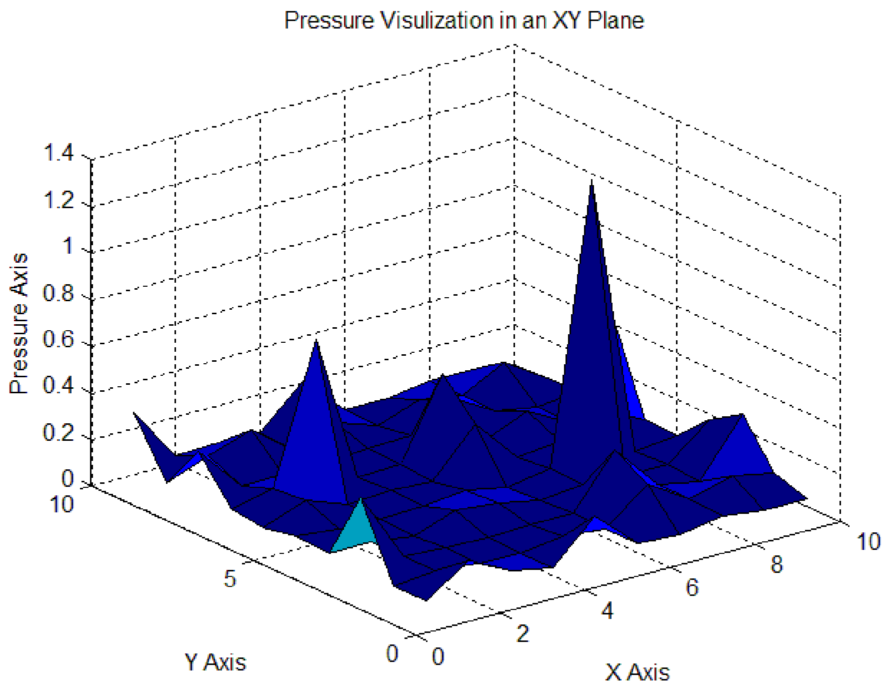


Figure 19 This is surface with XY plane holding the coordinates of the points and the z axis represents the value of the calculated pressure.

3-2-3 Working with a three dimensional matrix:

The researcher usually encounters this for a 3 dimensional space disceretized case. This example is concerned in constructing a calculation process of the change of momentum in the Navier stokes equation on the x axis.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = \vartheta\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{3.2}$$

we will split the equation into a left hand side and a right hand side

$$RS = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \tag{3.3}$$

We neglect the time derivative so that we represent a steady state flow case

$$RS = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \tag{3.4}$$

The left hand side for our studied case is not of interest.

$$LS = \vartheta\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{3.5}$$

Now that we have the form of the equation we want to construct a numerical algorithm, we look at each part of the equation separately and see what is that part a function of as stated below.

The velocity u in the x axis direction as a function of i j and k node number.

$$u = f(i, j, k) \tag{3.6}$$

The velocity v in the y axis direction as a function of i j and k node number.

$$v = f(i, j, k) \tag{3.7}$$

The velocity w in the z axis direction as a function of i j and k node number.

$$w = f(i,j,k) \tag{3.8}$$

The derivative of u velocity in the x axis direction as a function of i j and k node number.

$$\frac{\partial u}{\partial x} = f(i,j,k) \tag{3.9}$$

The derivative of u velocity in the y axis direction as a function of i j and k node number.

$$\frac{\partial u}{\partial y} = f(i,j,k) \tag{3.10}$$

The derivative of u velocity in the z axis direction as a function of i j and k node number.

$$\frac{\partial u}{\partial z} = f(i,j,k) \tag{3.11}$$

The Right hand side of the steady state form of the Navier Stokes equation (RS) as a function of i j and k node number.

$$RS = f(i,j,k) \tag{3.12}$$

The velocity u in the x axis direction as a function of i j and k node number.

$$RS = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \tag{3.13}$$

A note to the reader this example is intended to create a working code based on a random set of data. This will help the student to construct more complex examples in near future.

```
clc
clear
L=20;
M=20;
```

```
N=20;
contourdensity=12;
for i=1:L;
for j=1:M;
for k=1:N;
u(i,j,k)=randn(1,1);
v(i,j,k)=randn(1,1);
w(i,j,k)=randn(1,1);
dudx(i,j,k)=randn(1,1);
dudy(i,j,k)=randn(1,1);
dudz(i,j,k)=randn(1,1);
RS(i,j,k)=u(i,j,k)*dudx(i,j,k)+v(i,j,k)*dudy(i,j,k)+w(i,
j,k)*dudz(i,j,k);
end
end
end
figure('Position',[100 100 1700 900])
contour(RS(:,:,5), contourdensity)
xlabel('x axis')
ylabel('y axis')
title('Plotting the Contour of the RS Values at Z=5')
set(gca,'XLim',[1 M],'YLim',[0 N])
grid on
```

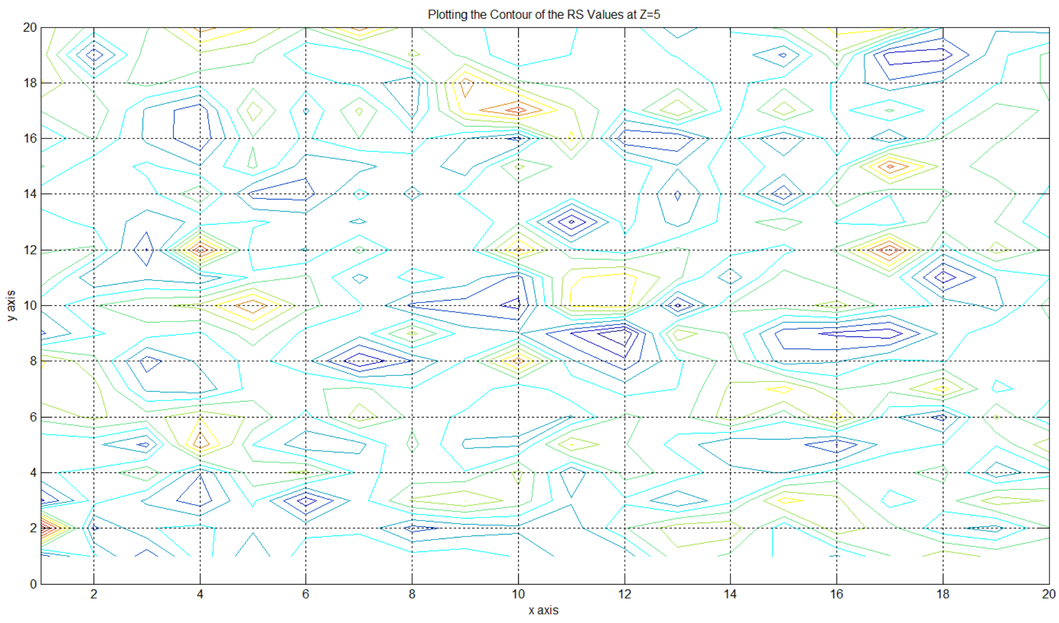


Figure 20 A contour plot at z plane number 5.

