Ecole polytechnique Nantes **Ecole Central Nantes**

Application and development of inverse theory to Shock Tube problem

by

Mohd Afeef BADRI

Guided by:

Dr. Yann FAVENNEC & Dr. Ahmed Ould El MOCTAR

A thesis submitted in partial fulfillment for the
degree of Master Sciences in Thermal Science and Energy
(Mécaniques Appliquées Spécialité Thermique-Energétique)

Jury:

President: Dr. Ahmed Ould El MOCTAR	*LTN,Ecole Polytecnique Nantes*
Dr. Bertrand GARNIER	*LTN,Ecole Polytecnique Nantes*
Dr. Yann FAVENNEC	*LTN,Ecole Polytecnique Nantes*

July 2015

"If we knew what it was we were doing, it would not be called research, would it?"

Albert Einstein

Abstract

Dr. Yann FAVENNEC & Dr. Ahmed Ould El MOCTAR
Department of Thermique Energetique

Master of Science

by Mohd Afeef BADRI

An inverse fluid dynamics problem has been investigated for a non linear compressible flow phenomenon in a shock tube. There exists no a prior knowledge of initial conditions for the Shock Tube, and these conditions have been retrieved using inverse problem theory. This problem being categorized as parameter estimation problem in inverse computational science, is solve using only one experimental measurement inside the shock tube. Many inverse algorithms based on different optimization techniques have been proposed to solve the in hand problem. A sensitivity analysis of these different methods has been presented. In order to enhance the performance of the optimizers for the concerned problem, two new hybrid optimizers, PSO1CG2 and PSO2CG1 have been developed and tested.

Acknowledgements

It would be an honor to express my deep sense of reverence and gratitude to **Dr. Yann FAVENNEC**, my guru, for introducing me to fascinating world of inverse problems in general and in research work in particular. His methodology of giving absolute freedom at par with excellent work environment and exposure helped me a lot in acquiring knowledge in the field of code development in inverse computations. I would equally like to thank my co mentor **Dr. Ahmed Ould El MOCTAR** for encouraging me both at educational and personal levels. His advice's and experience in research helped me glide smoothly throughout the span of my thesis. I am deeply indebted of both my supervisors for guiding me through all the challenges, both academic and personal.

I would take this opportunity to acknowledge help and cooperation provided by many people in planning and execution of this research. I am thankful to Mr. Manoj Joishi and Miss Smriti Dhingra with whom I worked to develop the initial EULER solver. I would also thank my friends Mr. Nitin Kukreja, Mr. Shailesh B. G, Mr. Anurag Singh, Mr. Geetesh Waghela and Miss Ketaki Mishra for always being there in time of any academic or personal help.

Finally I will always remain in debt to my parents and my brother for limitless love, care and support that they have always provided me with.

Contents

List of Figures

List of Tables

Abbreviations

1D	**1**(one) **D**imensional
2D	**2**(two) **D**imensional
AKA	**A**lso **K**nown **A**s
BFGS	**B**royden **F**letcher **G**oldfrab **S**hanno
CFD	**C**omputational **F**luid **D**ynamics
CFL	**C**ourant **F**riedrichs **L**ewy
CG	**C**onjugate **G**radient
DFO	**D**erivative **F**ree **O**ptimization
DFP	**D**avidson **F**letcjer **P**owell
FDM	**F**inite **D**ifference **M**ethod
FVM	**F**inite **V**olume **M**ethod
GN	**G**auss **N**ewton
GUI	**G**raphical **U**ser **I**nterface
ISTP	**I**nverse **S**hock **T**ube **P**roblem
NM	**N**edler **M**ead
PDE	**P**artial **D**iffrential **E**quations
PSO	**P**article **S**warm **O**ptimization
RK	**R**unge **K**utta
SG	**S**teepest **G**radient

Symbols

a	speed of sound	m/sec
a_l	speed of sound in left section	m/sec
a_r	speed of sound in right section	m/sec
a_1	speed of sound in region 1	m/sec
a_2	speed of sound in region 2	m/sec
a_e	speed of sound in expansion fan region	m/sec
C	characteristic line	m
C^-	left running characteristic line	m
C^+	right running characteristic line	m
$C1$	coefficient in PSO	
$C2$	coefficient in PSO	
d^k	displacement direction	m
d^0	initial displacement direction	m
E	total energy	J
E	Error vector	
F	flux vector	
$F_{\perp J}$	interface flux through interface	
$F_{\perp L}$	interface flux towards left cell	
$F_{\perp R}$	interface flux towards right cell	
G_{best}	global best position	
H	total enthalpy	J
J	jacobian matrix	
j	cost function	
$j(\psi)$	cost function	
$\mathcal{J}(Y_{mo})$	global cost function	
l	length	m
L_2	euclidean	
M	mach number	
M_s	mach number of normal shock	

M_{max}	maximum mach number	
n	unit normal	
p	pressure	N/m^2
p_l	left section pressure	N/m^2
p_{mo}	forward model pressure	N/m^2
p_{best}	particles best position	N/m^2
p_r	right section pressure	N/m^2
p_1	pressure in section 1	N/m^2
p_2	pressure in section 2	N/m^2
p_l	pressure in expansion fan	N/m^2
R^n	region space	
$R1$	random coefficient in PSO	
$R2$	random coefficient in PSO	
$\mathbb{R}$	real Numbers	
S	Sensitivity matrix	
T	temperature	K
T_l	left section temperature	K
T_r	right section temperature	K
T_1	temperature in section 1	K
T_2	temperature in section 2	K
T_e	temperature in expansion fan	K
t	time	sec
u	flow velocity	m/sec
u_l	left section flow velocity	m/sec
u_{mo}	forward model flow velocity	m/sec
u_r	right section flow velocity	m/sec
u_1	flow velocity in section 1	m/sec
u_2	flow velocity in section 2	m/sec
u_s	velocity of normal shock wave	m/sec
u_s	velocity in expansion fan	m/sec
$u_\perp$	velocity at the face	m/sec
U	state vector	
U^t	state vector at time level t	
$\overline{U}_i$	cell averaged states	
V_x^p	velocity vector of particles	
x_0	diaphragm location	m
x_{data}	sensor location	m
x_x^p	position vector of particles	

Y	measurements vector	
Y_{mo}	forward model values vector	
Y'_{mo}	derivative in model space	
α	step size	
ε	Venkatakrishnan coefficient	
ρ	density	kg/m^3
ρ_l	density in left section	kg/m^3
ρ_r	density in right section	kg/m^3
ρ_1	density in section 1	kg/m^3
ρ_2	density in section 2	kg/m^3
ρ_e	density in expansion fan	kg/m^3
β_k	coefficient in RK method	
β	coefficient in CG method	
ζ_i	trial functions	
$\forall$	for all	
λ	eigen value	
ν	eigen vector	
ψ	parameter vector	
χ	simplex	
χ	initial simplex	
$\overline{\psi}$	minimum parameter vector	
ψ^0	first guess parameter vector	
ψ^R	reflected parameter vector	
ψ^C	contracted parameter vector	
ψ^E	Expanded parameter vector	
ψ^m	mid point parameter vector	
ψ^u	best point parameter vector	
ψ^v	good parameter vector	
ψ^w	worst parameter vector	
Ω	finite volume	m^3
δt	perturbation in time	sec
ϕ_j	reconstruction limiting coefficient	
ω	coefficient in PSO	
$\triangle\, U_i$	state gradient at cell	$/m$
$\triangle\, S_j$	interface surface area	m^2
$\triangle\, S_j$	interface surface area	m^2
$\triangle\, x$	cell size	m

$\triangle\, j(\psi)$	cost function gradient
$\frac{\partial}{\partial x}$	partial derivative in x direction
$\frac{\partial}{\partial y}$	partial derivative in y direction
$\frac{\partial}{\partial t}$	partial derivative in time
$\frac{d}{dt}$	total derivative in time

Chapter 1

Introduction

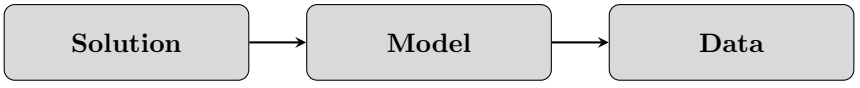
Outline

This work involves solving inverse fluid dynamics problem for a non linear flow phenomenon inside of a shock tube. Section 1.1 introduces the concept of inverse problem and section 1.3 introduces more specific class of *inverse fluid dynamic* problem. In inverse problems a strong background of forward modeling is needed, in light of this, section 1.2 introduces computational fluid dynamics as tool for forward modeling. Finally in section 1.4 inverse shock tube problem is introduced as a real world application of inverse problem theory.

1.1 Introduction to Inverse problems

"Inverse problem" is the branch of science that deals with calculating from effect (observations or experimental data) the cause that led to the effect. The name *inverse problem* arises from the fact that it starts with results and then processes the cause. This is inverse of any forward modeling (refer Figure 1.1), which begins with the cause and then processes the results. In general outline inverse problems are used to convert the observed measurements (experimental data) into information about a physical object or a system.

(a) Forward problem



(b) Inverse problem

FIGURE 1.1: Difference between a Forward and Inverse Model

The term *"inverse problems"* has been steadily and surely gaining admiration and popularity in modern science since the mid 20th century. Inverse problems find their application in several scientific fields, including, thermal sciences, fluid Dynamics, medical imaging, Nuclear Sciences, astrophysics, image processing, sub-surface prospecting, and geophysics [1].

Generally inverse problems are more difficult than the simply reversing the forward model. To cite an example, though Earth's gravitational field is governed by Newtons law of gravitation, inverse problem involved in finding sub-surface structure from perturbations in gravitational field is an extremely challenging. Well built inverse problem solution requires specially tailored algorithms that can tolerate errors in measured data and predict the results (cause) correctly. Numerical simulation for any forward problem may or may not be well posed whilst inverse problem is always *ill posed*.

1.2 Introduction to CFD

Experimental study, Theoretical study and Numerical Study (computational dynamics) are the three possible ways for studying a typical engineering problem. These three work hand in hand for engineering problem solutions and facts regarding them can be realized by Figure 1.2. Computational Fluid Dynamics (CFD) is incorporation of Numerical study, science for studying and solving problems in engineering mechanics, it involves use of computers to solve or study transport phenomena problems using numerical techniques. Computational Fluid Dynamics (CFD) is the science of predicting fluid flow, heat transfer, mass transfer, chemical reactions, and related phenomena (transport phenomena) by solving the mathematical equations which govern these processes using numerical techniques.

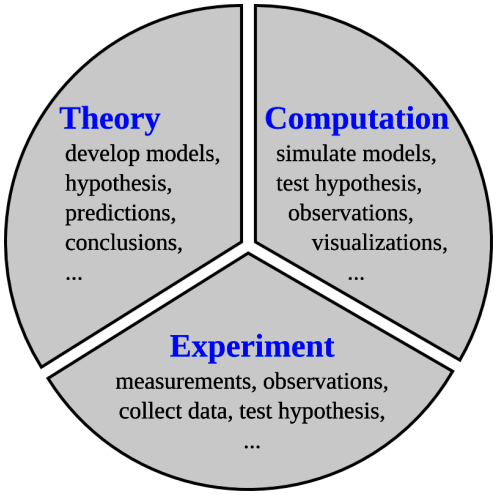


FIGURE 1.2: Theory, Computation and Experiments

Computational Fluid Mechanics enables us to go much beyond what can be done analytically solving fluid flow or heat transfer problems. We as Engineers can easily predict how flow will behave in a pipe, or how it would behave for a sphere under creeping flow conditions and one can easily characterize the pressure drop, drag coefficient or lift generated, for such problems. But when it comes to real world problems we face difficulty in solving such problems analytically, one such engineering problem is: solution of a shock tube problem that is governed by complicated non-linear hyperbolic partial differential equation. *Solving shock tube problem by a forward model is itself challenging and using inverse problem theory over it adds to the magnitude of challenge.*

1.3 Introduction to inverse fluid dynamics

Inverse problems may now have been widely studied and explored in many fields of science but it still remains in its infancy when it comes to fluid mechanics. Inverse fluid dynamics is a sub branch of inverse problems that deals with inverse solutions for flow phenomena. Fluid dynamic design problems generally come with a computationally expensive requirement, inverse analysis of such problems would add to the complexity and computational cost of such problems.

Today there exist many methods to solve inverse problems, there are several articles [2–4] and books [5–8] that have been published upto date on this subject. However there are only few articles that cover the area inverse fluid dynamic, and those that do, often cover it as an aside to heat transfer problems. Study of this limited resourced archive suggests a simple yet effective method to solve inverse fluid dynamics problems, this method involves iterating the numerical simulation until the result matches the observed set of data or is atleast close to it, often demanding many iteration henceforth increasing computational cost. To cite examples that support this statement are papers by Liu et al [3] and Knight et al [9], where the former deals with convection problem and the latter deals with prediction of velocity and temperature for a jet in crosswind.

1.4 Introduction to Inverse problem in a shock tube

A shock tube is a device commonly used for detonation, supersonic or hyper-sonic testing. Flow inside the tube is transient in nature and highly non-linear, governed by non-liner hyperbolic PDEs the Euler equations. Real life flow can be simulated by solving the Eulers equations by any computational technique (*forward problem*). To solve the forward problem initial conditions of primitive variables (p, ρ, u) at $t = 0$ serve as input to the forward problem. To introduce the inverse shock tube problem (ISTP), the initial conditions in the tube are no more available and need to be formulated by *inverse problem theory*. Experimental data at time $t > 0$ is served as input to the inverse problem.

This ISTP that concerns initial condition estimation can find real world applications, as many shock tubes that run today still treat the initial conditions as missing data [10] and these initial conditions are approximated by some or the other method. This fact can be realized by initiation of flow/detonation in the shock tube that takes place by rupturing of a diaphragm and the conditions at which this rupture occurs is unknown. Knowing the initial conditions by inverse computation will help analyzing the flow/detonation in a better manner and will also bring improvement concerning further use of the shock tube.

In the next chapter

Now that concept of inverse problem, forward modeling and a real world application inverse problem on shock tubes is introduced. There is a need to understand the physic and formulate the mathematics for flow inside the shock tube this is dealt within the following chapter.

Chapter 2

Shock Tube

Outline

Since inverse problems demand a strong background in the physical phenomenon involved, this chapter takes into consideration, understanding of the shock tube functioning, physics and mathematics. Sections 2.1 and 2.2 highlight some basic facts about the shock tube and introduces some basic terminology used in shock tube flow analysis. To develop more interest in shock tubes and show its importance in the scientific community, history of shock tubes is traced out in section 2.3. To understand the flow occurring in the shock tube, qualitative and mathematical reasoning of the flow inside the tube are presented in sections 2.4 and 2.5. Analytical and numerical solutions of the flow are presented in sections 2.6 and 2.7 respectively.

2.1 Introduction

A simple yet very useful apparatus to produce supersonic flow, ever since its invention till present years interest has been revived in this apparatus which was first developed around 1900 [11]. This apparatus, known today as a shock tube or Sods Shock tube, is one of the simplest ways to produce a supersonic flow. Interest in studying the Sods shock tube is threefold. 1) From a fundamental point of view, shock tubes offers an interesting framework to introduce some basic notions about nonlinear hyperbolic systems of partial differential equations (PDEs). 2) From a numerical point of view, this problem constitutes, since analytical solution is available, an inevitable and difficult test case for any numerical method dealing with noncontinuous solutions. 3) Finally, there is a practical interest, since this model is used to describe real shock tube experimental devices.

2.2 Physical description of Shock tube

A shock tube consists of a simple pipe which may be close or open ended. A diaphragm divides the tube into two sections, initially containing fluid (generally gases) at different pressure conditions. The low pressure section is termed as *working section* and the high pressure section as *driven section*. Normally working section length can exceed driven section lengths up to ten times or more. Rupturing the diaphragm ensures flow characterized by discontinuties i.e shocks, with expansion wave moving towards high pressure section and a normal shock moving towards low pressure section; these waves are separated by a region of constant velocity flow (Region (1) & (2)), refer Figure 2.1 for pictorial representation of this phenomenon. Conventionally shock tubes use a constant cross sectional area with air being the medium of driven and working section.

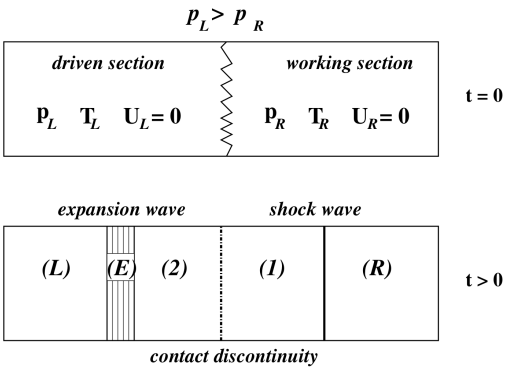


FIGURE 2.1: pictorial representation of initial configuration in the shock tube at $t = 0$ and waves propagation in the tube after the diaphragm breakdown $t > 0$.