3-2-4 Working with a four dimensional matrix:

This case comes up when working with a space disceretized case with the introduction of time as the fourth dimension for the matrix. This example is concerned in constructing a calculation process of the change of momentum in the Navier stokes equation on the x axis.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = \vartheta\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{3.14}$$

we will split the equation into a left hand side and a right hand side

$$RS = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \tag{3.15}$$

The left hand side for our studied case is not of interest.

$$LS = \vartheta\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{3.16}$$

Now that we have the form of the equation we want to construct a numerical algorithm, we look at each part of the equation separately and see what is that part a function of as stated below.

The velocity u in the x axis direction as a function of space and time of a studied node.

$$u = f(i, j, k, t) \tag{3.17}$$

The velocity v in the y axis direction as a function of space and time of a studied node.

$$v = f(i, j, k, t) \tag{3.18}$$

The velocity w in the z axis direction as a function of space and time of a studied node.

$$w = f(i, j, k, t) \tag{3.19}$$

The derivative of u velocity in the x axis direction as a function of space and time of a studied node.

$$\frac{\partial u}{\partial x} = f(i,j,k,t) \tag{3.20}$$

The derivative of u velocity in the y axis direction as a function of space and time of a studied node.

$$\frac{\partial u}{\partial y} = f(i,j,k,t) \tag{3.21}$$

The derivative of u velocity in the z axis direction as a function of space and time of a studied node.

$$\frac{\partial u}{\partial z} = f(i,j,k,t) \tag{3.22}$$

The Right hand side of the steady state form of the Navier Stokes equation (RS) as a function of space and time of a studied node.

$$RS = f(i,j,k.t) \tag{3.23}$$

This finally leads us to calculate the right hand side at each specified point at a specified time.

$$RS = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \tag{3.24}$$

By assigning $RS = 0$ we can evaluate the value of the velocity at time step 2

$$u = u_0 + \Delta t(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}) \tag{3.25}$$

A note to the reader this example is intended to create a working code based on a random set of data which can be used to provide some kind of initialization for a simulation. This will help the student to construct more complex examples in near future. If you can construct it for a one-dimensional case you can construct it for a three dimensional case.

```
clc
clear
L=30;
M=30;
N=30;
dt=1;
time=1;
for t=1:time;
for i=1:L;
for j=1:M;
for k=1:N;
u0(i,j,k,t)=0;
u(i,j,k,t)=randn(1,1);
v(i,j,k,t)=randn(1,1);
w(i,j,k,t)=randn(1,1);
dudx(i,j,k,t)=randn(1,1);
dudy(i,j,k,t)=randn(1,1);
dudz(i,j,k,t)=randn(1,1);
u(i,j,k,t+1)=
u0(i,j,k,t)+dt*(u(i,j,k,t)*dudx(i,j,k,t)+v(i,j,k,t)*dudy
(i,j,k,t)+w(i,j,k,t)*dudz(i,j,k,t));
end
end
end
end
figure('Position',[100 100 1700 900])
contour(u(:,:,1,2),12)
xlabel('x axis')
ylabel('y axis')
title('Plotting the U Velocity at Time Step 2')
set(gca,'XLim',[0 M],'YLim',[0 N])
grid on
```

3-2-5 Working with a five dimensional matrix:

This case comes up when working with a space disceretized case adding to it a time variable and a scalar variable. The variable s refers to the number of studied scalars. This method shows how much size cut down of the code can be done in comparison with the previous example. This method can prove to be efficient but hard to understand if the researcher wants to go through a code that he hasn't written by himself. Programmers have their own styles of coding. This is something a researcher would encounter during his studies.

For our example S can take three values which means the box domain that we have broken into discrete parts, can have a u , v and w scalar quantities. At the moment we are relying on a random number generator for future examples we will rely on more realistic numbers through the solution process of a number of partial differential equations which form the Navier Stokes Equations. The box domain is assigned scalar values for a 10 second steps. The plotting part draws a contour plot of a z=1 plane cross section for scalar velocity w at time 2 second, this is the interpretation of the command statement `contour(Q(:,:,1,3,2))`.

```
clc
clear
L=30;
M=30;
N=30;
time=10;
for s=1:3;
for t=1:time;
for i=1:L;
for j=1:M;
for k=1:N;
Q(i,j,k,t,s)=randn(1,1);
end
end
end
end
end
figure('Position',[100 100 1700 900])
contour(Q(:,:,1,3,2),12)
xlabel('x axis')
ylabel('y axis')
title('Plotting the Contour of the v Velocity')
set(gca,'XLim',[0 M],'YLim',[0 N])
grid on
```

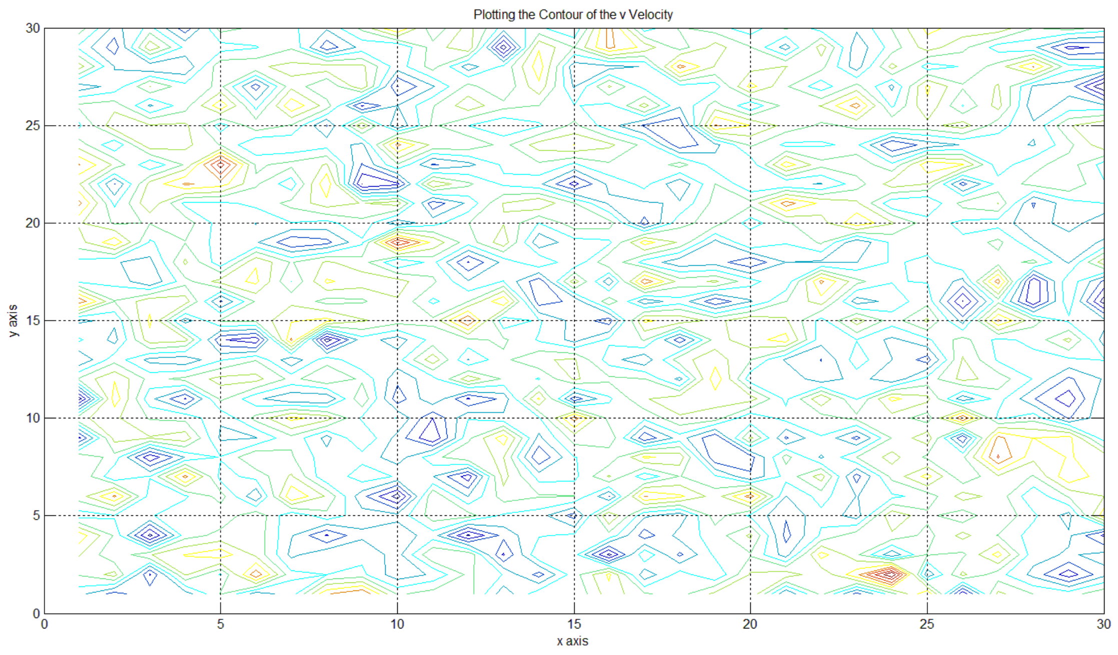


Figure 21 The plotting part draws a contour plot of a z=1 plane cross section for scalar velocity ***w*** at time 2 second.

3-3 Matrix use for Storing Data:

One of the fundamental issues in constructing your code is in generating your data sets which are required for the later stage of data analysis. The following example has two for loops. The first loop uses a random number generator to create the data set, assigning a cell number to each matrix index tells MATLAB to store the generated number. The next loop stores the generated temperature after converting it from Celsius to Kelvin. Then comes the data plotting part.

```
clc
clear
L=10;
for i=1:L;
    t(i)=1.2*randn;
end
for i=1:L;
    T(i)=t(i)+273;
end
plot(T)
```

```
title('Temperature Vs Number of Readings')
grid on
xlabel('Number of Readings of Temperature')
ylabel('Temperature (K)')
```

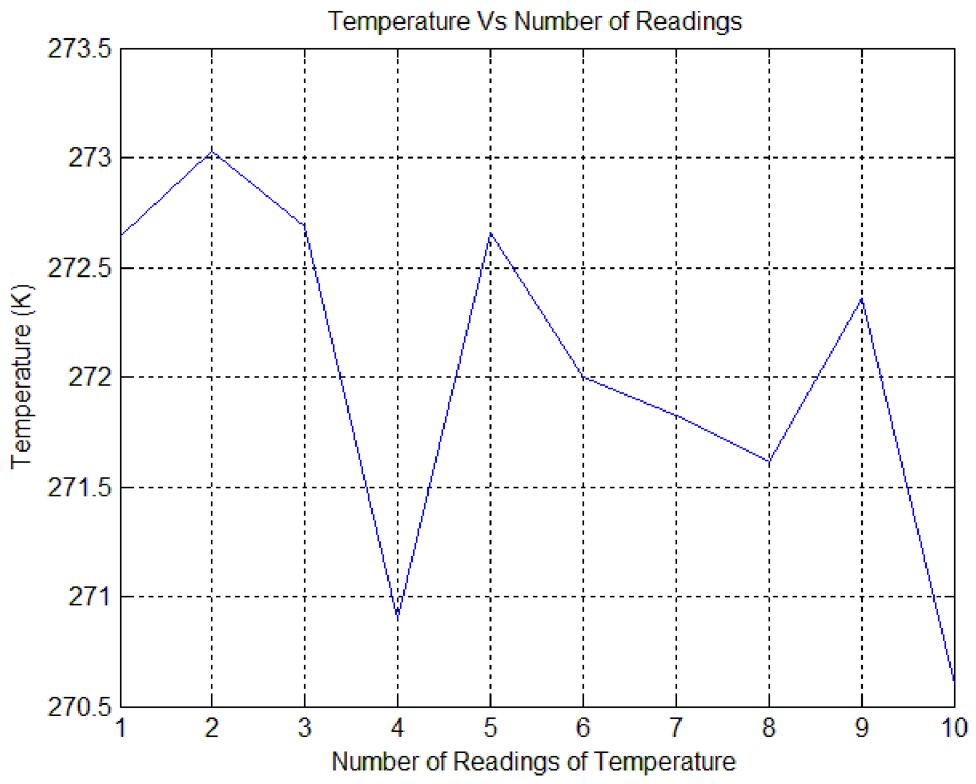


Figure 22 Plotting the temperature fluctuating data against number readings.

3-4 Generating Discrete data:

Why do we need to generate discrete data? Making data tables requires generating sets of data within a pre-specified range. The added value can have a static value or a dynamic value depending on the studied case. The first two examples here represent a problem that the researcher would come across. The first example generates random velocities v and random discrete Δt then the code calculates the CFL condition. The last section of the example is the plotting part. Using the counter we can time step. Time stepping is through adding a Δt value to the previous time t. For a computer complier it understands it in the following form while it is stated in a for loop:

$$t = t + \Delta t \tag{3.26}$$

We write it when we want to solve it by hand as follows:

$$t = t_0 + \Delta t \tag{3.27}$$

Then depending on how many times we want to repeat the loop

$$t_1 = t_0 + \Delta t \tag{3.28}$$

$$t_2 = t_1 + \Delta t \tag{3.29}$$

$$t_3 = t_2 + \Delta t \tag{3.30}$$

....

$$t_{n+1} = t_n + \Delta t \tag{3.31}$$

While for the compiler we just have to state the following lines where M refers to the number of times of repetition of the loop process. The initial value for the loop counter is `t(1)=0`. Assigning dynamic positive numbers for the discrete added value is through using `dt(i)=0.001*rand(1,1)`.

```
M=30;
t(1)=0;
for i=1:M;
dt(i)=0.001*rand(1,1)
t(i+1)=t(i)+dt(i);
end
```

The CFL formula for each time step is as follows:

$$\mathrm{CFL} = \frac{u\,\Delta t}{\Delta x} \tag{3.32}$$

The **Courant–Friedrichs–Lewy condition** (CFL condition) is a necessary condition for convergence while solving certain partial differential equations (usually hyperbolic PDEs) numerically by the method of finite differences. It arises when explicit time-marching schemes are used for the numerical solution. As a consequence, the time step must be less than a certain time in many explicit time-marching computer simulations, otherwise the simulation will produce wildly incorrect results.

```
clc
clear
M=30;
t(1)=0;
for i=1:M;
dx(i)=0.3;
V(i)=rand(1,1);
dt(i)=0.001*rand(1,1)
t(i+1)=t(i)+dt(i);
CFL(i)=(V(i)*dt(i))/dx(i);
end
plot(t(1:M),CFL,'-*')
grid on
title(' CFL vs Time Stepping')
xlabel('Time Stepping')
ylabel('CFL')
```

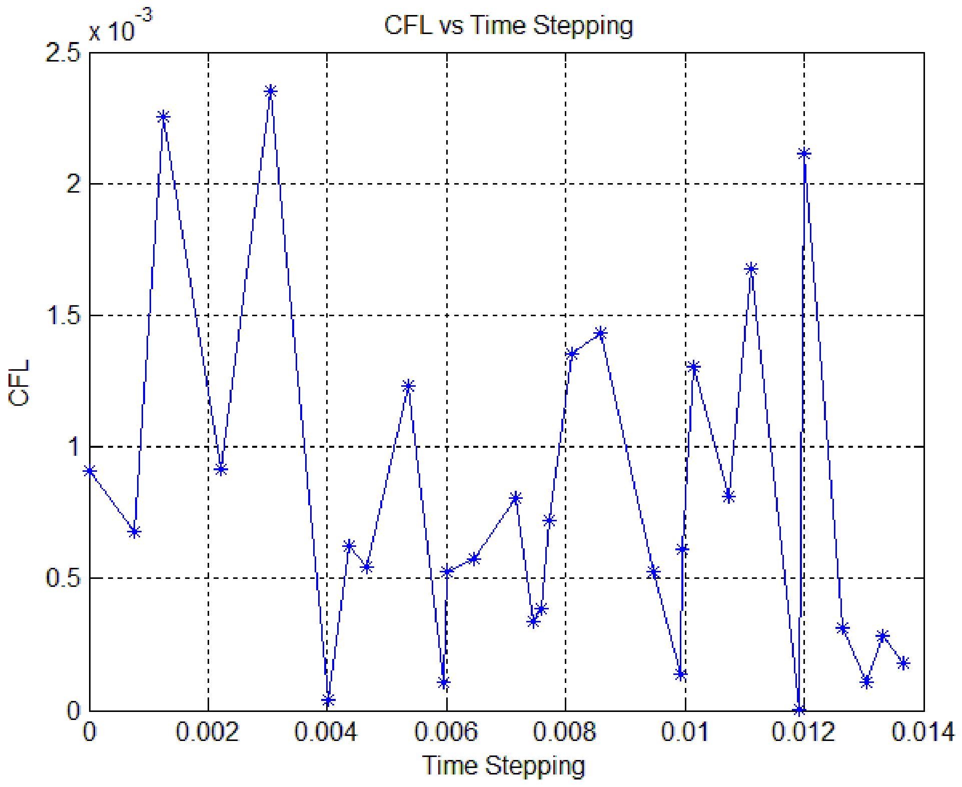


Figure 23 Its clear how the CFL number is fluctuating and that is due to the variable values of the discrete Δt, the x axis representing the time advance shows how the time steps differ from one point to the other.