2.3 History of Shock tube

The use of Shock tube has varied over time since the time of its invention, and it may be interesting to trace back its history. First shock tube was developed due to growth of interest in study of propagation speeds of flame fronts and detonation waves. This led to construction of the first shock tube by Vieille [11] in France (1899). He studied detonating with shock waves in a 22 mm diameter, 20 ft long tube. In his experimental work, Vieille could measure shock velocities that were twice the speed of sound in air. Despite of this breakthrough, shock tube was neglected for the nearly next forty years, until W. Payman and W. C. F. Shepherd [12] published the classic paper in which shock tube was investigated in details. This work found its application in problem of safety in mines and shock tube was once again reason of interest in the field of detonation. Contrary to techniques used today in a shock tube thin sheet copper was used as a diaphragms and rupturing was initiated by a sudden rise in pressure across the diaphragm. Thus, back then it was impossible to know the pressure difference at which the diaphragm ruptured. The theory driving the flow inside shock tube was verified first time during the experimental study on use of shock tubes in aerodynamics by B.D. Henshall [10].

In 1949, it was realized that quasi-steady region possessed by the shock tube (region (1) & (2), Figure 2.1) can be used to study subsonic and supersonic flows, making shock tube a conveniently simple aerodynamic wind tunnel. Shock tubes were also frequently used for studying transient flows. Using slotted walls with shock tubes made it possible to conveniently study transonic phenomena inside the shock tube. In 1952 at Princeton university shock tubes were used extensively to experiment flow in Mach number range of 0.86 to 1.16 [13].

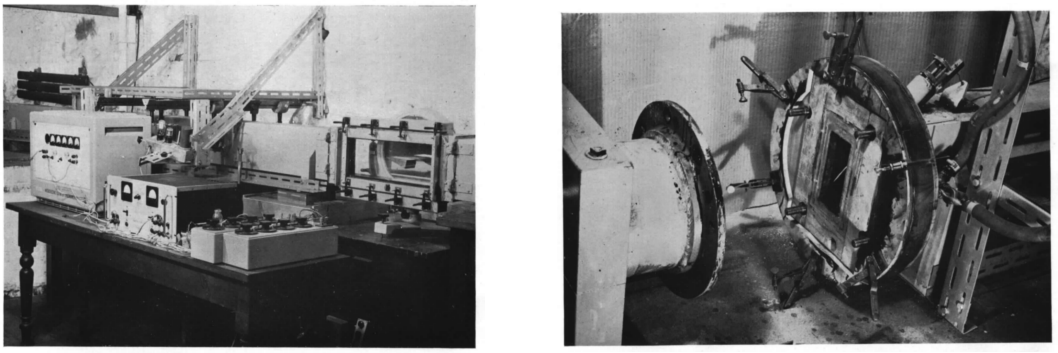


FIGURE 2.2: Historic shock tube in Bristol University, Aeronautical Extension Laboratory

Theoretically infinite pressure difference across the diaphragm would produce finite Mach number of $M_{max} = 1.73$ (refer Table 2.1) for more accurate and complete data refer [14].

The limitation on Mach number M was overcome by using a divergent duct in the low pressure region and supersonic Mach number of $M = 4.2$ was realized by Hertzberg [15] which was later followed by hypersonic flow being produced [16].

TABLE 2.1: Mach numbers for diffrent initial pressure diffrence in a shock tube (P_l/P_r)

p_l/p_r	Shock Strength (Static Pressure Ratio)	Mach number behind the shock Region(1)	Mach number Behind contact discontinuity Region (2)
1.0	1.0	0	0
10.38	2.89	.71	1
41.38	4.83	1.00	1.80
100	6.40	1.15	2.41
∞	44.14	1.73	∞

The argument presented above made it clear that, historically shock tube has always stayed a hot topic of research and interest. Till date still shock tubes are intensively being studied, [17–20] are some of the recently published articles in this field. Shock tubes are being used in many leading labs as an apparatus to study detonation, Transonic, supersonic and Hypersonic flows (refer to Figure 2.2 , 2.3 for some shock tubes).

FIGURE 2.3: New modern shock tube at JP Aerospace, USA

2.4 Qualitative analysis of flow inside shock tube

Diaphragm rupturing in shock tube brings about ideal unsteady flow to occur inside the tube and the flow will be treated mathematically in the topic to follow; however, at this stage, a qualitative survey of the process may be helpful.

Consider a conventional shock tube (shown in Figure 2.1) with closed ends to be maintained at an idealized pressure ratio P_l/P_r occurring across the diaphragm. Rupturing of diaphragm ensues a unsteady flow phenomena inside the tube. Expansion waves initially originating at the diaphragm propagate towards the high pressure region the driven section. A normal shock moves towards the low pressure region with a supersonic velocity U. This results in compression of fluid particles in the low pressure region hence they acquire a uniform velocity $u(u < U)$ in the direction of normal shock. This velocity u may or may not be supersonic depending on initial pressure difference P_l/P_r.

Consider now the driven section. The expansion fan in this area has the front side of the wave traveling with local velocity of sound a_0, whilst the direction and the velocity of the opposite end of it depends in initial pressure difference P_l/P_r. Due to movement of expansion fan towards left fluid particles that are initially at rest are set into motion towards right side, that is the same direction of the shock wave. These motions in different sections of the tube give rise to the unsteady flow with, expansion fan moving left and normal shock moving right, separated by constant velocity regime. If a close ended shock tube is used the shock hits the walls and reflects back traveling as a reflected shock. This phenomenon is clearly shown by Figure 2.4.

Consequences of the above mentioned flow is that compressed region behind the shock expe-

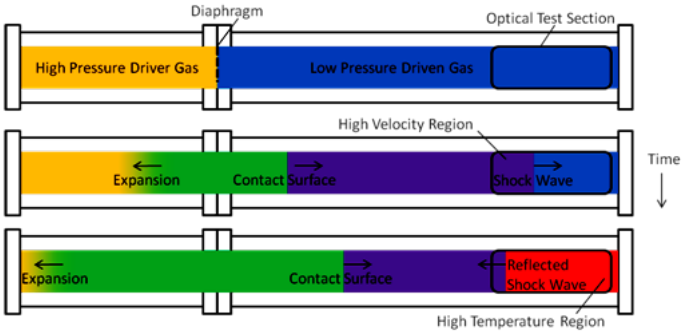


FIGURE 2.4: Working of a shock tube

riences decrease in density ρ and increase in temperature T than the initial density and temperature; and for the region behind expansion fans i.e expanded region, its density ρ increases and temperature T decreases than the initial density and temperature. Hence discontinuity of density and temperature is sensed at the point were diaphragm was ruptured. Presence of this discontinuity leads to discontinuity in speed of sound $a = \sqrt{\gamma \Re T}$ henceforth leading to different Mach numbers across the discontinuity.

Any point at any distance x from diaphragm. Flow record past this point would exhibit:

- A normal shock wave moving towards right, with velocity U,

- a section of 'quasi-steady' uniform flow to the right moving with a velocity u,
- a density and temperature discontinuity moving towards right,
- end of expansion fan moving towards right.

This phenomenon can be noted in the Figure 2.4. Generally, there exists very good agreement in between theory and actual shock tube flow, especially for initial pressure differences of less than 100 : 1 [10]. It should be noted that the flow behind the contact discontinuity may not be considered as true discontinuity, for the flow across it only density and temperature posses discontinuities but velocity and pressure are still continuous.

2.5 Mathematics Involved in shock tube

Flow inside the shock tube is governed by a non linear hyperbolic PDEs known as Euler Equation. In order to simplify mathematics involved, consider the following assumptions :

- the tube is of infinite length (avoiding reflection at the ends);
- the flow inside the shock tube is considered non viscous and compressible;
- the diaphragm is removed completely from the tube at t = 0.

Under these hypotheses the flow inside the tube is governed by one-dimensional Euler PDEs that are non-linear and Hyperbolic in nature (see, [1]). These equations are the continuity equation, the Momentum equation and the energy equation:

$$\text{Continuity} \quad \frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0 \tag{2.1}$$

$$\text{Momentum} \quad \frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} + \frac{\partial p}{\partial x} = 0 \tag{2.2}$$

$$\text{Energy} \quad \frac{\partial E}{\partial t} + \frac{\partial E u}{\partial x} + \frac{\partial p u}{\partial x} = 0 \tag{2.3}$$

These can be written as

$$\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \rho u \\ \rho u^2 + p \\ (E+p)u \end{pmatrix} = 0 \tag{2.4}$$

here ρ the density of fluid is related with the total energy E :

$$E = \frac{p}{\gamma - 1} + \frac{\rho u^2}{2} \tag{2.5}$$

The equations can be simplified by writing it in state vector U and Flux vector $F(U)$ as

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} = 0 \tag{2.6}$$

with

$$U = \begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ (E+p)u \end{pmatrix} \tag{2.7}$$

This equation 2.6 is the *conserved form* equation of the flow. To close the set of equations we need the additional ideal gas equation

$$p = \rho \Re T \tag{2.8}$$

Constants $\Re$ and γ involved in equations 2.8 and 2.5, are thermodynamic properties of the chosen fluid. Also assume a, M and H are the local speed of sound, the Mach number and the Enthalpy of the fluid:

$$a = \sqrt{\gamma \Re T} = \sqrt{\gamma \frac{p}{\rho}}; \quad M = \frac{u}{a}, \tag{2.9}$$

$$H = \frac{E+p}{\rho} = \frac{a^2}{\gamma - 1} + \frac{u^2}{2} \tag{2.10}$$

Assuming the diaphragm to be standing at x_0 at time $t = 0$ the initial conditions for the tube may be given as

$$U(x,0) = \begin{cases} (\rho_l, \rho_l u_l, E_l), & x < x_0, \\ (\rho_r, \rho_r u_r, E_r), & x > x_0, \end{cases} \tag{2.11}$$

Considering *quasi-linear form* of the PDEs

$$\frac{\partial U}{\partial t} + J \frac{\partial U}{\partial x} = 0 \tag{2.12}$$

with the Jacobian matrix

$$J = \frac{\partial F}{\partial U} = \begin{pmatrix} 0 & 1 & 0 \\ \frac{(\gamma-3)u^2}{2} & (3-\gamma)u & \gamma - 1 \\ \frac{(\gamma-1)u^3 - uH}{2} & H - (\gamma-1)u^2 & \gamma u \end{pmatrix} \tag{2.13}$$

Eigenvalues corresponding to the Jacobian matrix are given by

$$\lambda^0 = u, \quad \lambda^+ = u + a, \quad \lambda^- = u - a \tag{2.14}$$

with the corresponding eigen vectors

$$\nu^0 = \begin{pmatrix} 1 \\ u \\ \frac{u^2}{2} \end{pmatrix}, \quad \nu^+ = \begin{pmatrix} 1 \\ u + a \\ H + au \end{pmatrix}, \quad \nu^- = \begin{pmatrix} 1 \\ u - a \\ H - au \end{pmatrix} \tag{2.15}$$

Hence we can conclude that the Jacobian matrix J can be diagonalized as there exist real eigenvalues. This also means that the system of equations is *hyperbolic* in nature. For numerical point of view, hyperbolic equation suggests upwinding is the simple way to calculate the solution of such equation. For any point $P(x,t)$ the solution can be obtained by gathering all the information transported through the characteristics starting from poiny P and going back to regions where the solution is already known [21].

2.6 Analytic solution for Euler equation

Shock tube being characterized by presence of multiple flow discontinuities (shocks), exhibits variant flow physics in its different regions (Regions (L),(E),(2),(1) & (R) Figure 2.1). Each section is governed by a separate physics, hence different empirical relations exist for each section separately. The analytical solution of Euler equations involves solving these empirical relations within their respective regions and combing the piece-wise results in order to give total results of the shock tube. Analytical solutions are at utmost importance in ISTP, since the results of these solutions will serve as input to the ISTP instead of experimental results. Hence the data from the analytical solution is being used as pseudo-experimental data. For this purpose a in house analytical Euler solver (EULER_EAFZ) is developed. If the reader wishes to understand the mathematics involved in development of the solver (EULER_EAFZ) appendix C.1 can be referred.

2.7 Numerical Modeling of Euler equation (CFD)

During the past decade numerous numerical method schemes have been proposed to solve the non linear hyperbolic equations i.e. the Euler equations 2.6. In the previous section it was proven that Euler equation possess non linearity due to presence of discontinuities in solution (due to Shocks) and the system of equations are hyperbollic in nature. Due to this fact it is difficult to solve the system of equations numerically. Inherited in any numerical simulation scheme is assumption of continuity of the solution, typically near to shock these schemes produce unwanted oscillations. Also there exist problems due to diffusion of truncation error across the discontinuities and with each time step this error is getting worse [22].

There have been many FDM schemes that were investigated in the Sod's classic paper on solving non linear hyperbolic PDEs [22], in which the results indicate that none of the FDM methods replicate the exact solution to high level of precision. Hence a FVM solver (EULER_AFZ) to solve the shock tube problem has been developed, that uses n^{th} order Runge–Kutta scheme for temporal discretization and Van Leers scheme for flux discretization. The solver uses concept of artificial viscosity via Venkatakrishnan [23] slop limiters to solve the problem oscillations associated with numerical solution flow characterized by shocks. If the reader wishes for more detailed explanation for development of this solver (EULER_AFZ) appendix C.2 can be referred.

In the next chapter

With physics and mathematical background of shock tube being detailed in this chapter, further to formulate the inverse problem solution in the shock tube, detailed study of inverse problems in relevance to the shock tube problem is presented in the following chapter. Mathematics and strategy of inverse computations in the shock tube is also presented.

Chapter 3

Inverse problems

Outline

This chapter is presented in order to have in-depth understanding of how inverse problems work. General introduction of inverse problems and a real world application of the field is presented in section 3.1. It was introduced in the first chapter that inverse problem is always an ill posed problem, theory in support of this is presented in 3.2. To elaborate on classification of inverse problems section 3.3 is presented. Finally section 3.4 details the mathematics involved and the methodology followed to solve the ISTP.

3.1 Inverse problems an introduction

Theoretical physics allows us to make predictions i.e., it gives description of the physical system under consideration and it can predict the outcome (observations or measurements). Class of problems which predict measurement are called the *forward problem*, the *simulation problem* or the *modelization problem*. The *inverse problem* in contrary consists using measurements or observations to infer values of parameters that characterize the physical phenomenon (system). Inverse problems is encountered in almost all the branches of engineering; astrophysicists, statisticians and mathematicians are among those who are interested in subject of inverse problems.

To cite and example for use of inverse problems, due to aerodynamic heating surface temperatures for a atmospheric reentry vehicle is so high that temperature sensors are impossible to be planted on it, instead sensors are placed beneath the surface where they can withstand the temperature, and from measurements at this location, surface temperature of the vehicle is calculated (inverse problem). For such problem inverse problems can also be used to estimate the thermophysical properties at such high temperatures.

3.2 Inverse problems- well posed or ill posed

In order to recognize the difficulties associated with inverse problems, mathematically any inverse problem belongs to a class of *ill posed problem* [24], whereas any forward model problem is *well posed problem*. A well posed problem described by Hadamard [25] should satisfy the following:

1. The problem should have a unique solution;
2. The solution of the problem must exist;
3. For small perturbations in input data solution must be stable.

Analyzing the three conditions, 1) in case of a inverse problems unique solution only exists for some special cases [26]. 2) Existence of solution for inverse problem may be assured by physical reasoning. 3) Any inverse problem is very sensitive to errors in input data and mostly requires special techniques in order to satisfy condition 3. Hence from this discussion it may be proved that inverse problem is always ill posed in nature.

For decades it was reasoned that violating any of the conditions of well-posedness would mean that problem is unsolvable and results obtained are meaningless. This belief caused decrease in popularity of inverse problems among mathematicians, engineers and physicists. It was *Beck's function estimation approach* [26], *Alifanov's iterative regularization techniques* [27] and *Tikhonov's regularization procedure* [24] that revitalized this interest. Hence for successful solution of any inverse problem it needs to be reformulated from a ill-posed problem to an approximated well-posed problem.

3.3 Inverse problems- types

In general inverse problems can be classified into two types *parameter estimation inverse problem* and *function estimation inverse problem*. Parameter estimation problems as the name suggests is associated with inverse calculations of limited number of parameters, that can be as small as one, might be as large as half a dozen or occasionally could go higher. Whilst for a function estimation problem, the number of parameters to be described as generally large, maybe hundreds or thousands. Parameter estimation problems may or may not be ill posed but function estimation problems are always ill posed in nature.