The next example is based on the same structure of the previous example except in having a constant value of Δt. The after effects

```
clc
clear
M=30;
t(1)=0;
for i=1:M;
dx(i)=0.3;
V(i)=rand(1,1);
dt(i)=0.001;%*rand(1,1)
t(i+1)=t(i)+dt(i);
CFL(i)=(V(i)*dt(i))/dx(i);
end
plot(t(1:M),CFL,'-*')
grid on
title(' CFL vs Time Stepping')
xlabel('Time Stepping')
ylabel('CFL')
```

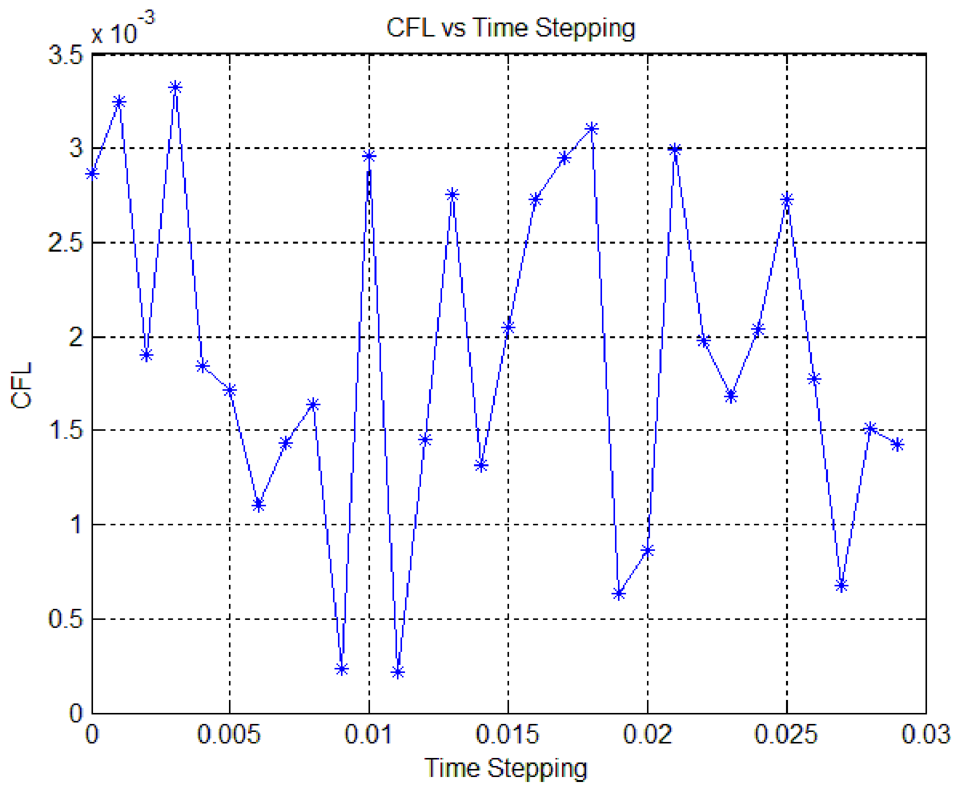


Figure 24 Plotting the CFL number against time. The x axis is from 0 to 0.03 where the advancement of $\boldsymbol{\Delta t}$ is constant.

3-5 Using Random Number generators:

The two most commonly used random generators in this book are rand and randn. The first generates positive numbers and the second produces positive and negative numbers. In this book random number generators are used to construct the numerical model. Once the code is constructed and data is generated plotting the output data is to check that the plot is similar to ones found in scientific literature. In previous examples the random functions had been used on the basis.

3-6 Plotting data:

The user can use this code for his data analysis, it represents a very simple case of plotting a point having a coordinate x=1 and y=1. The set of commands are used regularly for the visualization part of the code, you can copy and paste this code into the command window. The text in red can be modified by the user to suite his needs.

```
plot(1,1,'*');
grid on
figure(1)
xlabel('Enter x Axis Label')
ylabel('Enter y Axis Label')
Title('Enter Here the Title of the Table')
legend('Enter Data Name')
```

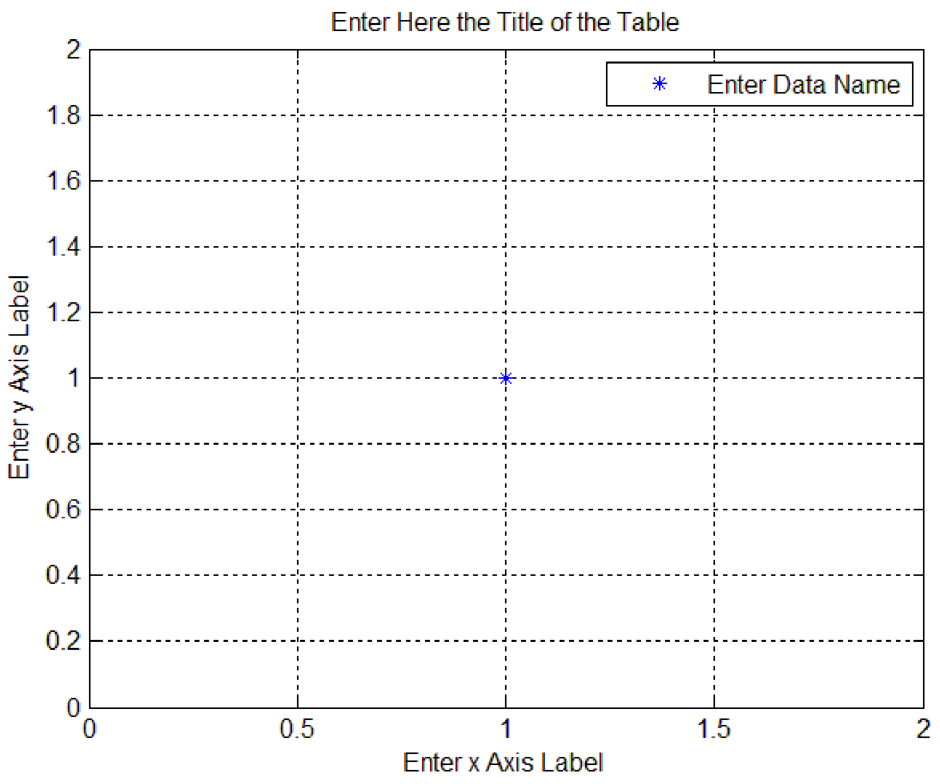


Figure 25 By copying and pasting the code into the command window the output should look like this.

3-7 Recommended Reading List

Helpful References in the use of matrix operations .The classification is based on the level of the researcher**.**

For Beginners:

1- Will be added in next draft.

For Intermediates:

1-Will be added in next draft.

For Professionals:

This is where the researchers part comes in finding the required journal.