3.4 Inverse problem- solving inverse shock tube problem

Significance of inverse problems can be visualized in a better way by referring to the problem of flow inside a shock tube. Physical, qualitative and mathematical behavior of this problem were elaborated in sections 2.2, 2.4 and 2.5. Mathematically the 1D flow phenomena is formulated by equations the continuity equation, the momentum equation and the energy equation given in

section 2.5 and the initial conditions:

$$\text{for} \quad t = 0 \begin{cases} (\rho_l, p_l, u_l), & x < x_0, \\ \rho_r, p_r, u_r), & x > x_0, \end{cases} \tag{3.1}$$

Numerical modeling (CFD) can be used in order to solve equations of mass, momentum and energy simultaneously, for any parameter (ρ, p, u) at a given time $t > 0$ by using initial conditions (3.1). Solution of this problem will be called as *direct problem* solution. Now consider solving

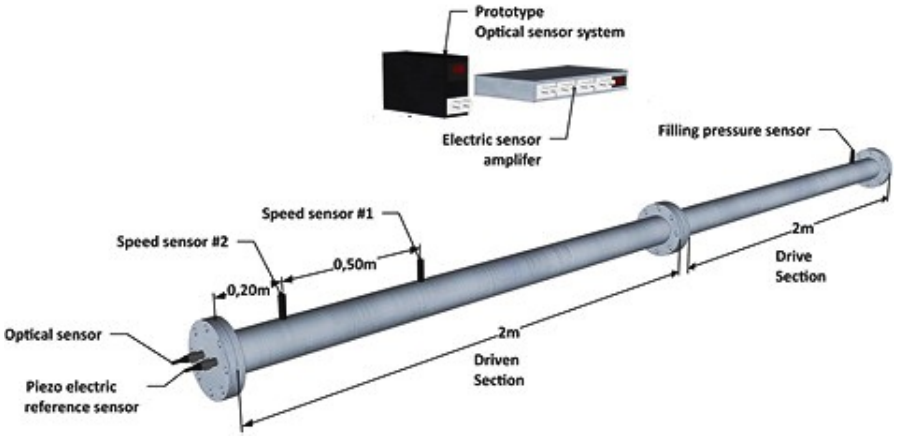


FIGURE 3.1: Experimental sensing in shock tubes

problem that is similar to above but initial conditions (ρ_r, p_r, u_r) at $t = 0$ are missing or not known while other initial conditions (ρ_l, p_l, u_l) are known. This can be visualized by imagining a gas in the working section of tube (refer to Figure 2.1) whose properties are unknown while it went under pressurization. The problem in hand now requires to determine the unknowns (ρ_r, p_r, u_r), in order to compensate the lack of information due to missing initial conditions experimental measurements of $p(x_{data}, t)$ or $u(x_{data}, t) \equiv Y$ at a given points (*data point*) inside the tube x_{data} and at time t are provided as inputs to the inverse problem. Figure 3.1 shows some of the common techniques of extracting velocity or pressure in a shock tube. This now becomes an *inverse problem* as it concerns solving the flow problem in reverse time to *estimate* the initial conditions. The basic idea of this is explained in Figure 3.2. The reason why *estimate* was in place of determination, is that experimental data contains errors as a result inverse analysis will always recover estimated outputs not exact. The *inverse problem* can be mathematically formulated by using the three equations, continuity, momentum and energy and

$$\text{for} \quad t = 0 \Big\{ (\rho_l, p_l, u_l), \quad x \leq x_0 \quad \text{(Initial} \quad \text{condition)}, \tag{3.2}$$

$$p(x_{data}, t) = \text{(Experimental} \quad \text{data)} \tag{3.3}$$

Inverse problem given above has missing initial conditions, hence is referred to as *boundary value inverse time problem*. For the given set of *missing values* inverse problem consists in minimizing the distance between the *predictions* and the related *measurements*. Also there may exists inverse problems concerning other unknowns energy generation, thermophysical properties etc. ISTP falls under *parameter identification* problem, p, ρ, u at initial time are treated as parameters. In order to obtain solution for ISTP, iterative method as described by knight et al [9] is used:

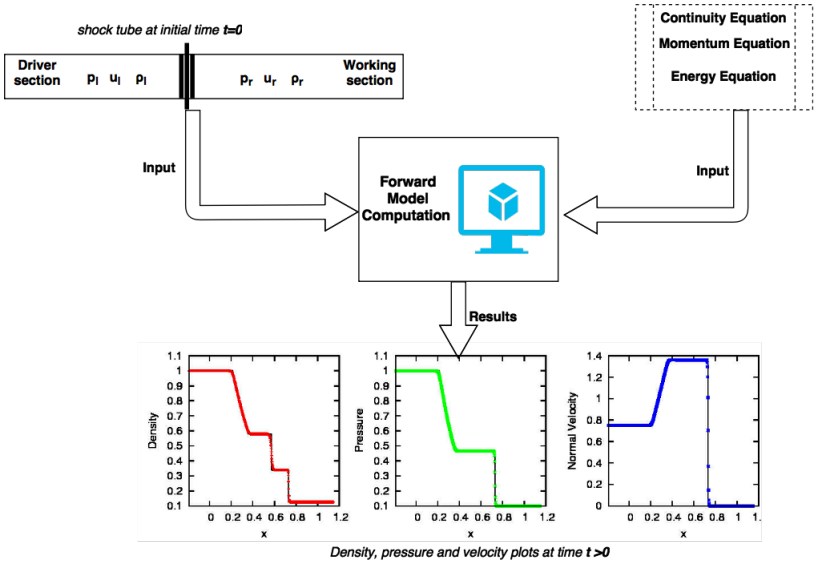


FIGURE 3.2: Forward and inverse shock tube problem

it requires iterating the forward model until *cost function* is minimized. *Cost function* may be defined as a function of model data (Y_{mo}) and experimental data (Y) which when minimized gives the solution to the inverse problem .

3.4.1 Cost function

The cost function also known as *objective function* is often expressed as norm of difference between selected data from the forward model Y_{mo} and experimental data at the same point Y. Mathematically cost function is given as

$$\mathcal{J}(Y_{mo}) = \| Y - Y_{mo} \|_{\mathcal{X}}^2 \tag{3.4}$$

Here $\mathcal{X}$ specifies the choice of norm (L_2, euclidean, etc.) and the function is squared in order to remove any discontinuities present, typically one uses $\|u\|_x^2 = \sum_i u_i^2$. Although cost function can be explicitly given in terms of model data Y_{mo}, but the minimization cost function used in inverse problems is also a function of parameter ψ (what we search for) since $Y_{mo} = f(\psi)$, hence cost function to be minimized (AKA reduced cost) is given as

$$j(\psi) = \mathcal{J}(Y_{mo}) \tag{3.5}$$

In the case of ISTP, experimental data (Y) is a pressure p or velocity u reading at the data point. Exact Euler solver developed on analytical solution theory mentioned in section 2.6 is used to obtain the pseudo-experimental data Y. It needs to be mentioned that the input to inverse problem, the experimental data, always contains errors, as the exact solver is analytical solution it is error free, but we need to keep in mind the numerical error that exists due to computations in forward modeling these errors are assumed to simulate the experimental errors. The model data Y_{mo} is obtained by running the FVM forward model code (Euler solver EULER_AFZ). The fact to be mentioned is as CFD solution via the Euler solver is always approximate this adds to the noise or error while solving the ISTP.

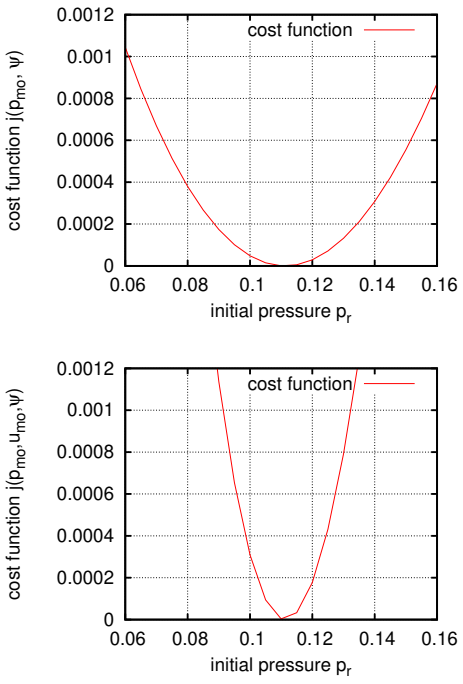


FIGURE 3.3: One parameter estimation ($\psi = p_r$) cost function Left- pressure based cost function $j(\psi) = \| p - p_{mo} \|^2$ Right- pressure and velocity based cost function $j(\psi) = \| u - u_{mo} \|^2 + \| p - p_{mo} \|^2$

For this study using equation 3.4 following cost functions are used

$$j(\psi) = \frac{1}{2} \| p - p_{mo} \|^2 \tag{3.6}$$

$$j(\psi) = \frac{1}{2} \| u - u_{mo} \|^2 \tag{3.7}$$

$$j(\psi) = \frac{1}{2} \| u - u_{mo} \|^2 + \| p - p_{mo} \|^2 \tag{3.8}$$

As parameter estimation inverse time problem of recovering initial conditions p_r, ρ_r, u_r is the topic of this study, It can be sub divided into two or one parameter identification problem of p_r, ρ_r. Note the choice to exclude u_r from the identification parameter comes from the fact that at $t = 0$, u is always zero. hence we can have three cases of parameter identification

- parameter identification of p_r here $\psi = \left(p_r\right)$
- parameter identification of ρ_r here $\psi = \left(\rho_r\right)$
- parameter identification of p_r and ρ_r here $\psi = (p_r \quad \rho_r)^T$

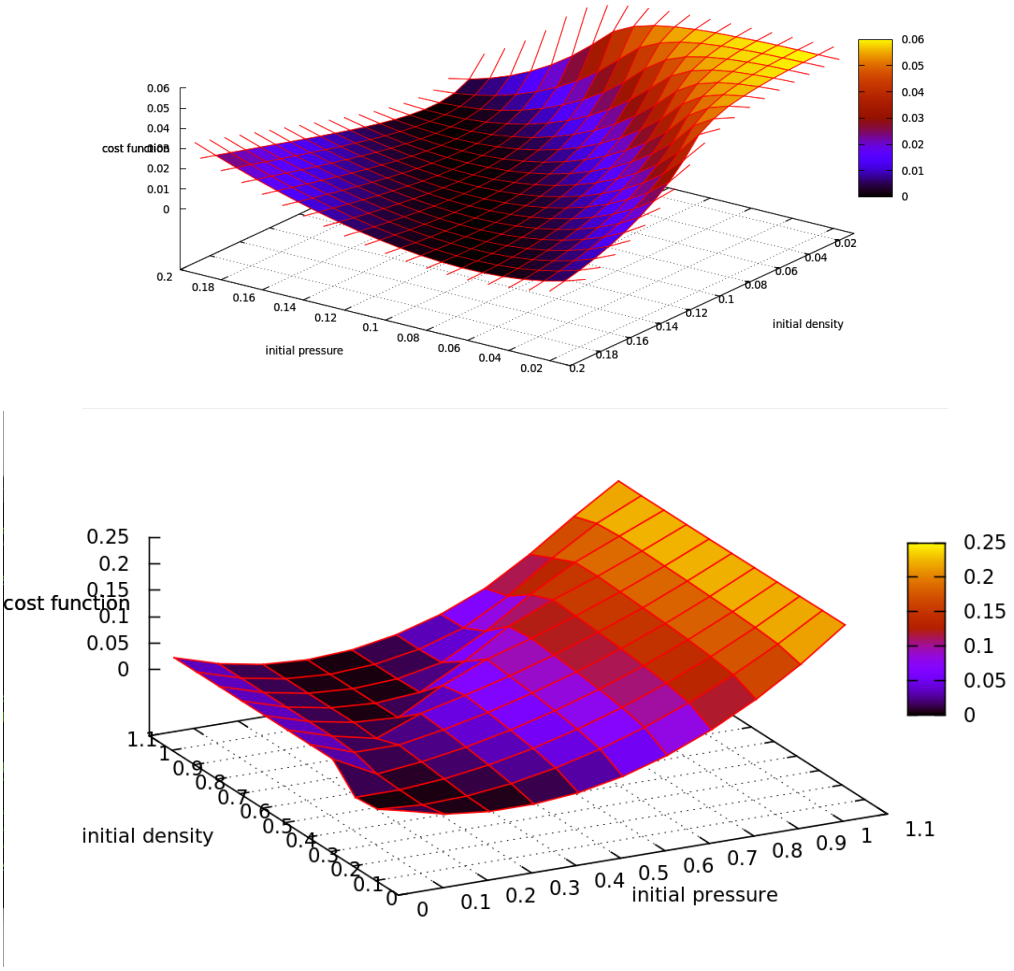


FIGURE 3.4: Two parameter estimation ($\psi = (p_r)$) cost function Top- pressure based cost function $j(\psi) = \| p - p_{mo} \|^2$ Bottom- pressure and velocity based cost function $j(\psi) = \| u - u_{mo} \|^2 + \| p - p_{mo} \|^2$

In order to check the convexity of cost functions, some plots of cost functions have been presented in Figures 3.3 and 3.4, these depict cost function for one parameter identification and two parameter identification respectively. Figure 3.4 shows approximately convex cost function and a non linear cost function being developed using different cost function formula.

3.4.2 Method to solve the inverse shock tube problem

Iterative algorithm as used by King et al [9] has been used to solve the parameter identification inverse problem. Steps involved to solve a two parameter problem (p_r, ρ_r) have been explained below and the algorithm of which is given in appendix A.8

- Start by guessing the values of parameters to be estimated $\psi = \psi_i^0$ here $i = 1, 2, ..., m$, m is total number of parameters in case of ISTP $i = 2$.

- Use these guess parameters as inputs to forward model solution via Euler solver. In case of ISTP initialize the forward model with guess values of (p_r, ρ_r).
- Obtain the cost function j by using the forward model result (Y_{mo}) at experimental measurements at the data point (Y). In the case of ISTP Y_{mo} is given by running the Euler solver (EULER_AFZ) for guessed parameters (p_r, ρ_r) and experimental results (Y) are simulated by analytical solver (EULER_EAFZ).
- Check if the cost function has reached the desired minimum value i.e. is the *stopping criteria* is met, based on this either stop or continue the iteration. In order to continue the iterations *optimization technique* is used to improve upon the next guess parameter ψ values.

In general the iterative algorithm is computationally very expensive so there exists a need to apply *optimization* so that the minima of the cost function can be obtained in less number of iterations.

3.4.3 Stopping criteria

Generally convergence of iterative algorithms is non finite, hence the need of stopping criteria. Different iterative schemes demand different stopping criteria for optimal performance of the algorithm. Following are some of the stopping criteria used for cost function minimization

$$\| \bigtriangledown j(\psi^k)\|_2 \leq \varepsilon; \tag{3.9}$$

$$\| \bigtriangledown j(\psi^k)\|_\infty \leq \varepsilon; \tag{3.10}$$

$$|j(\psi^k) - j(\psi^{k-1})| \leq \varepsilon; \tag{3.11}$$

$$j(\psi^k) - j(\psi^{k-1}) \leq \varepsilon; \tag{3.12}$$

$$j(\psi^k) \leq \varepsilon; \tag{3.13}$$

Here k is the iteration number and ψ^k is parameter at iteration k. The last choice 3.13 is most commonly used in inverse problem i.e. when cost function reaches a desired minimum iterations should stop [27].

In the next chapter In section 3.4.2 of this chapter need of optimization was emphasized in order to reduce the computational expense of solving the inverse problem. Next chapter highlights, formulates the use of optimization in inverse problems. Theory and mathematics of some optimization techniques that are relevant to ISTP has been developed in the chapter to follow.

Chapter 4

Optimization

Outline

The fact that optimization is one of the important ingredient for a well solved inverse problem, this chapter introduces some of the optimization algorithms that have been used with ISTP. Section 4.1 introduces optimization in general, to follow sections 4.2 and 4.3 present typical classification of optimization methods. To detail on the algorithms compatible with ISTP a few applicable optimization methods have been reviewed in sections 4.4, 4.5, 4.6, 4.7, and 4.8.

4.1 Optimization- Introduction

Optimization or *mathematical programming* is a widely used mathematical tool that assists in selection of best element, for some given criteria, among a set of elements. Optimization is popular in mathematics, statistics, operational research, computer science and other engineering fields. An elementary case optimization problem may consist of minimizing of maximizing a function $f(x)$ by systemic choice of inputs from within the allowed set of inputs or parameters. Optimization in inverse problem includes search of *best available* value of the cost function in a constrained domain. Mathematically function optimization problem can be given as

$$f(x) \rightarrow \mathbb{R} \quad \forall \quad x = [x_1, x_2, x_3, ...]^T, \quad \exists \quad \mathbb{R}$$

From the vector $x = [x_1, x_2, x_3, ...]^T$ it is required to find element $\overline{x}$ such that

$$\begin{aligned} \text{minimization problem:} \quad & f(\overline{x}) \leq f(x) \quad \forall \quad x_1, x_2, x_3, ... \\ \text{maximization problem:} \quad & f(\overline{x}) \geq f(x) \quad \forall \quad x_1, x_2, x_3, ... \end{aligned}$$

In ISTP, as the goal demands to find parameters $(\overline{\psi})$ that amount to minimum cost function $(j(\overline{\psi}))$, this problem can be categorized under *function minimization optimization* problem. For a cost function to be minimum, *optimality conditions* needs to be satisfied. In search for the

desired parameter $\overline{\psi}$ that gives the minimum of cost function $j(\psi)$ optimality condition demands the following [28]

1. In the function plane $(f(\psi), \psi)$, $\overline{\psi}$ is a stationary point, i.e. $\bigtriangledown j(\overline{\psi}) = 0$
2. The second derivative of cost function (AKA Hessian) $\bigtriangledown^2(\overline{\psi}) = (\partial^2(\overline{\psi})/\partial\psi_i\partial\psi_j)$ is positive definite i.e. $\forall y \in \mathbb{R}, \quad y \neq 0, \quad (\bigtriangledown^2 j(Y_{mo})y, y) > 0$

These two conditions need to be satisfied for a function to be minimum globally or locally. For more in depth knowledge on optimization some well known books and articles [29–32] can be referred.

In general *n-dimensional function optimization* can be classified into two types

- *Zero order n-dimensional optimization*
- *Gradient type optimization*

Note that there are vast number of algorithms that are based on either of the two optimization techniques but among these explanation of only few will be made (ones used with ISTP).

4.2 Zero order n-dimensional optimization

Zero order n-dimensional optimization AKA *Derivative free optimization* (DFO), computes the minimum of a cost function $j(\psi)$ without looking for its gradient $\bigtriangledown j(\psi)$. Such methods are most commonly used when there is presence of local minima withing the function. The method can also be used at times when the gradient $\bigtriangledown j(\psi)$ is not available or very difficult to calculate. In the recent past there has been an increase in development of computational tools solving optimization problems based of DFO methods [33]. Among the many available DFO methods, *Simplex method* and *Particle Swarm Optimization* (PSO) are used in solving the ISTP.

4.3 Gradient type n-dimensional optimization methods

These are the most widely used methods to find minimum in case of a convex cost function. This method utilizes the gradient wise descent approach to find $(\overline{\psi})$. In order to search for stationary point $\overline{\psi}$, following iterative procedure is adapted

- start by guessing a random value ψ^0.
- At each iteration k, the next better guess value ψ^{k+1} is formulated as $\psi^{k+1} = \psi^k + \alpha^k d^k$ were d^k is displacement direction and α^k step size. This displacement direction d^k depends on the choice of descend direction, but always it remains a function of gradient $\bigtriangledown j(\psi^k)n$.

Many algorithms *BFGS, Levenberg Marquardt, DFP, steepest descent, conjugate gradient,* etc. use the Gradient type n-dimensional optimization principle to solve the cost function minimization problem. Among these *steepest descent* and *conjugate gradient* method are used in ISTP.

4.4 Gauss-Newton optimization

This matrix based algorithm is classified under Gradient type n-dimensional optimization. The method is widely used to solve the least square problems, in case of ISTP the cost function $j = \frac{1}{2}(Y - Y_{mo})^2$ makes it an interesting case to solve via Gauss-Newton (GN) Algorithm. This method possess advantage that the hessian is not to be calculated. The algorithm is efficient for cases involving less parameters, in terms of a ISTP as two parameters (ψ_1, ψ_2) are to be searched for GN forms a reasonable choice for the choice of optimizer.

Gauss-Newton approach is an iterative approach, were it improves the initial guess parameters with each iteration k in order to find the optimal parameter vector $\overline{\psi}$ that minimizes the cost $j(\psi)$. The algorithm of this method is given in appendix B.1. The improved guess parameters after each iteration are given as a function of sensitivity matrix S and error vector E. Mathematically

$$\psi^{k+1} = \psi^k + d\psi \tag{4.1}$$

$$d\psi = f(S, E) \tag{4.2}$$

$$d\psi = -[S^T S]^{-1} S^T E \tag{4.3}$$

As it can be inferred from equations 4.3 that working of algorithm depends on inversion $[S^T S]$ matrix, hence forth dependent on sensitivity matrix S. To get more insight how the sensitivity matrix and error vector are formulated appendix B.1 can be referred to.

4.5 Simplex or Nedler-Mead optimization

Listed in the top 10 algorithm of 20^{th} century by journal or Computing in Science and Engineering, Simplex method AKA Nedler-Mead (NM) method, is one of the most popularly used gradient free (Zero order n-dimensional) optimization method for function based optimization. The method is simple to understand and code. The method can be well understood by the classic paper of Nedler and Mead [34] who devised this method.

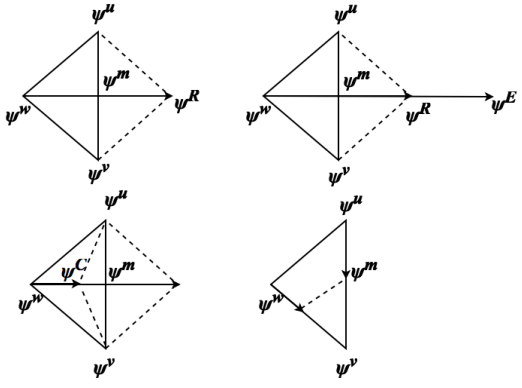


FIGURE 4.1: clockwise from top left ; Reflection, extension, contraction and inside contraction

A simplex (χ_0) is defined as structure formed by $n+1$ points, with n being total number of parameters $(n = \dim(\psi))$. A simplex (χ_0) is defined in the parameter plane with each point represented by a guessed parameter vector ψ_i^0 and having a costs $(j(\psi_i^0))$ with $i = 1, 2, 3, .., n+1$. NM method being an iterative procedure, generates new simplex at each iteration that is closer to the minimum eliminating the points (ψ_i) with worse costs $(j(\psi_i))$.

The in hand ISTP concerns finding out two parameters hence dim $\psi = 2$ this means the simplex generated contains $(2+1 = 3)$ points, or in other words simplex here is triangle. Hence a triangle is generated at each iteration which gets closer and closer to minimum iteration after iteration. The strategy for forming better simplex in case of ISTP (dim $\psi = 2$), starts by sorting out the vertices of triangles according to one with best cost to the worst, hence

$$j(\psi^u) < j(\psi^v) < j(\psi^w) \tag{4.4}$$

with ψ^u being the best point and ψ^w the worst one.

This is followed by improvement in simplex via *reflection, extension* or *contraction*, the methodology of which is explained below

1. *Reflection* consists of reflecting the worst point ψ^w to ψ^R perpendicular to the segment $\psi^u\psi^v$. Depending on the cost $j(\psi^R)$ in comparison to ψ^m, (ψ^m is mid point of $\psi^u\psi^v$) the parametric space may either be extended or contracted.
2. *Extension* for condition $j(\psi^R) < j(\psi^m)$ being satisfied the parametric space is extended to ψ^E. After establishing the point ψ^E, if $j(\psi^E) < j(\psi^R)$ new point ψ^E is accepted and ψ^R is rejected and the new point replaces ψ^w.
3. *Contraction* if extension is not an option i.e. $j(\psi^R) > j(\psi^m)$, parametric domain is contracted to point ψ^C and following the condition if $j(\psi^C) < j(\psi^R)$, ψ^w is replaced by ψ^C else the whole simplex is contracted.

Above mentioned strategy can be well understood by going trough the Figure 4.1. The Algorithm used for ISTP is mentioned in details in appendix B.2.

4.6 Particle swarm optimization

Particle swarm optimization, like simplex also belongs to gradient free optimization. It is a genetic algorithm that was devised by Kennedy and Eberhart in 1955 [35]. It involves particles being initiated in the parameter space (ψ), and these particles move in search of minimum cost function $j(\overline{\psi})$, the particles here mimic the movement of group of birds (swarm) in search of food (minimum cost).

Initially particles (value of ψ in parameter space) are randomly placed in the parameter space. Each particle is mobile and can move with a certain velocity V_k^p. At each successive iterations particles change its position ($x_k^p = \psi^p$), moving with velocity V_k^p. The particle velocity V_k^p is function of best location of any particle in swarm $G.best_k$ (global-best) and its own best location in the past iteration $p.best$ (particle-best). Mathematically velocity at an iteration k is given by

$$V_k^p = \omega V_{k-1}^p + C1R1(p.best_k - x_k^p) + C2R2(G.best_k - x_k^p) \tag{4.5}$$

The values of $\omega = .7$ and $C1 = C2 = 1.46$ [36] and $R1$ & $R2$ are random variables that follow uniform distribution in [0,1]. Updating the velocity of the particle by equation 4.5, particle

position is updated as

$$x_k^p = x_{k-1}^p + V_k^p \tag{4.6}$$

Qualitatively the above two equations influence the particle movement by

- how much particle trusts its experience ($p.best$) now and in the past,
- how much particle trusts its neighborhoods experience ($G.best$).

This phenomena is well detailed in Figure 4.2. Usually the size of swarm is 20 to 30 particles [36], but for some problems it may go upto 100. The stopping criterion for this method is generally the maximum number of iterations or a critical cost function ($j(\overline{\psi})$) value [28]. The algorithm works quite well with non linear functions, and to identify local minma in the function, but it comes with a disadvantage of being computationally expensive as at each iteration the forward model is run several number of times (depending on swarm size).

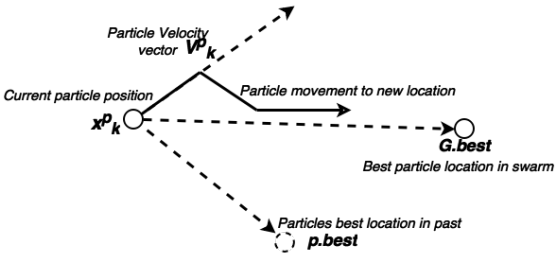


FIGURE 4.2: PSO : Particle movement to new location

The algorithm developed for ISTP case is given in appendix B.3. The algorithm is well suited for ISTP as the cost function contains presence of local minima this can be observed in Figure 3.4.

4.7 Steepest gradient optimization

Steepest gradient (SG) method is simplest of all methods classified under Gradient type optimization method. This algorithm makes use of gradient ($\nabla j(\psi)$) at each iteration in order to descent to the minima ($\overline{\psi}$). At every iteration k, gradient $\nabla j(\psi)$ gives largest descent in increase of cost $j(\psi)$. The new point according to the algorithm depends on step size (α^k) and is mathematically given as

$$\psi^{k+1} = \psi^k - \alpha \frac{\nabla j(\psi^k)}{\|\nabla j(\psi^k)\|} \tag{4.7}$$

As the name suggests here the step size α is predefined and generally chosen as function of iteration $\alpha = \alpha(k)$. The iterations converge to minima $\overline{\psi}$ provided

$$\alpha \to 0 \quad as \quad p \to \infty$$

For the ISTP the value of α chosen to satisfy the above criteria is given by $\alpha = 1/k$. The algorithm of the method used with ISTP is given in appendix B.4. The method is simple to understand and code but has its main drawback of slow convergence rate compared to other gradient based optimizers.

4.8 Conjugate gradient optimization

Developed by Magnus Hestenes and Eduard Stiefel in 1952 [37], this method is one of the most popular and efficient methods among gradient based optimization methods. The method has two variants, one applicable to *quadratic functions minimization* and the other for any *arbitrary function minimization*. In ISTP the latter one is used, the algorithm of which is given in appendix B.5. This method is most popularly used to minimize linear set of equations (matrix systems) but it should be noted that this method is equally efficient for the case of function minimization also.

The algorithm starts by guessing parameters ψ^0 and iteratively converging this guess into minimum $\overline{\psi}$. At the beginning of iterations the direction d^0 is given by the negative gradient $-\nabla j(\psi^0)$. In CG the step size is given as a function of forward model data ($\alpha = f(Y_{mo}, Y'_{mo})$), thus a optimal choice of step size unlike SG algorithm. In CG the direction of descent depends on gradient as well as on previous descent direction, again an optimal choice for the descent direction as well. Mathematically step size α and direction of descent are given as

$$\alpha = -\frac{\langle Y'_{mo}, Y^-_{mo} - Y_{mo}\rangle}{\langle Y'_{mo}, Y'_{mo}\rangle} \tag{4.8}$$

$$d_k = -\nabla j(\psi^{k-1}) + \beta d_{k-1} \tag{4.9}$$

with k being the iteration number and β is given by

$$\beta = \frac{\langle \nabla j(\psi^k), \nabla j(\psi^k)\rangle}{\langle \nabla j(\psi^{k-1}), \nabla j(\psi^{k-1})\rangle}$$

here $\langle a, b\rangle = \sum_i a_i b_i$ From the above equations the next better guess is given by

$$\psi^k = \psi^{k-1} + \alpha d_k \tag{4.10}$$

Mathematics involved behind the equations can be well understood by referring [28]. The method proves advantageous as it converges very fast (for a quadratic function converges in two iterations) saving computations expenses but the method does not work well for a function that contains local minma.

In the next chapter: In previous chapters mathematics, theory and methodology of inverse problems has been discussed in details. Shock tube problem mathematics, theory and forward modeling has also been developed in the previous chapters. In order to present its application and performance, results and discussions on use of inverse problems with the shock tube problem (ISTP) is presented in the next chapter.

Chapter 5

Results, Discussions and Conclusions

Outline

Application results of inverse problems with shock tube has been presented in this chapter. To quantify the problem section 5.1 has been presented, data used in ISTP is mentioned in this section. Results concerning some pre-preparations before solving the actual ISTP have been presented in section 5.2. In order to begin with inverse problems first a simpler ISTP concerning one parameter identification was solved results of which are presented in section 5.3. Results and analysis concerning the main ISTP have been detailed in section 5.4, the section also details how different optimization algorithms work with ISTP. Two new hybrid optimization algorithms that are tailored for ISTP these are presented in section 5.6.

5.1 Problem Data

In this project a shock tube problem of *non sonic refraction* (normal shock does not reach the ends of the tube) is considered, this is the classic problem of Sod [22]. For the *forward problem* the following initial conditions exist

$$\begin{cases} \rho_l & = 1.0 \quad p_l = 1.0, \quad u_l = 0.0, \quad \forall \quad x < x_0, \\ \rho_r & = 0.125, \quad p_r = 0.1, \quad u_r = 0.0, \quad \forall \quad x > x_0, \end{cases} \tag{5.1}$$

Note that the primitive variables (p, ρ, u) have non dimensional values [0,1]. The length of the shock tube runs from $x = -5.0$ to the left to $x = +5.0$ to right, with a total length of $l = 10.0$. The diaphragm stands at $x = x_0 = 0.0$.

The *pseudo-experimental calculations* (The exact solution in this case) have been taken at time $t = 1.7$, before the shock reaches the ends. The diaphragm is assumed to rupture at time

$t = 0.0$. The data point at which the sensors are located, to derive the pressure and velocity, is at $x = 0.5$, the reason to chose this point is given in section 5.2 (sensitivity analysis). At this point the pseudo-experimental values at $t = 1.7$ are

$$p_{x=.5} = Y_1 = 0.3129 \quad u_{x=.5} = Y_2 = 0.9048 \tag{5.2}$$

For our case of ISTP, parameter estimation problem is considered i.e., the goal is to recover the initial conditions (p_r, ρ_r at $t = 0$) for the shock tube under consideration. As the initial conditions on the right side of the tube are already given in equation 5.1, this data will serve as validation for inverse modeling.

Note that all the results to follow have been developed using code INVERSE_ISTP, flow chart explaining the code is mentioned in appendix A.8.

5.2 Some prerequisite results

Concerning the forward model

Inverse problems are always ill posed as was emphasized in section 3.2. Due to such nature of inverse problems the errors in forward modeling should be minimal. In order to achieve so, the following has been adapted in the forward model code.

Grid convergence: A uniform grid with 1000 grid cells is chosen, the reason for such high number of cells lies in the fact that it allows to capture the discontinuities better. The choice is justified by referring to Figure 5.1. As it can be noticed that 4000 cells would have been a better option to capture the shock properly but the fact that it will computationally be very expensive, 1000 cells is preferred over it. ***Hence for ISTP 1000 celled grid is chosen.***

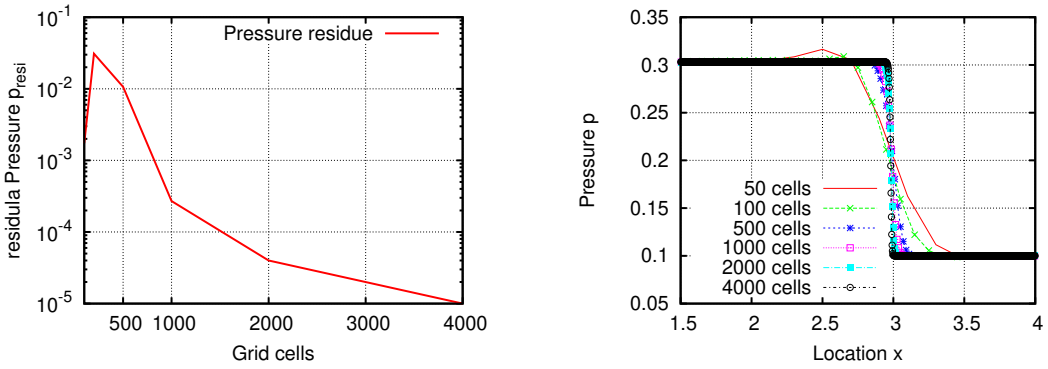


FIGURE 5.1: Grid convergence performed on forward model: Left) Residue pressure in comparison to experimental results at $x = 3$; Right) Different grid behavior at the normal shock at $t = 1.7$ sec

Courant number: The choice of courant number also effects the error in the forward model, optimal courant number between [0,1] should be chosen to give the minimal error. ***The courant number 0.95 is chosen for ISTP***, this is purely based on literature survey and the choice can be validated by referring [38].