Chapter 4

Mesh Generation

4-1 Introduction:

This chapter covers several code examples this material is just to give the reader a push into seeing how is the grid generated and what parameters effect. You can save the simulation run time if you run the mesh generation file independently and save the generated data into a data file format and read it to your main code every time you run a simulation. Most CFD software come with interface tools that allow you to generate the mesh without going into the complexity and the hundreds lines of written code. Encountering complex meshes happens in engineering applications such as turbine blades, aerofoil profiles, casted metal solid parts, ...etc.

4-2 Why do we need a mesh ?

When you solve a problem you need to set its dimensions. By allocating a certain volume to solve your problem in space. The next step comes in breaking this volume into discrete volumes. Then you assign each discrete volume a centre of gravity. This means once you assign the discrete volume its centre you can extract its coordinates in relation to a preselected point which is most probably be (0,0,0). Then you assign the parameter that you want to study at that point. The parameter can represent pressure, Temperature or some scalar quantities such as heat flux or velocities this list can go on and on. Once you select the parameter you want to study for example velocity that means you need to use the momentum equations which are the Navier

Stokes equations. The derivatives in the equations are linked to the generated nodes. There are various types of Meshes that are used

4-3 Uniform Mesh:

The uniform mesh is usually the first type of mesh the student gets exposed to. A staggered grid is used where you have a grid to represent the space discretization while the other grid is for the pressure term. This kind of grid is encountered in incompressible cases. The following example code starts with the clearing command of any data stored from a previous run. The starting section is to define the limits of the region of study. This is done by assigning the x_{min} minimum and x_{max} maximum values in the x and y axis. By subtracting the two values of the minimum and maximum you obtain the distance value between the two points.

$$L_x = x_{max} - x_{min} \tag{4.1}$$

Hence

$$L_y = y_{max} - y_{min} \tag{4.2}$$

Then by assigning the number of grid nodes in the x and y direction you can find the discrete distances between the grid nodes.

$$dx = \frac{L_x}{m} \tag{4.3}$$

Where

$$dy = \frac{L_y}{n} \tag{4.4}$$

So each point is related to

$$x(i,j) = i.dx \tag{4.5}$$

Hence

$$y(i, j) = j.dy \tag{4.6}$$

a not to the researcher that students always have difficulty in differentiating between the generated data of coordinates and node number due to their interconnection with the matrix indexing. We first find the absolute mesh and we then find the relative mesh.

$$x(i, j) = x_0(i, j) + i.dx \tag{4.7}$$

In addition

$$y(i, j) = y_0(i, j) + j.dy \tag{4.8}$$

One of the strong functions that MATLAB is provided with is the meshgrid:

```
[x,y]=meshgrid(x0:dx:x00,y0:dy:y00);
```

```
clc
clear
%Absolute Reference Mesh Input Data
x0=0;
x00=1;
y0=0;
y00=1;
lx=x00-x0;
ly=y00-y0;
M=10;
N=10;
dx=lx/M;
dy=ly/N;
%offsetting Distance
ddx=+0.5;
ddy=+0.5;
%Relative Reference Mesh Input Data
x10=0;
x11=1;
y10=0;
y11=1;
lx1=(x11-x10)+ddx;
ly1=(y11-y10)+ddy;
M1=10;
N1=10;
```

```
dx1=lx1/M1;
dy1=ly1/N1;
% Matrix Mesh Generation matlab Command
[x,y]=meshgrid(x0:dx:x00,y0:dy:y00);
[x1,y1]=meshgrid(x10:dx1:x11,y10:dy1:y11);
% Output Data Visualization
% figure(1)
% plot(x,y,'*b')
% %figure(2)
% hold on
% plot(x1,y1,'*r')
% hold off
% grid on
figure(1)
subplot(2,2,1)
plot(x,y,'*b')
xlabel('x')
ylabel('y')
grid on
title('First Grid ')
subplot(2,2,2)
plot(x1,y1,'*r')
xlabel('x')
ylabel('y')
grid on
title('Second Grid ')
subplot(2,2,[3 4])
plot(x,y,'*b',x1,y1,'*r')
xlabel('x')
ylabel('y')
grid(gca,'minor')
title('Staggered Grid ')
```

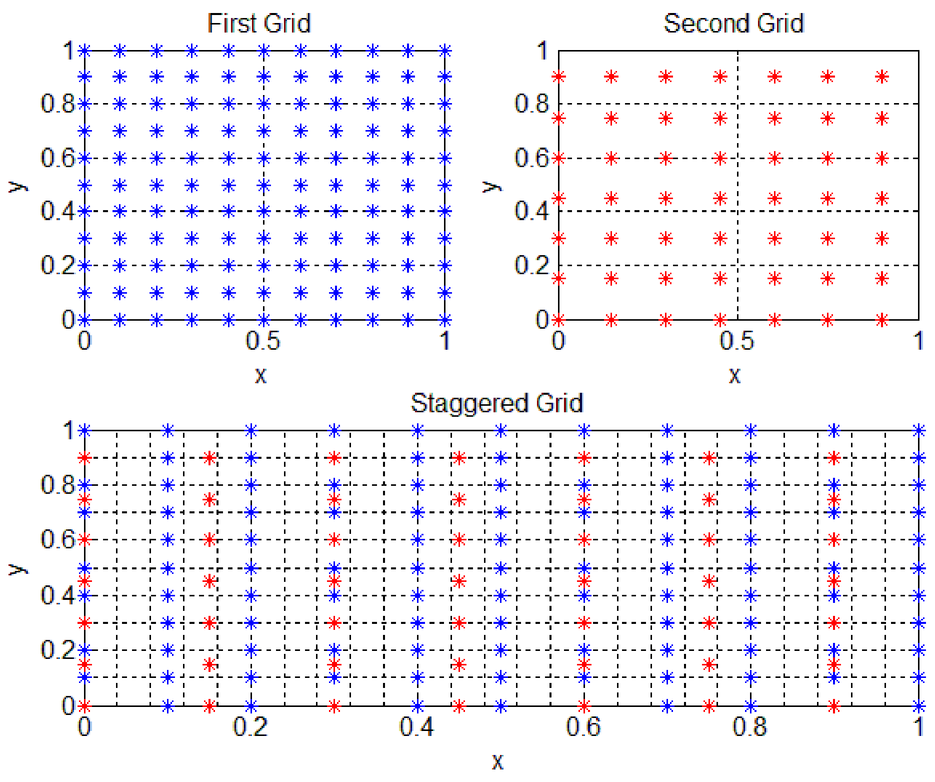


Figure 26 The use of three sub plots where the top left one is for the space discreteization while the top right is for the pressure grid, the two compound grids together is shown at the bottom one.