Concerning the Inverse model

Sensitivity analysis: In order to establish the *data point*, location at which the sensor is to be placed, the sensitivity of the output solution was checked varying the inlet parameters (p_r, ρ_r). It was established that are area between the right end of expansion fan and the normal shock (*quasi-steady region*) i.e sections (2) and (1) (refer Figure 2.1) form the best choice for placing the sensor. The reason being that this area is very much reactive to any perturbation in inlet parameters (p_r, ρ_r) and approximately there exists a linear relation between the output solution and the inlet parameters. ***A through investigation reveled that point x=0.5 forms the best choice for sensor locations***. Figure 5.2 shows the linear plots that exist between the inlet $(p_r, \rho_r$ at $t = 0)$ and the output parameters $(p, u$ at $t > 0)$. The fact that also validates the location being optimal for the sensor is that cost function obtained at this point is nearly convex this can be noticed in Figure 3.4. It can be noticed from the Figure 5.2 that not all the plots show a linear trend, but that outputs have a monotonous trends with respect to input. Note that for the case of output pressure variation with initial pressure, output pressure varies linearly with initial pressure hence the cost function will be perfectly convex for this case.

Sensor choice: Out of the available pressure, velocity and temperature sensors for providing the experimental results, ***temperature sensors were excluded and only pressure and velocity sensors were used for ISTP***. The reason behind can be realized by referring to Figure C.2, it can be noticed that density possess a discontinuity at the contact discontinuity, this would in turn mean temperature being discontinuous around the region. Hence the sensitive region which was explained above would shrink so temperature sensing is avoided.

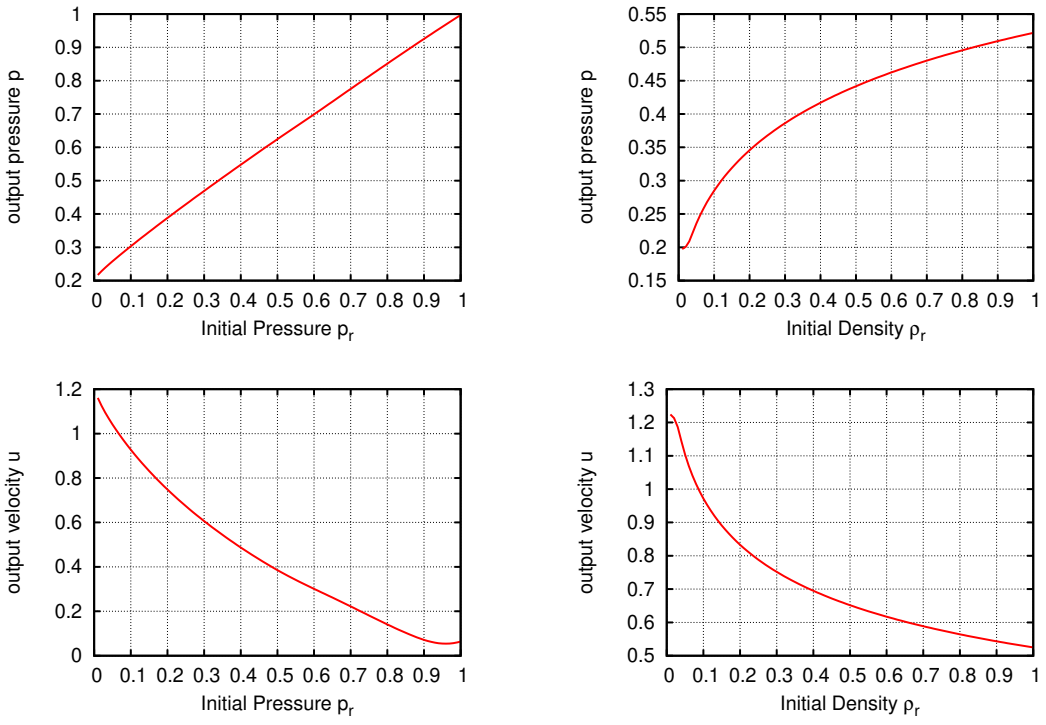


FIGURE 5.2: Sensitivity analysis between inlet $(p_r, \rho_r$ at $t = 0)$ and the output parameters $(p, u$ at $t > 0)$

5.3 One parameter estimation problem

For the sake of getting numbers on how inverse problems perform with one parameter identification, tests were run on, identifying initial pressure p_r assuming initial density ρ_r known and identifying initial density ρ_r assuming the initial pressure p_r to be known. Note, no optimization method was used as the problem is not computationally expensive. As the parameter to be retrieved is in non dimensional form hence ($\overline{\psi} < 1$). $\psi = 1.0$ forms the starting guess (ψ^0) for each case, with each iteration the value is decreased by a small perturbation ε. At each iteration the slope of cost function ($dj/d\psi$) is calculated, the value of ψ for which the slope changes its sign is the required $\overline{\psi}$. The two cases for the one parameter identification problem are,

- *case 1:* Identifying p_r here $\psi = p_r$ and $\rho_r = 0.125$ (given)
- *case2:* Identifying ρ_r here $\psi = \rho_r$ and $p_r = 0.1$ (given)

As it has already been discussed in section 5.1, for the forward problem the initial conditions are given by $\psi_1 = p_r = .1$ and $\psi_2 = \rho_r = .125$, based on this data the error in the parameters estimated from inverse problems is given in Tables 5.1 and 5.2.

TABLE 5.1: case 1: one parameter analysis: Pressure

Cost Function	Initial Pressure $\psi_1 = p_r$	Error %
$j(\psi) = \frac{1}{2}(p_{mo} - p_{x=0.5})^2$	0.11122	11.22
$j(\psi) = \frac{1}{2}(u_{mo} - U_{x=0.5})^2$	0.11128	11.28
$j(\psi) = \frac{1}{2}(p_{mo} - p_{x=0.5})^2 + \frac{1}{2}(u_{mo} - U_{x=0.5})^2$	0.11127	11.27

TABLE 5.2: case 2: one parameter analysis: Density

Cost Function	Initial Density $\psi_2 = \rho_r$	Error %
$j(\psi) = \frac{1}{2}(p_{mo} - p_{x=0.5})^2$	0.14011	12.087
$j(\psi) = \frac{1}{2}(u_{mo} - U_{x=0.5})^2$	0.14018	12.1439
$j(\psi) = \frac{1}{2}(p_{mo} - p_{x=0.5})^2 + \frac{1}{2}(u_{mo} - U_{x=0.5})^2$	0.14017	12.1359

It can be noticed from the tables that ***error in the output solution, for the three chosen cost functions remains approximately same i.e. 11 % for pressure calculation and 12 % for density***. However for the sake of judging pressure based cost functions perform slightly better, hence ***it can be a wise choice to install a pressure sensor in the working section for inverse problem measurements***.

5.4 Two parameter estimation problem

The problem of two parameter estimation involves having no a priori knowledge of gas in the working section at the initial time $t = 0$. Mathematically the problem concerns identification of two parameters

$$\begin{pmatrix} p_r \\ \rho_r \end{pmatrix} = \begin{pmatrix} \psi_1 \\ \psi_2 \end{pmatrix} \tag{5.3}$$

The solution of the problem was obtained using five different optimization algorithms, the details of which were mentioned in chapter 4:

1. *Gauss-Newton algorithm*
2. *Simplex algorithm*
3. *PSO algorithm*
4. *Steepest gradient algorithm*
5. *Conjugate gradient algorithm*

The reason for the choice and order of these methods is as follows

- *Gauss-Newtons (GN):* This is one of the basic optimization technique used with inverse methods. This matrix based method works very well for convex functions. The method is also the simplest of all the matrix based optimizers, hence a positive result from this method would implicate use of other matrix based optimizers while vice versa is true for negative result.
- *Simplex (NM) and PSO:* These are zero order methods, not requiring to find the cost derivative (gradient) making them easy to implement and code. Also the methods would converge in any case, with or without convex cost function.
- *Steepest Gradient (SG) and CG:* These methods are 1^{st} order gradient type methods. Not needing any matrix operations these methods are equally fast and reliable for convex cost functions.

It should be noted that for all the algorithms pressure based cost function was used, choice of such cost function was influenced by the fact that pressure based cost functions would perform well in comparison to other cost functions, this was noticed in section 5.2. Following are the results that were obtained for two parameter analysis

5.4.1 Gauss-Newtons algorithm

The matrix based ***algorithm failed*** to produce any results. As discussed in the section 4.4, the algorithm involves the inversion of matrix $[S^T S]$, it was established that the matrix was ill conditioned. Results for the 1^{st} iteration (the only iteration after which the algorithm crashes) shows

for iteration $k = 1$, with starting guess $\psi_1^0 = 0.5 \;\; \psi_2^0 = 0.5$

$$S = \begin{pmatrix} .88 & .672 \\ -2.04 & -1.552 \end{pmatrix} \quad [S^T S] = \begin{pmatrix} 4.9360 & 3.7574 \\ 3.7574 & 2.8603 \end{pmatrix}$$

The iteration ended up giving the following

$$\psi^{k+1} = \begin{pmatrix} -3.3562 \\ 4.3969 \end{pmatrix}$$

The result explains the reason why the algorithm crashed, as it can be noticed that obtained pressure ψ_1 is negative, which is not possible practically. This negative pressure when used with the forward model crashes the code. Mathematical analysis reveals that the condition number of $[S^T S]$ is 1522.7, which is too high. For this method to perform condition number should be close to 1.0. A conclusion can be made that due to correlation existing between ψ_1 and ψ_2, GN algorithm failed.

Although the result of the ***algorithm ended up in failure, but it eliminated the use of all other algorithms that involve matrix operation*** hence proving fruitful result.

5.4.2 Nedler-Mead algorithm (Simplex Method)

The gradient free algorithm performed well, acquiring convergence (notion of how fast algorithm outputs $\overline{\psi}$) in 10-20 iterations. As was discussed in section 4.5, the algorithm involves augmentation of simplex χ(triangle in this case) till stopping criteria is met. This phenomenon can been noticed in Figures 5.3 and 5.4. These Figures also explains how the algorithm reaches convergence by shrinking the huge simplex to a smaller ones.

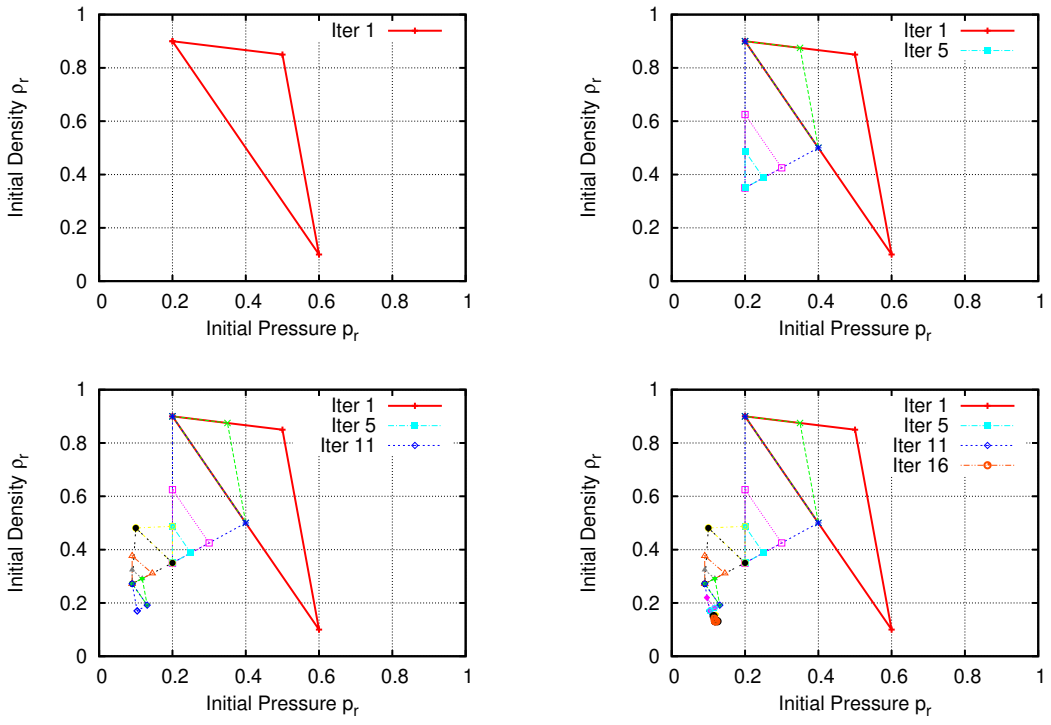


FIGURE 5.3: simplex Algorithm for change χ^0: From top left clockwise; at 1^{st} iteration , 5^{th} iterations, 11^{th} iterations and 16^{th} iteration.

In order to chose the initial simplex χ^0 (triangle), the three vertexes are chosen wide away from each other, so that maximal area in parameter space (ψ) can be covered. Concerning the stopping criteria, the iterations are stopped when the triangle (simplex χ) is minimized to almost a point, i.e all three vertices almost coincide. This is noticed in Figures 5.3 and 5.4 by observing the last iteration of the figures, notice that all three vertices almost overlap.

It should be noted that the results presented in this section are for a specified cases were three initial coordinates of the triangle (χ^0) i.e. the three guessed values in parameter space are

$$\text{case I (Figure 5.3):}\quad \psi_1^0 = \begin{pmatrix} .2 \\ .9 \end{pmatrix},\quad \psi_2^0 = \begin{pmatrix} .6 \\ .1 \end{pmatrix},\quad \psi_1^0 = \begin{pmatrix} .5 \\ .85 \end{pmatrix}$$

$$\text{case II (Figure 5.4):}\quad \psi_1^0 = \begin{pmatrix} .6 \\ .1 \end{pmatrix},\quad \psi_2^0 = \begin{pmatrix} .12 \\ .17 \end{pmatrix},\quad \psi_1^0 = \begin{pmatrix} .9 \\ .9 \end{pmatrix}$$

By running these different tests on this algorithm ***it is established that changing the initial guess values may lead to change in convergence of the algorithm, it may also result in some minor changes in the output parameters.*** To emphasize on this a test runs with different initial simple χ^0 are shown in Figure 5.4 and 5.3. However ***in general this algorithm produces stable results.***

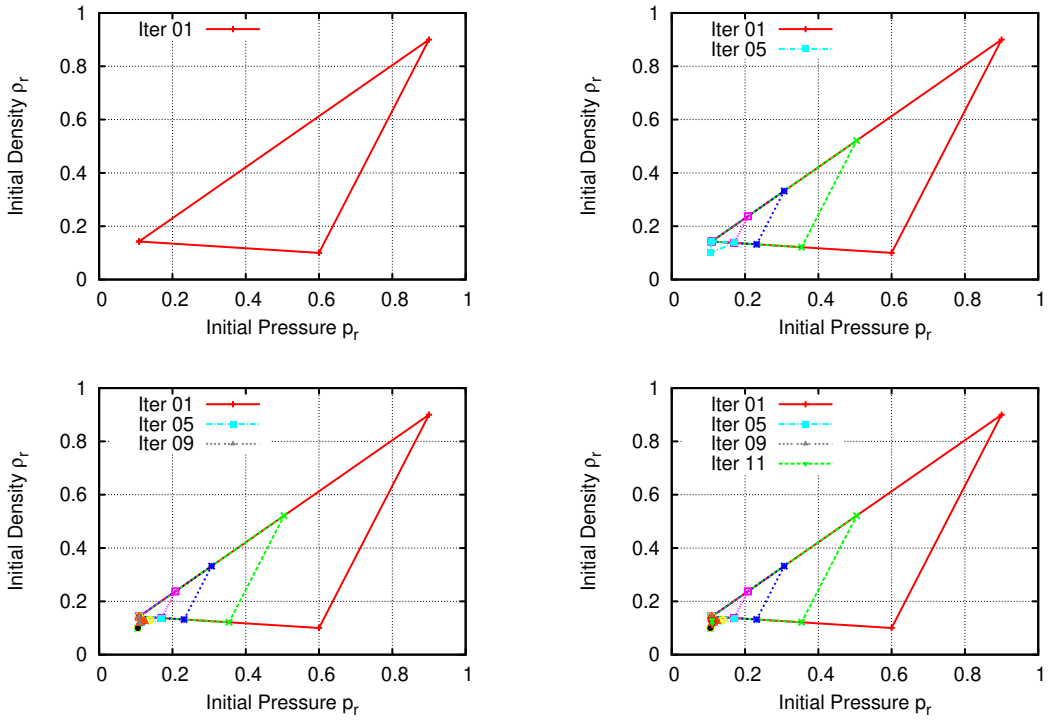


FIGURE 5.4: simplex Algorithm: From top left clockwise; at 1^{st} iteration , 5^{th} iterations, 7^{th} iterations and 11^{th} iteration.

To sum up the algorithm table 5.3 is presented

TABLE 5.3: Simplex algorithm results

	Iteration	Initial Pressure $\psi_1 = p_r$	Initial Density $\psi_2 = \rho_r$	Error in ψ_1	Error in ψ_2
Case I	16	0.118	0.131	18.1 %	4.8 %
Case II	11	0.112	0.122	12.8 %	2.8 %

5.4.3 PSO algorithm

The gradient free algorithm performed extremely well in the case of ISTP; ***very low errors in density and pressure were observed***, owing to the fact that the cost function is not perfectly convex and that this algorithm is specially tailored for non convex type cost functions. However ***the algorithm was computationally expensive, taking up high CPU time in comparison to other algorithms.*** As explained in section 4.6 the algorithm involves initiating some particles and altering their position till every particle is in close vicinity of a single point, two cases were analyzed, one with 10 particles (Figure 5.6) and one with 20 particles (Figure 5.5). Table 5.4 below sums up the result from the two cases

TABLE 5.4: Particle swarm algorithm

Particles	Iteration	Initial Pressure $\psi_1 = p_r$	Initial Density $\psi_2 = \rho_r$	Error in ψ_1	Error in ψ_2
10	170	.10631	.13151	6.31 %	4.8 %
20	46	.10633	.13021	6.33 %	4.16 %

To explain visually how these case acquired convergence Figure 5.5 and Figure 5.6 are presented. It can be noticed that with each iteration particles move towards the minimum $\overline{\psi}$ at (0.1,0.125), represented by a black circle. Notice that in the last iteration, i.e. when the algorithm converged, each particle is inside the minima circle, giving unanimous value of ψ and this ψ is nothing but the required $\overline{\psi}$.