This example creates a box which is something that the researcher would encounter every now and then when he constructs his model and wants to try it out the most convenient mesh is the box one. Remember that if you can generate a grid on a 2d plane you can generate one in a 2d plane. Note that in this example because we are working on a 3d object 3 dimensional for loops are used. This is done by assigning the x_{min} minimum and x_{max} maximum values in the x and y axis. By subtracting the two values of the minimum and maximum you obtain the distance value between the two points.

$$L_x = x_{max} - x_{min} \tag{4.9}$$

Hence

$$L_y = y_{max} - y_{min} \tag{4.10}$$

Also

$$L_z = z_{max} - z_{min} \tag{4.11}$$

Then by assigning the number of grid nodes in the x and y direction you can find the discrete distances between the grid nodes.

$$dx = \frac{L_x}{m} \tag{4.12}$$

Where

$$dy = \frac{L_y}{n} \tag{4.13}$$

Also

$$dz = \frac{L_z}{l} \tag{4.14}$$

So each point is related to

$$x(i, j, k) = i.dx \tag{4.15}$$

Hence

$$y(i, j, k) = j.dy \tag{4.16}$$

In addition

$$z(i, j, k) = k.dz \tag{4.17}$$

```
clc
clear

M=7;
N=7;
L=7;
LX=1;
LY=1;
LZ=1;
dx=LX/M;
dy=LY/N;
dz=LZ/L;

for k=1:L;
    if (k==1 |k==L)
    c=0;
    else
        c=1;
    end
for i=1:M;
    for j=1:N;
        if (j==1 | i==1)
        u(i,j,k)=0*c;
        elseif (j==N | i==M)
        u(i,j,k)=0;
        else
        u(i,j,k)=c*randn(1,1);
        end
    end
end
end


for i=1:M;
    for j=1:N;
        for k=1:L;
            x(i,j,k)=i*dx;
            y(i,j,k)=j*dy;
            z(i,j,k)=k*dz;
        end
    end
end


figure
for i=1:M;
    for j=1:N;
        for k=1:L;
            plot3(x(i,j,k),y(i,j,k),z(i,j,k),'*r')
            end
        hold on
        end
end
title('Domain Points')
xlabel('x')
ylabel('y')
zlabel('z')

tic
for i=1:M;
    for j=1:N;
        for k=1:L;

xc=x(i,j,k);
yc=y(i,j,k);
zc=z(i,j,k);
hold on
plot3(xc,yc,zc,'*b')
axis equal
%--------------------------------------------------------
-
%Bottom Plane

xc1=xc+dx/2;
yc1=yc+dy/2;
zc1=zc-dz/2;

xc2=xc-dx/2;
yc2=yc+dy/2;
zc2=zc-dz/2;

xc3=xc-dx/2;
yc3=yc-dy/2;
zc3=zc-dz/2;

xc4=xc+dx/2;
yc4=yc-dy/2;
zc4=zc-dz/2;

v1=[xc1,xc2];
v2=[yc1,yc2];
v3=[zc1,zc2];
f=line(v1,v2,v3);

v1=[xc2,xc3];
v2=[yc2,yc3];
v3=[zc2,zc3];
f=line(v1,v2,v3);

v1=[xc3,xc4];
v2=[yc3,yc4];
v3=[zc3,zc4];
```

```
f=line(v1,v2,v3);

v1=[xc4,xc1];
v2=[yc4,yc1];
v3=[zc4,zc1];
f=line(v1,v2,v3);
 %-------------------------------------------------------
--
%Top Plane

xc5=xc+dx/2;
yc5=yc+dy/2;
zc5=zc+dz/2;

xc6=xc-dx/2;
yc6=yc+dy/2;
zc6=zc+dz/2;

xc7=xc-dx/2;
yc7=yc-dy/2;
zc7=zc+dz/2;

xc8=xc+dx/2;
yc8=yc-dy/2;
zc8=zc+dz/2;

v1=[xc5,xc6];
v2=[yc5,yc6];
v3=[zc5,zc6];
f=line(v1,v2,v3);

v1=[xc6,xc7];
v2=[yc6,yc7];
v3=[zc6,zc7];
f=line(v1,v2,v3);

v1=[xc7,xc8];
v2=[yc7,yc8];
v3=[zc7,zc8];
f=line(v1,v2,v3);

v1=[xc8,xc5];
v2=[yc8,yc5];
v3=[zc8,zc5];
f=line(v1,v2,v3);
%--------------------------------------------------------
-
% Side Planes

v1=[xc1,xc5];
v2=[yc1,yc5];
```

```
v3=[zc1,zc5];
f=line(v1,v2,v3);

v1=[xc2,xc6];
v2=[yc2,yc6];
v3=[zc2,zc6];
f=line(v1,v2,v3);

v1=[xc3,xc7];
v2=[yc3,yc7];
v3=[zc3,zc7];
f=line(v1,v2,v3);

v1=[xc4,xc8];
v2=[yc4,yc8];
v3=[zc4,zc8];
f=line(v1,v2,v3);

%---------------------------------------------------------
-
        end
    end
end
toc
```

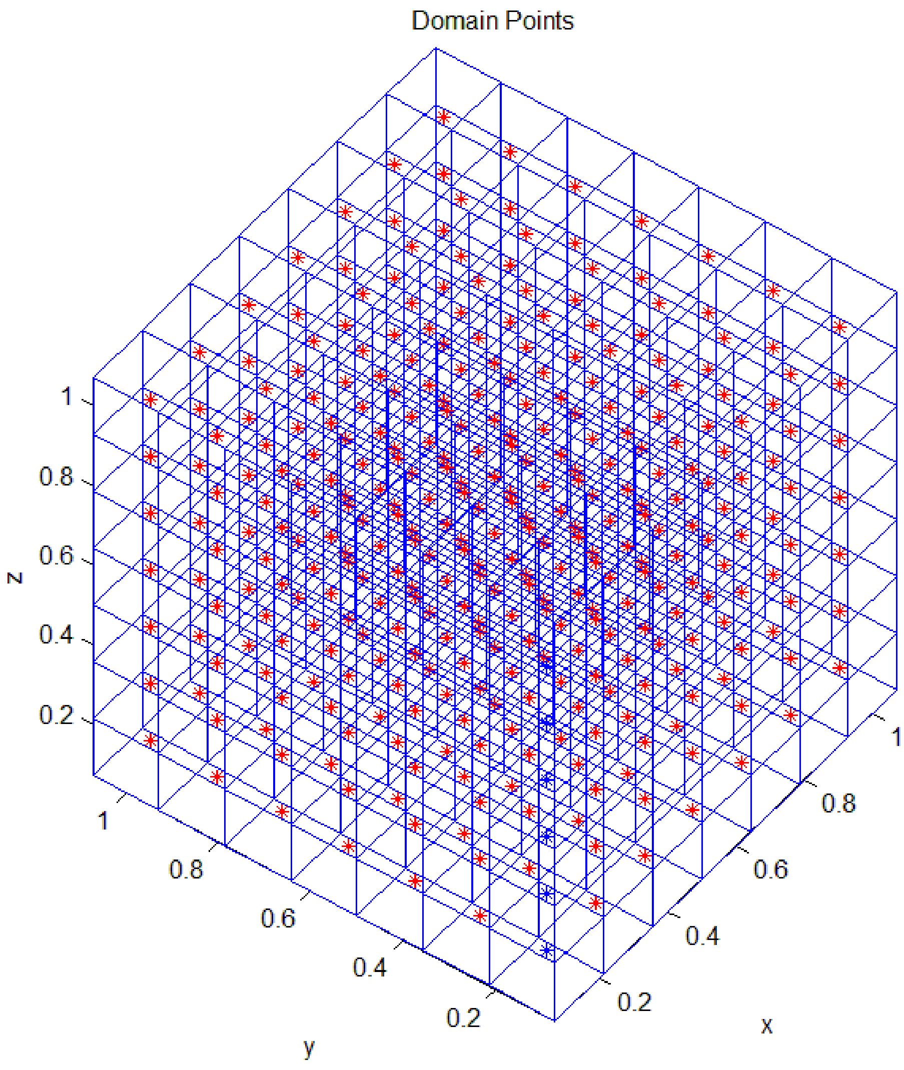


Figure 27 This example represents a box case where the red points represent the grid points that are linked to the differential equations.

During your research you will be assigned to research projects that dont have the usual dimensions as the

```
clc
clear
M=8;
N=8;
L=8;
LX=1;
LY=1;
LZ=1;
dx=LX/M;
dy=LY/N;
dz=LZ/L;
for i=1:M;
for j=1:N;
for k=1:L;
x(i,j,k)=i*dx;
y0(i,j,k)=j*dy;
y(i,j,k)=(x(i,j,k))^2+y0(i,j,k);
z(i,j,k)=k*dz;
end
end
end
for i=1:M;
for j=1:N;
for k=1:L;
plot3(x(i,j,k),y(i,j,k),z(i,j,k),'*r')
hold on
grid on
xlabel('x')
ylabel('y')
zlabel('z')
end
end
end
for k=1:L;
for i=1:M;
for j=1:N;
        if (i<M)
        xx=[ x(i+1,j,k) x(i,j,k) ];
        yy=[ y(i+1,j,k) y(i,j,k) ];
        zz=[ z(i,j,k) z(i,j,k) ];
        f=line(xx,yy,zz);
        elseif (i==M)
        xx=[ x(i,j,k) x(i,j,k) ];
        yy=[ y(i,j,k) y(i,j,k) ];
        zz=[ z(i,j,k) z(i,j,k)];
        f=line(xx,yy,zz);
```

```
end
end
end
end
for k=1:L;
for i=1:M;
for j=1:N;
        if (j<N)
        xxx=[ x(i,j+1,k) x(i,j,k) ];
        yyy=[ y(i,j+1,k) y(i,j,k) ];
        zzz=[ z(i,j,k) z(i,j,k) ];
        f=line(xxx,yyy,zzz);
        elseif (j==N)
        xxx=[ x(i,j,k) x(i,j,k) ];
        yyy=[ y(i,j,k) y(i,j,k) ];
        zzz=[ z(i,j,k) z(i,j,k) ];
        f=line(xxx,yyy,zzz);
end
end
end
end
for k=1:L;
for i=1:M;
for j=1:N;
if (k<L)
        xxxx=[ x(i,j,k) x(i,j,k) ];
        yyyy=[ y(i,j,k) y(i,j,k) ];
        zzzz=[ z(i,j,k+1) z(i,j,k) ];
        f=line(xxxx,yyyy,zzzz);
elseif (k==L)
xxxx=[ x(i,j,k) x(i,j,k) ];
yyyy=[ y(i,j,k) y(i,j,k) ];
zzzz=[ z(i,j,k) z(i,j,k) ];
f=line(xxxx,yyyy,zzzz);
end
end
end
end
```

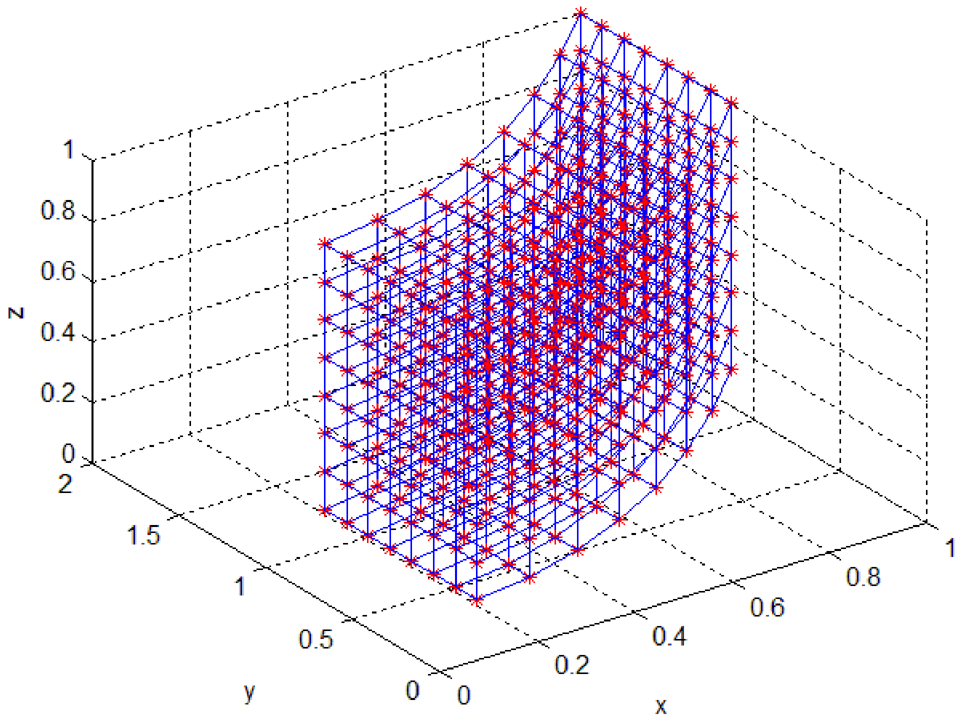


Figure 28 The Code generated mesh.

4-4 Working with Cylindrical Mesh:

This some application that has use in many engineering applications.

4-4-1 Example 1

```
clc
clear
box on
[x,y,z]=cylinder(1);
[x1,y1,z1]=cylinder(2);
xx=x1;
yy=y1;
zz=z1;
plot(x,y,'*r',xx,yy,'*b');
c=0;
for i=1:40;
    c=c+1;
v1=[x(c) xx(c)];
v2=[y(c) yy(c)];
v12=[ z(c)  zz(c) ];
f=line(v1,v2,v12);
v3=[x(c) x(c+1)];
v4=[y(c) y(c+1)];
v34=[ 0 0 ];
f=line(v3,v4,v34);
v5=[xx(c) xx(c+1)];
v6=[yy(c) yy(c+1)];
v56=[ 0 0 ];
f=line(v5,v6,v56);
end
grid on
```