The tests were conducted by running the algorithm start and again, it was revealed that results remained the same always making ***PSO a stable algorithm for the case of ISTP. CPU time taken by 20 particles was comparatively lower than the 10 particle one making it a better choice.***

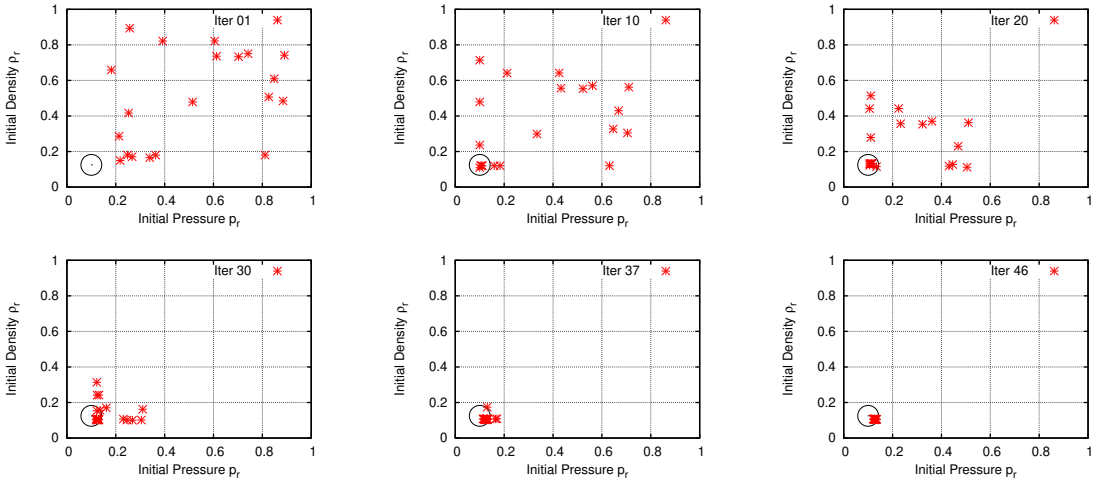


FIGURE 5.5: Particle swarm optimization with 20 particles: from top left 1, 10, 20, 30, 37 and 46^{th} iteration

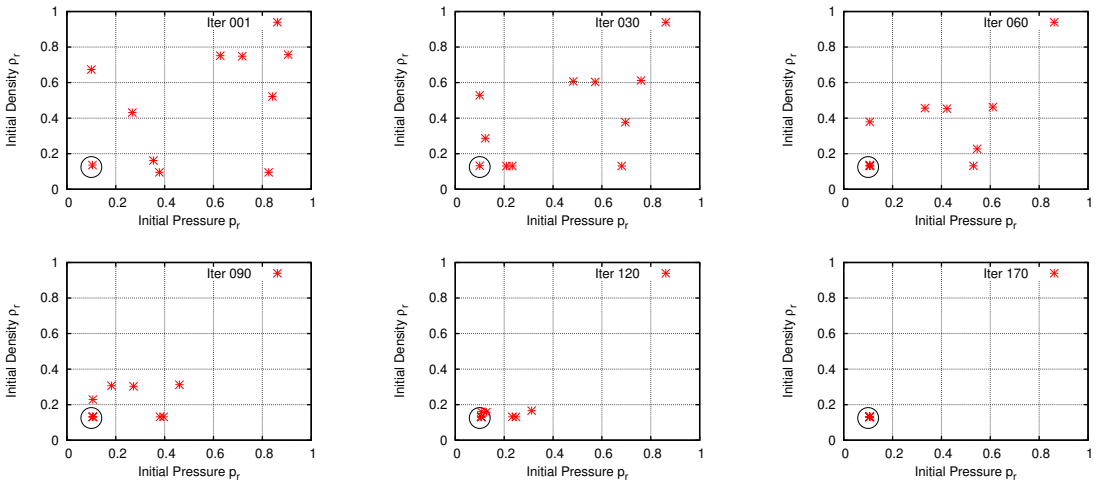


FIGURE 5.6: Particle swarm optimization with 10 particles: from top left 1,30,60,90,120 and 170^{th} iteration

5.4.4 Steepest gradient method

The gradient based method ***proved efficient for the case of ISTP; producing satisfying results with very less number of iterations***. For most of the cases with different initial guess ψ^0, the algorithm converged in 4-15 iterations. Some of the results are presented in Figure 5.7. Owing to the fact that cost function was not perfectly convex, it can be noticed the ***algorithm is not stable***. It can be noticed that all three ***different tests presented follow different path towards the minimum***. Also note, the fact that not only the algorithm is not stable but also test 3 did not output satisfactory results. It was also established that the ***algorithm produced different results ($\overline{\psi}$) with different initial guess values (ψ^0), but in most of the cases the output remained in the vicinity of the desired result***. It has to be emphasized that the ***algorithm should be run many times in order to get the correct value of*** $\overline{\psi}$.

Table 5.5 summarizes the three test results that were performed using steepest gradient algorithm with three different initial guess values. Note that all of them result in different values of $\overline{\psi}$, test 1 being the best case with test 3 the worst.

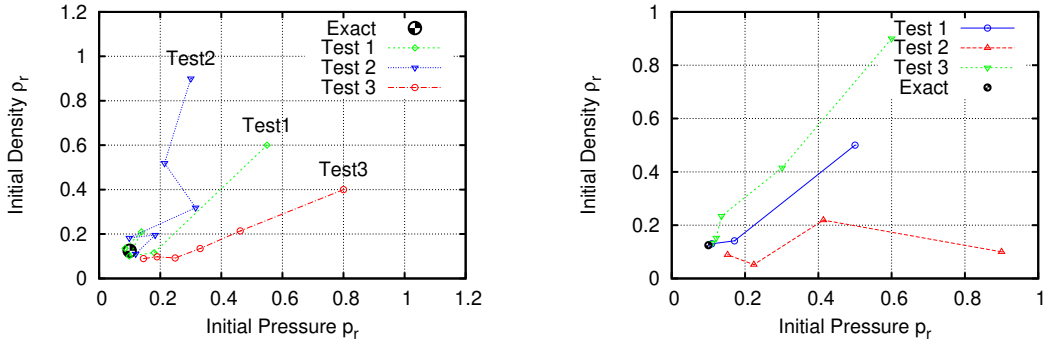


FIGURE 5.7: Convergence in SG and CG algorithms: Left- three tests for SG algorithm; Right- three tests for CG algorithm

TABLE 5.5: Steepest Gradient algorithm results

	Iteration	Initial guess ψ^0	Minimum $\overline{\psi}$	Error in $\psi_1(p_r)$	Error in $\psi_2(\rho_r)$
Test 1	4	$[0.55 \quad 0.6]^T$	$[0.085 \quad 0.134]^T$	15.0 %	7.2 %
Test 2	5	$[0.30 \quad 0.9]^T$	$[0.119 \quad 0.110]^T$	19.0 %	12.2 %
Test 3	5	$[0.8 \quad 0.4]^T$	$[0.145 \quad 0.089]^T$	45.6 %	28.8 %

5.4.5 Conjugate Gradient Method

As CG method also falls under class of gradient type optimizer, the method showed similar characteristics to steepest gradient method, only that ***the number of iterations to converge were less than that of steepest gradient method. This algorithm also lacked stability, producing different results ($\overline{\psi}$) with different initial values(ψ^0).*** To improve the algorithm ψ was constrained within the available limits [0,1]. Although for certain initial guess values of ψ^0 the algorithm converged in 2 iteration, but in general the algorithm converged in 2-10 iterations, some of the results are given in Table 5.6 and visualized in Figure 5.7.

TABLE 5.6: Conjugate Gradient algorithm results

	Iteration	Initial guess ψ^0	Minimum $\overline{\psi}$	Error in $\psi_1(p_r)$	Error in $\psi_2(\rho_r)$
Test 1	2	$[0.5 \quad 0.5]^T$	$[0.109 \quad 0.129]^T$	9.9 %	3.2 %
Test 2	3	$[0.9 \quad 0.1]^T$	$[0.153 \quad 0.089]^T$	53.3 %	28.8 %
Test 3	4	$[0.6 \quad 0.9]^T$	$[0.110 \quad 0.133]^T$	10.1 %	6.4 %

5.5 Summary of two parameter estimation

To summarize all the results that have been presented in section 5.4, Table 5.7 has been presented and to visualize the comparison of how different algorithms perform for the two parameter estimation ISTP, Figure 5.8 is presented. Each point on Figure 5.8 represents averaged output of three test runs in each algorithm.

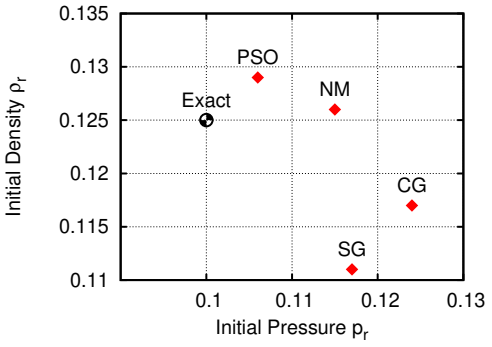


FIGURE 5.8: Visual Summary of two parameter analysis

This sums up the discussion ***GN being the worst algorithm for the ISTP and PSO being the best choice***, although it comes with a high computational cost. To be economical computationally and equally also produce good result simplex method (Nedler-Mead method) is not that bad of a choice. ***Gradient type methods form a good choice but care needs to be taken if the acquired ψ is $\overline{\psi}$ the algorithm needs to be run over and again.***

To provide the rating of how the alogrithms performed Table 5.7 has been presented. As it can be noticed from the Table 5.7 that, PSO does not perform well on iteration count and

computational time, while in on counter CG only performs well in these two areas. It would be a wise option to combine these two algorithms. In light of this two hybrid algorithms are tailored in order to solve the ISTP more efficiently.

- PSO1CG2 *(first PSO then CG)*
- PSO2CG1 *(first CG then PSO)*

TABLE 5.7: Rating of different algorithms for ISTP

	GN	**NM**	**PSO**	**SG**	**CG**
iterations	☆☆☆☆☆	★★★★☆	★★☆☆☆	★★★★★	★★★★★
Error	☆☆☆☆☆	★★★☆☆	★★★★★	★★☆☆☆	★★☆☆☆
Stability	☆☆☆☆☆	★★★★☆	★★★★★	★★★☆☆	★★☆☆☆
CPU time	☆☆☆☆☆	★★★★☆	★★☆☆☆	★★★★☆	★★★★★

5.6 Hyprib algorithm

As mentioned above this kind of algorithm combines the advantages of zero order and first order optimizers for ISTP.

5.6.1 PSO1CG2

The algorithm involves running the PSO algorithm first followed by CG algorithm. Here PSO algorithm is run for a particular number of iterations (iteration when the swarm is confined in a close vicinity). At the stopping PSO iteration from the location of all swarm particles ψ_i an average location is deduced ψ_{avgPSO}. This new location serves as initial guess for the CG algorithm $\psi^0 = \psi_{avgPSO}$. The fact that at the stopping iteration for PSO, the swarm particles are in close vicinity suggests that particles are in an area that will be convex like. Any guess value in this convex like area should form a descent guess for CG method, this forms the motivation for the algorithm PSO1CG2.

A test of PSO1CG2, a case run with 20 particles for PSO is presented in this section. Figure 5.9 presents the initial iteration, the stopping iteration and the average ψ value for PSO. Further CG algorithm run using ψ_{avgPSO} as initial value. The results of this algorithm is illustrated in Table 5.8.

TABLE 5.8: Hybrid optimizer PSO1CG2

Iteration (*PSO+CG*)	Initial Pressure $\psi_1 = p_r$	Initial Density $\psi_2 = \rho_r$	Error in ψ_1	Error in ψ_2
24+7=31	.1063	.138	6.3 %	10.4 %

It can be concluded that the ***algorithm performs much better than the two algorithms individually***. As seen in section 5.4.3 PSO algorithm with 20 particles ran alone, converged in 46 iterations but with PSO1CG2 PSO needs to be run only 24 times saving 22 iterations

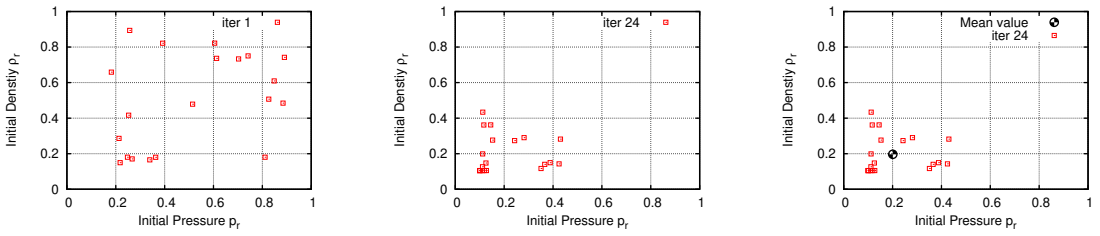


FIGURE 5.9: Hybrid optimizer PSO1CG2 showing initial guess, stopping guess and derived mean value

while producing almost the same results. It should also be noted that in actual $(22 \times 20 = 440)$ ***iterations in forward model are saved*** owing to the fact that each particle needs running of forward model once. But also it should be noted that conjugate gradient method did not converge in 2 iterations as it was expected to, this suggests that cost function is non convex even from the new starting guess ψ_{avgPSO}. For different tests run ***the algorithm in general was not fully stable.***

5.6.2 PSO2CG1

As mention above the cost function remains non convex even in the vicinity of the required $\overline{\psi}$, PSO2CG1 algorithm forms a good option in order to find the minimum. PSO2CG1 involves running the CG algorithm first followed by the PSO algorithm. CG algorithm outputs its minimum ψ_{minCG}, keeping this minimum at the center a small boxed shaped local parameter space is established around it. In this local parameter space PSO algorithm is applied.

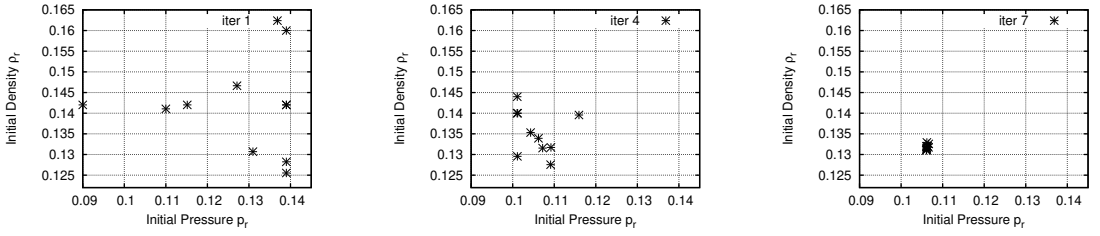


FIGURE 5.10: Hybrid optimizer PSO2CG1: particles initialized in local parameter space and converging in 7 iterations; From left- 1,4 and $7^{th} iteration$

A test case of 10 particle PSO algorithm combined with CG method is presented in this section. Running CG method for a random guess generated $\overline{\psi} = (.117, .1435)$ this guess is then converted to local parameter space and PSO algorithm with 10 particles is used. This can be seen in Figure 5.10, note that in the figure every particle is initialized in the local parameter space.

TABLE 5.9: Hybrid optimizer PSO2CG1

Iteration (CG+PSO)	Initial Pressure $\psi_1 = p_r$	Initial Density $\psi_2 = \rho_r$	Error in ψ_1	Error in ψ_2
6+7=13	.106	.131	6.1 %	4.16 %

Table 5.9 presents results of this algorithm. The ***algorithm ended up being stable, computationally inexpensive and with low error.*** To note PSO for this case converged in 7

iterations contrary to 170 iteration when used alone, ***saving*** $((170 - 7) \times 10 = 1630)$ ***forward computations***. Hence making this method ***most stable, economical and efficient for ISTP.***

To present summary of the two hybrid algorithms in comparison to PSO and CG methods Table 5.10 has been presented.

TABLE 5.10: Rating of hybrid algorithms

	CG	**PSO**	**PSO1CG2**	**PSO2CG1**
iterations	★★★★★	★★☆☆☆	★★★★☆	★★★★★
Error	★★½☆☆	★★★★★	★★★★☆	★★★★★
Stability	★★☆☆☆	★★★★★	★★★★☆	★★★★★
Computational time	★★★★★	★★☆☆☆	★★★☆☆	★★★★★

5.7 Conclusion

In the field of inverse computational science, problems concerning fluid mechanics still remain in its infancy. In order to explore deeper in this underdeveloped area, inverse problem in shock tubes, a nonlinear fluid phenomenon governed by non linear hyperbolic PDE, has been analyzed and then solved. Traditionally inverse problems are being extensively used with elliptic PDEs (Thermal studies) mostly concerning use of matrix system inversion; in this project an effort has been made to step out of this common notion and research inverse problems in a counter field (non matrix and non elliptic). Literature review on the subject of inverse fluid dynamics concluded that not much work has been done in this field and that there is a need to analyze *how and what* in inverse problem theory performs well with fluid systems. In light of this, the solution of ISTP has presented certain closures that may be followed in future to solve other inverse fluid system problems.