4-4-2 Example 2

```
clc
clear

M=20;
N=2*M;
L=4;
r=1;
r1=1.5;
r2=2;
T=-1;
TT=1;
for j=1:L;
thet(1)=0;
T=T+TT;
for i=1:M+1;
    dt=360/M;
    thet(i+1)=thet(i)+dt;
    theta(i)=(pi/180)*thet(i+1);
    x(i,j)=r*cos(theta(i));
    y(i,j)=r*sin(theta(i));
    z(i,j)=T;
end
thet(1)=0;
for i=1:N+1;
    dt=360/N;
    thet(i+1)=thet(i)+dt;
    theta(i)=(pi/180)*thet(i+1);
    x1(i,j)=r1*cos(theta(i));
    y1(i,j)=r1*sin(theta(i));
    z1(i,j)=T+0.5*TT;
end
thet(1)=0;
for i=1:M+1;
    dt=360/M;
    thet(i+1)=thet(i)+dt;
    theta(i)=(pi/180)*thet(i+1);
    x2(i,j)=r2*cos(theta(i));
    y2(i,j)=r2*sin(theta(i));
    z2(i,j)=T;
end
end

plot(x,y,'-',x2,y2,'-');
hold on

ff=0;
for j=1:L-1;
for i=1:N;
    ff=ff+1;
    a(i)=rem(ff,2);
    if (a(i)==1)
    plot3(x1(i,j),y1(i,j),z1(:,j),'-*r');
    hold on
    end
end
end

grid on
axis equal
xlabel('x')
ylabel('y')
zlabel('z')
title('Cylindrical Mesh')

for j=1:L;
for i=1:M
    v1=[x(i+1,j),x(i,j)];
    v2=[y(i+1,j),y(i,j)];
    v3=[z(i+1,j),z(i,j)];
    f=line(v1,v2,v3);
end
end

for j=1:L-1;
for i=1:N
    v1=[x1(i+1,j),x1(i,j)];
    v2=[y1(i+1,j),y1(i,j)];
    v3=[z1(i+1,j),z1(i,j)];
    f=line(v1,v2,v3);
end
end

for j=1:L;
for i=1:M
    v1=[x2(i+1,j),x2(i,j)];
    v2=[y2(i+1,j),y2(i,j)];
    v3=[z2(i+1,j),z2(i,j)];
    f=line(v1,v2,v3);
end
end

for j=1:L;
for i=1:M
    v1=[x2(i,j),x(i,j)];
    v2=[y2(i,j),y(i,j)];
    v3=[z2(i,j),z(i,j)];
    f=line(v1,v2,v3);
end
end

for j=1:L-1;
for i=1:M
    v1=[x(i,j),x(i,j)];
    v2=[y(i,j),y(i,j)];
    v3=[z(i,j+1),z(i,j)];
    f=line(v1,v2,v3);
end
end

for j=1:L-1;
for i=1:M
    v1=[x2(i,j),x2(i,j)];
    v2=[y2(i,j),y2(i,j)];
    v3=[z2(i,j+1),z2(i,j)];
    f=line(v1,v2,v3);
end
end
```

4-5 Recommended Reading List

Helpful References in the Field of Mesh Generation .The classification is based on the level of the researcher**.**

For Beginners:

1-Farrashkhalvat, M. Miles J.P. 2003 Basic Structured Grid Generation: With an introduction to unstructured grid generation, A Butterworth-Heinemann Title.

For Intermediates:

1-Thompson, J.F. Soni, B.K. Weatherill, N.P. 1998 Handbook of Grid Generation, CRC Press.

2-Liseikin, V.D. 2006 A Computational Differential Geometry Approach to Grid Generation (Scientific Computation),Springer.

For Professionals:

This is where the researcher’s part comes in finding the required journal.

Chapter 5

Guide Lines for CFD PhD Students

5-1 Introduction:

VERY IMPORTANT Guide Lines for PhD Students to What Research Area to Choose when Thinking of Working in the Field of Computational Fluid Dynamics. Once you get through these points you can get a clear picture in which area you want to work in.

A-You can choose to work either on ready Computational Fluid Dynamics packages:

These software packages are ANSYS-CFX, ANSYS-FLUENT, STAR-CCM+, ANSYS Mechanical, ANSYS PolyFlow, ANSYS Offshore, ANSYS Explicit, ... etc. It all depends on what problem will you be trying to solve during your PhD. It is a requirement for the supervisor to have knowledge about the software the student is going to use. This is because from there he can verify if the software can solve the problem and that the computer hardware available at the faculty is enough to conduct calculations that achieve the projects goals. Off course your supervisor at some point can't give you any assistance because these software's are complex, not easy to learn and are developing very fast. It won't be helpful in going to online forums and the user's manual provided by the company. These materials are made for people who have taken training courses and need to refresh what they have done before. At that point you would require a training course, these courses are expensive and are provided by the companies that make these kinds of software. Therefore in addition to paying the university fees it is also required to have a budget for the training courses. It is required to have a research group to work with. Because these software packages are formed of several software's each for a specified purpose. This area is preferable for engineering background students, because the researcher will be using the interface tools provided with the software. These interface tools will help him in generating his meshed geometry, to choose the numerical solution model, to choose what kind of flow case is studied flow (Subsonic, Sonic, Supersonic, Ultrasonic) to select the.

B-You can choose to work on Computational Fluid Dynamics models development:

To work in the field of models development then you would require a strong back ground in Programming, Numerical Analysis, Linear Algebra, Algorithms, Computational PDEs including, Code Parallelization, Statistics and Probability. Without this foundation the student would not be able to get anywhere, because the complexity of the problems he will face will be too difficult to solve in the PhD required period. In addition to the need to have a team of researchers in the group to help you out when you do get stuck. If a bug doesn't get solved no way you can get to the upgrade panel with results. This area is preferable to mathematicians and physicists and computer science researchers, not at all recommended to engineering students. This kind of project can be implemented on computer resources and software that are less costly than using the commercial codes. You can even acquire the required skills before you start your PhD through using OpenFoam while software's like ANSYS and STAR-CCM+ once you have no access to the software you won't be able to develop your skills using them.

C-You can choose to work on the development of Computational Fluid Dynamics packages:

This area would require the student to be a mathematician who has good knowledge in programming or a computer science scientist. This kind of project would be directly involved with the software company.

5-2 Initial Steps for the Researchers to Follow:

CFD projects are big projects for research and PhD students. Research projects require problem solving skills which you either have before or you need to gain with time. These kind of projects may take the researcher at least six months to build and calibrate the simulations to get the required results. Usually these projects are split into different stages to build the numerical model using the interface tools provided by the software. The majority of new researchers lack the back ground needed but such a project needs:

1-guidance provided to you through the group you are working with or through your supervisor.

2-If your research group cannot assist you then you can ask a researcher at your university, if not ask some at another university near where you live, if not ask someone on the net.

3-You can bypass your problems by asking your boss to back you up for a training course with ANSYS CFX.

4-You can contact ANSYS for any enquirers, if your research institute has the software and is currently using it, they will be happy to answer you quires I have contacted them several times.

5-3 Data Analysis

Hope these steps are useful.

Stage 1:

Make a simple drawing of your studied case. Then specify your inflow and out flow regions. Imagine it just like a Lego building blocks. As a first step make a very simple simulation with just an inflow velocity and an opening as an outlet. Then once you clarify that the simulation runs and results can be produced then comes the addition of new variables. These values are usually standard such as 1.29 kg/m^3 density of air 300 K ambient temperature 1 atm for pressure. Then try to see in publications if something similar has been done and try to extract from it values that you can use for your case. If the case is in your experimental lab that can be a straight forward job. After a calculation run had been done. Using the data visualization tools start plotting the data. Any research topic has several parameters that need to be studied to extract from them the occurring patterns. The supervisor would have given you a hint to them or would have asked you to extract them from published material. You will start relating these variables to geometrical constraints or to thermodynamic variables. At the first instance save all the captured plots for the different studied variables. Make print outs of these plots hang them on the wall and try relating the different variables with different plots. After a while you start seeing that for example gravitational effects are dominant or bounacy effects temperature etc. Next write all of them in a list. Now that you have the list.

Stage 2:

From the first stage use the same list make your plots in accurate form giving the axis titles and plots titles. At this stage start trying to make your simulation mimic your studied case as an example if you were in real life getting water to boil at 100 deg try to make it achieve the same value in the simulation. This is done by playing around with the existing variables in your simulation.

once you start getting the feel of the effects of the different variables that's when you can try to make your simulation look as real as possible.

Stage 3:

This represents the final stage of analysis process. At this stage you will choose whether you will need to add any extra stuff or not.