It was concluded that cost functions are not well behaved (non convex) in the field of inverse fluid dynamics. In case of ISTP, the analysis revealed that 2D cost functions remain always non linear. Tests should be run as prerequisite to inverse problem solutions in order to identify the cost function that is least non linear, in real world this is one out of the several other issues that governs the position of the sensor. It was established that pressure based cost functions perform the best among other cost functions. As a rule of thumb the choice of cost function in ISTP has been established, experimental data should be chosen from a point that does not interact with the shock. A conclusion was drawn that shock interacting with the data point leads to highly non linear cost functions, a bad news for any inverse problem solution.

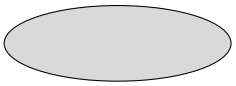
It can be emphasized that right choice of optimization algorithm can make or break the solution. Every optimization algorithm carried out with ISTP amounted to different outputs hence care should be taken to carry out the selection wisely. Due to non convexity in the cost it can be concluded that gradient free optimizers form the best choice to solve the inverse fluid system problems. The in house PSO1CG2 and PSO2CG1 being the best algorithms to perform in case of ISTP suggests that optimization algorithms need to be well tailored for every specific case of inverse fluid problem in order to enhance their efficiency.

Appendix A

Flow Charts

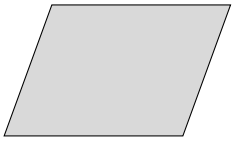
A.1 Terminology

1. Start and Stop:

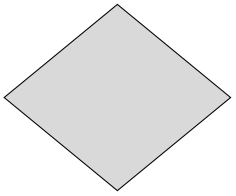


2. Process:

3. input and output:



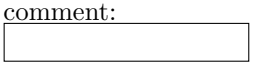
4. decision:



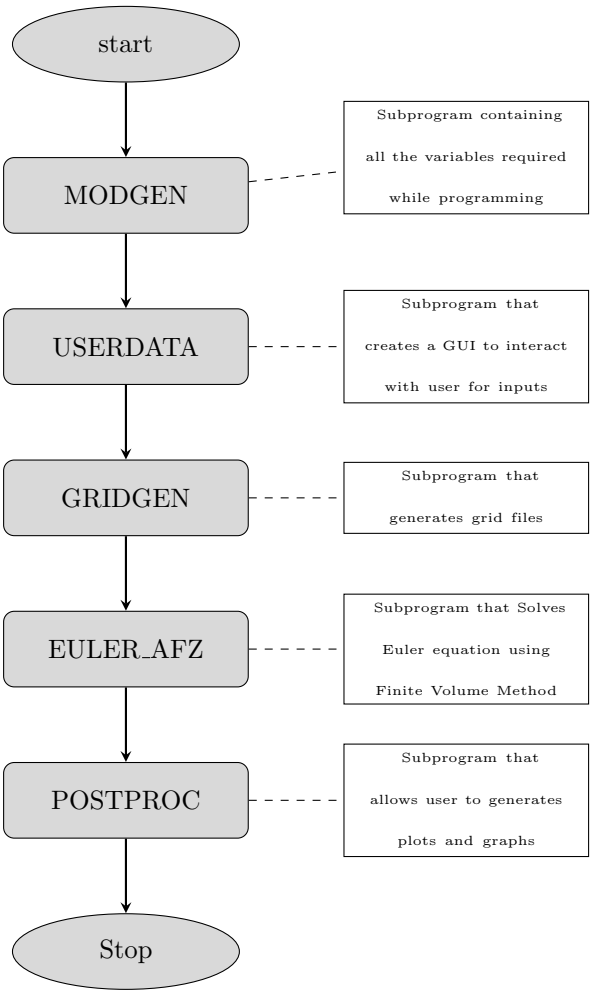
5. subroutiens and programs:

6. Do and for loops:

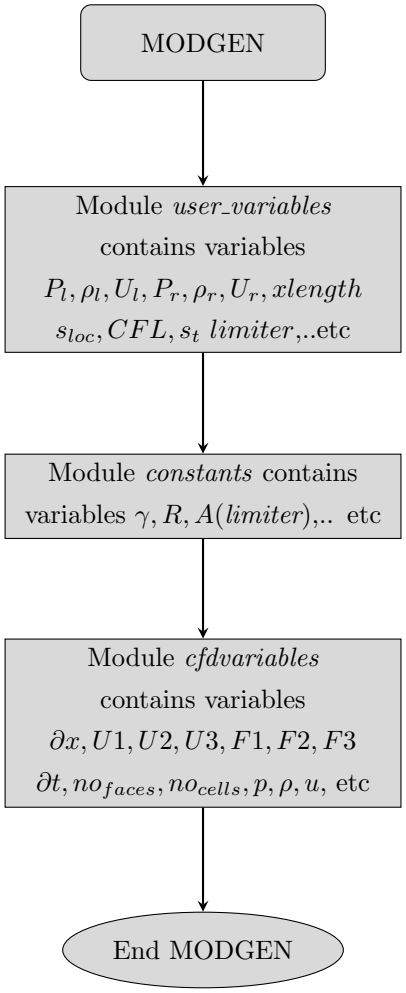
7. comment:



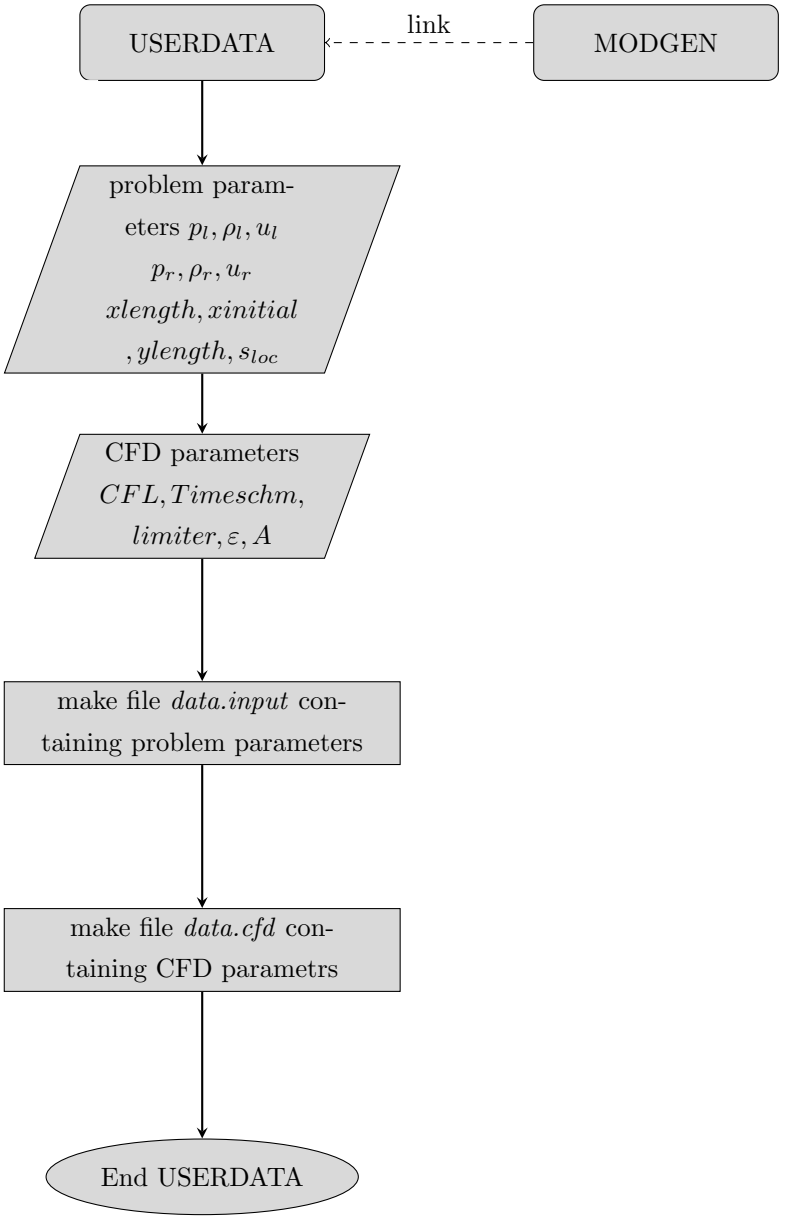
A.2 CFD EULER code



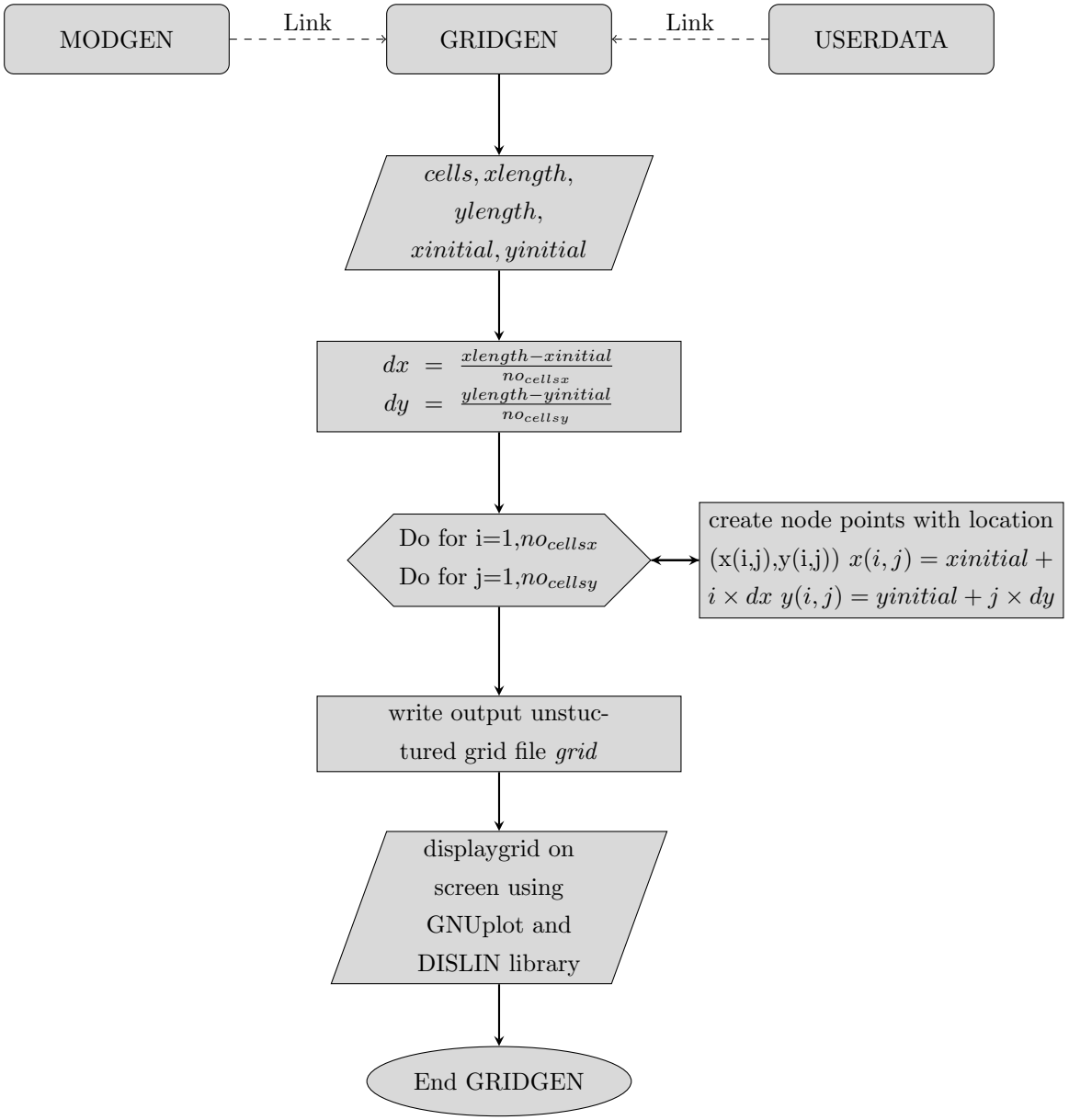
A.3 Program MODGEN



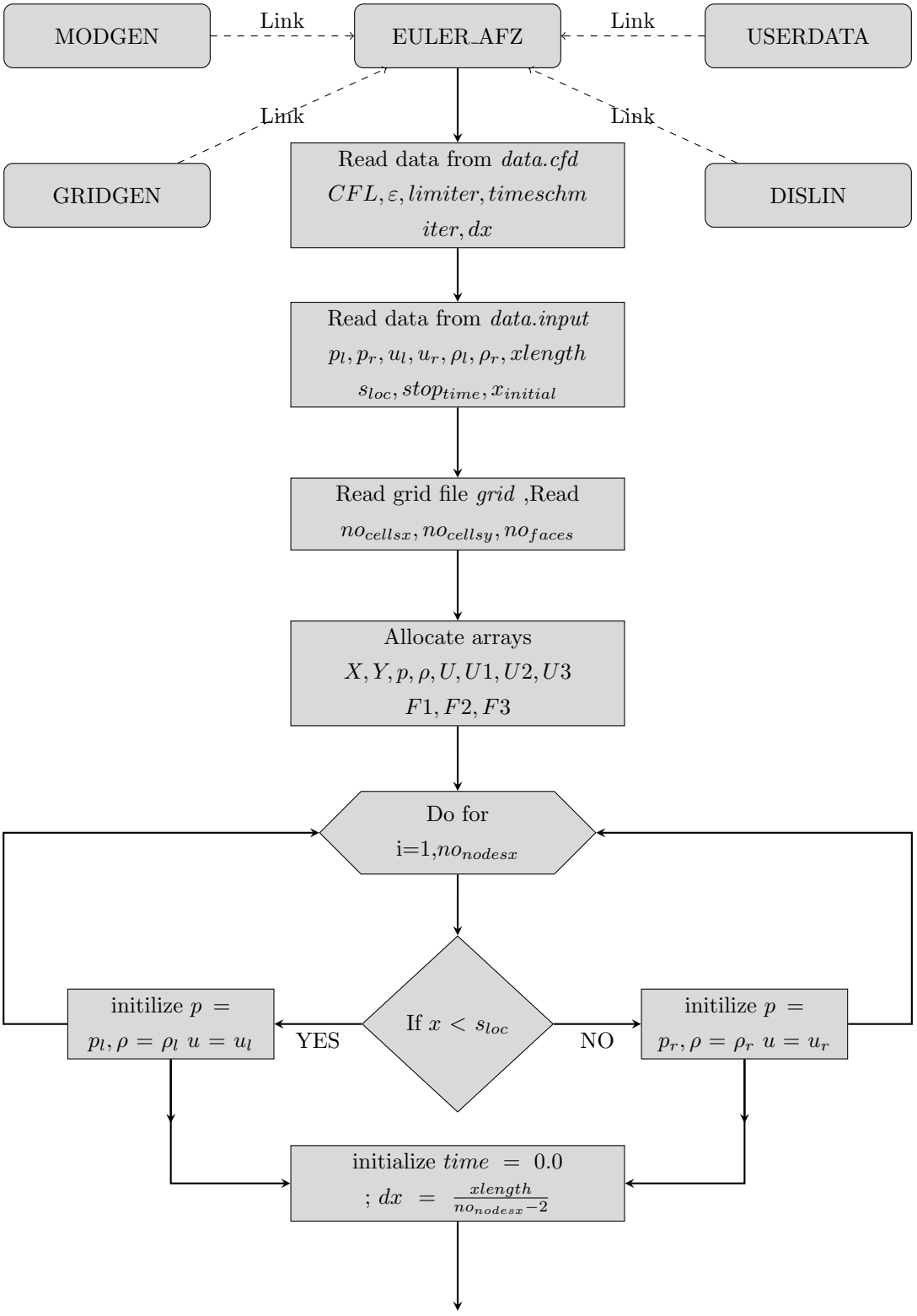
A.4 Program USERDATA

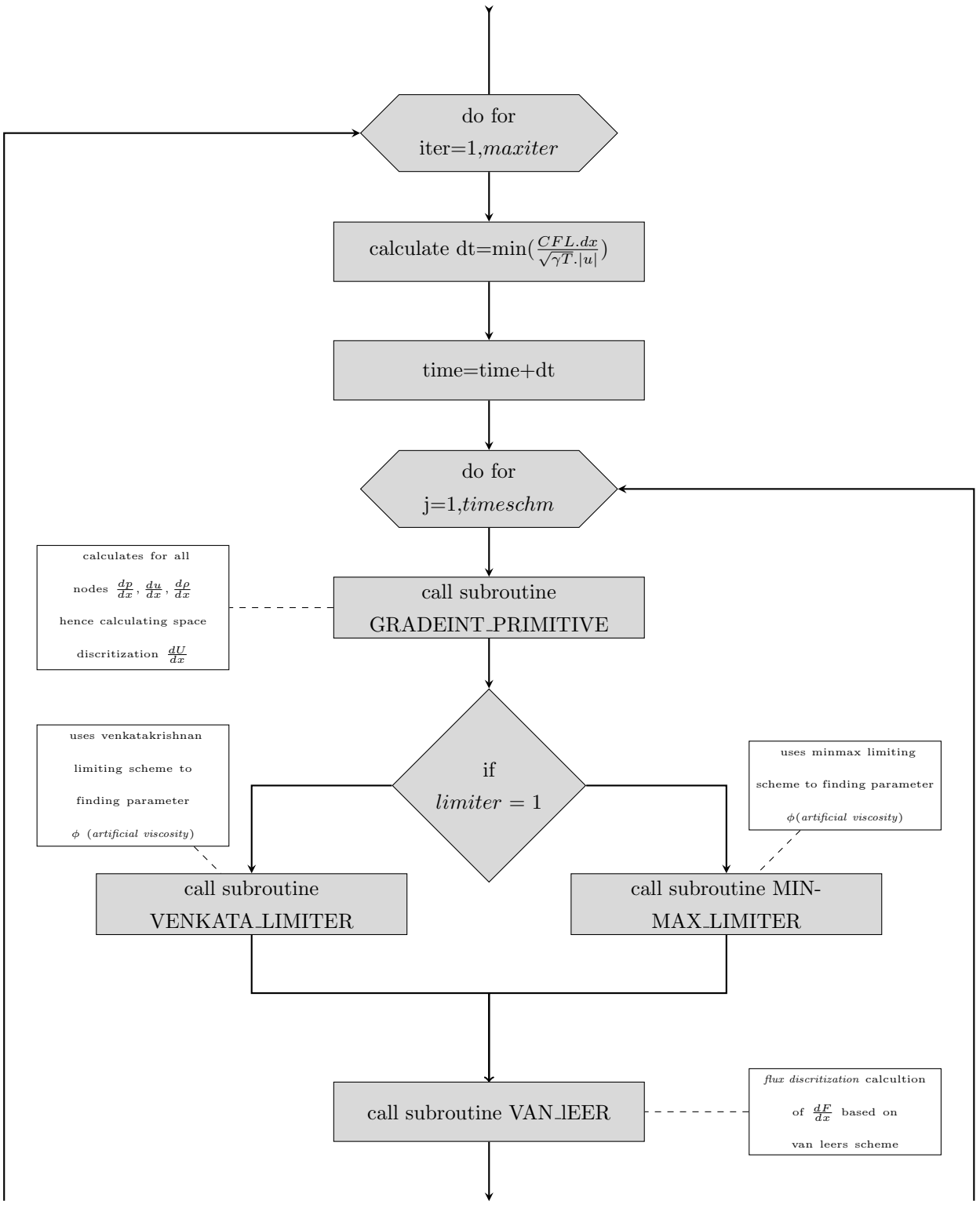


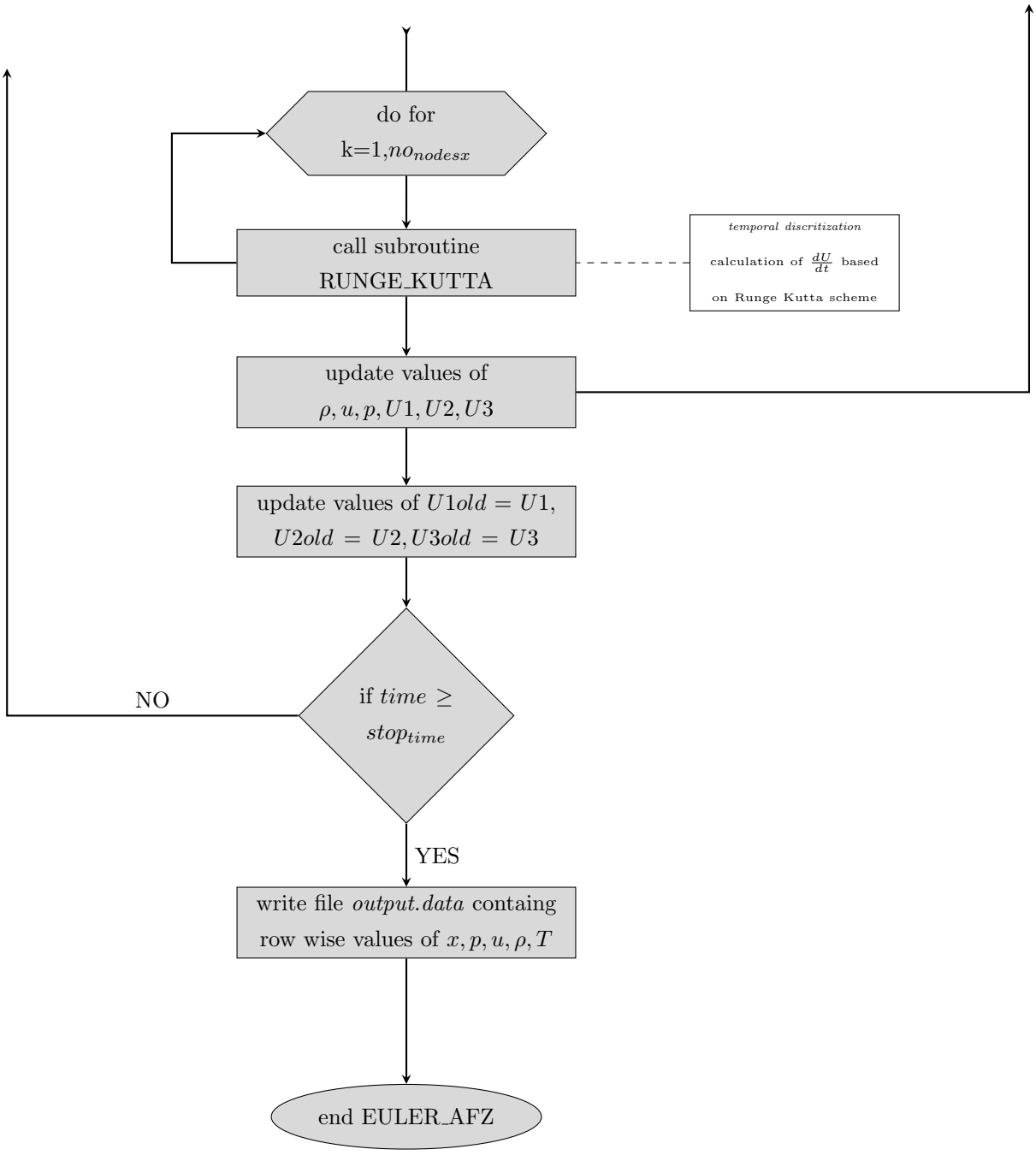
A.5 Program GRIDGEN



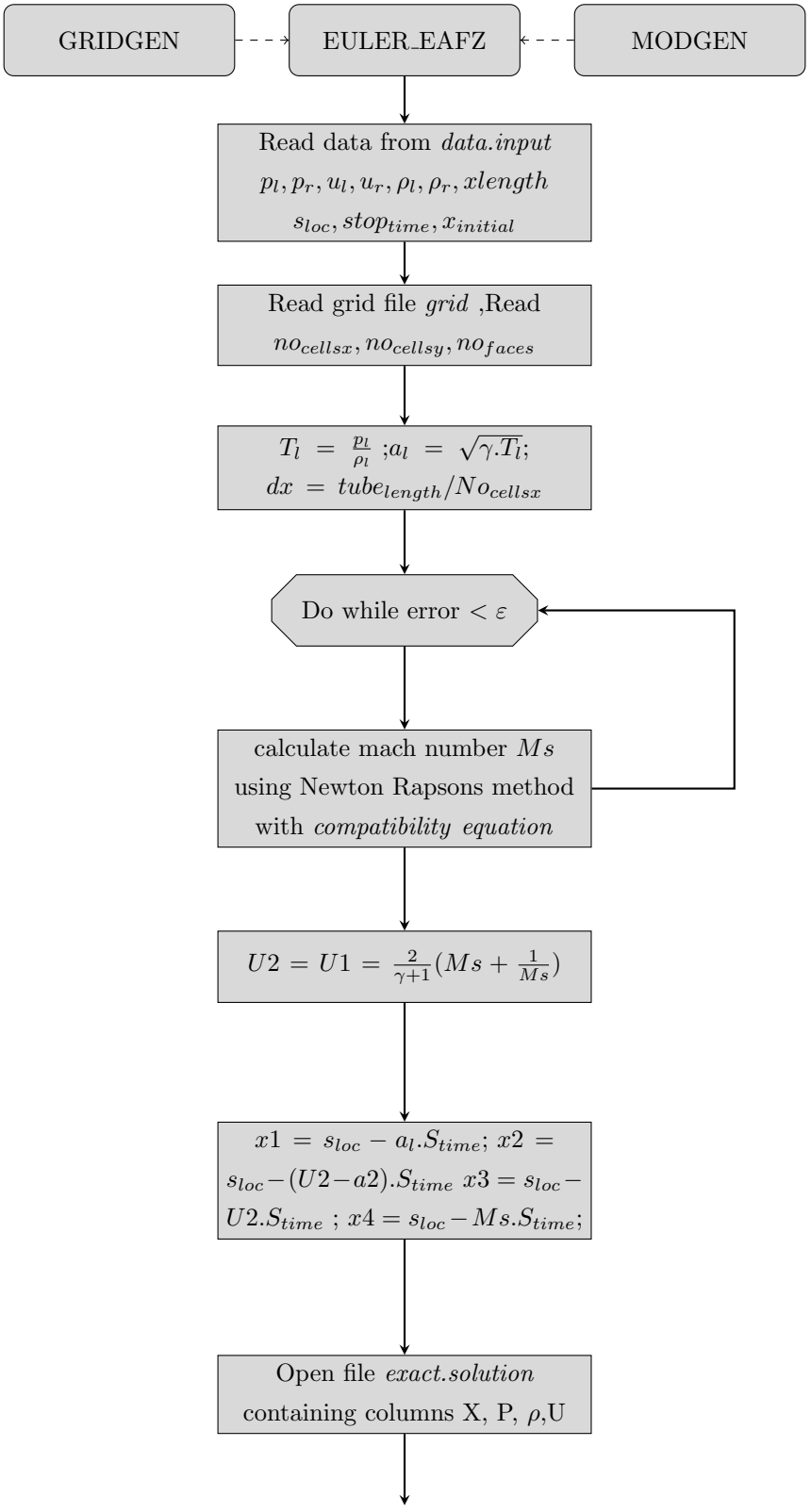
A.6 Program EULER

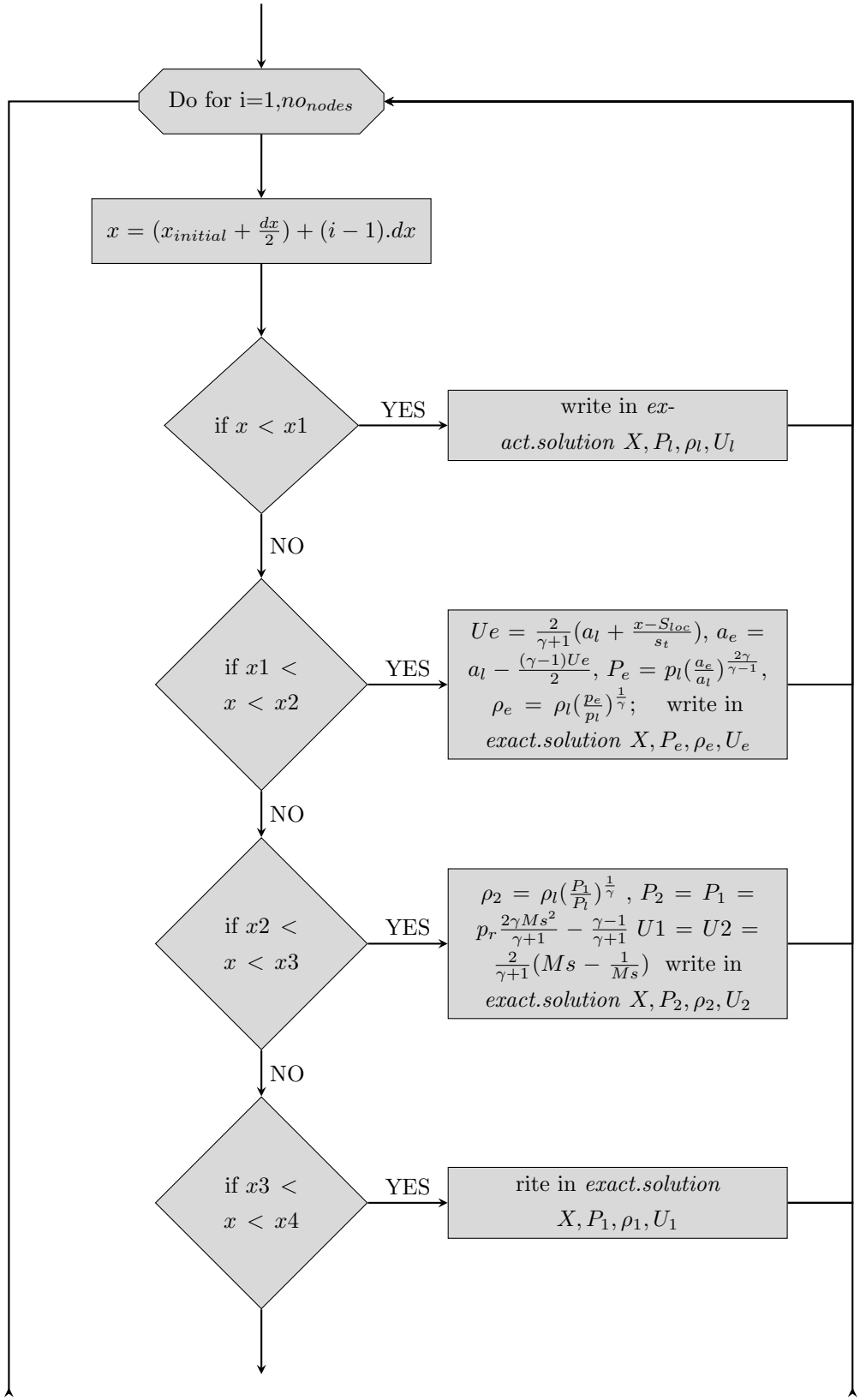


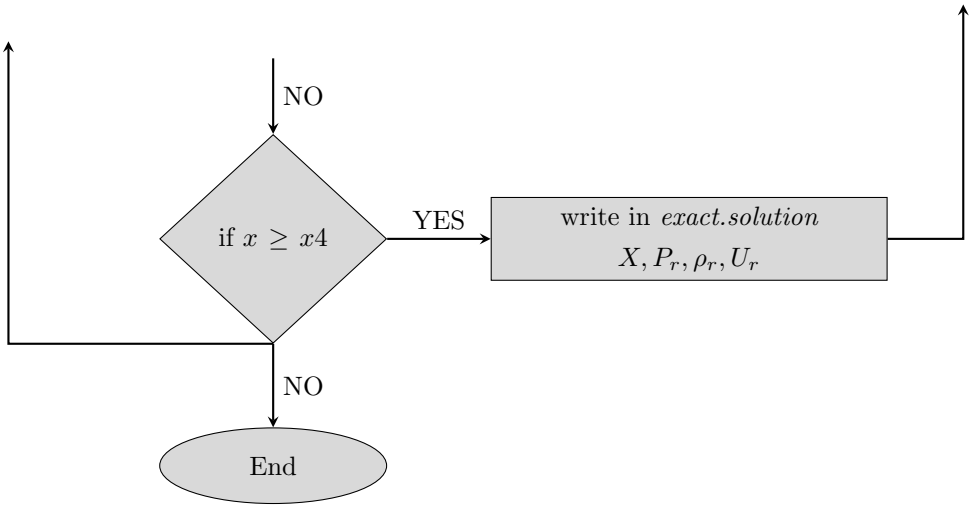

do for
iter=1,*maxiter*
calculate dt=min($\frac{CFL.dx}{\sqrt{\gamma T}.|u|}$)
time=time+dt
do for
j=1,*timeschm*
calculates for all
nodes $\frac{dp}{dx}$, $\frac{du}{dx}$, $\frac{d\rho}{dx}$
hence calculating space
discritization $\frac{dU}{dx}$
call subroutine
GRADEINT_PRIMITIVE
uses venkatakrishnan
limiting scheme to
finding parameter
ϕ (*artificial viscosity*)
if
limiter = 1
uses minmax limiting
scheme to finding parameter
ϕ(*artificial viscosity*)
call subroutine
VENKATA_LIMITER
call subroutine MIN-
MAX_LIMITER
call subroutine VAN_lEER
flux discritization calcultion
of $\frac{dF}{dx}$ based on
van leers scheme


do for
k=1,no_{nodesx}
call subroutine
RUNGE_KUTTA
temporal discritization
calculation of $\frac{dU}{dt}$ based
on Runge Kutta scheme
update values of
$\rho, u, p, U1, U2, U3$
update values of $U1old = U1$,
$U2old = U2, U3old = U3$
if $time \geq$
$stop_{time}$
NO
YES
write file *output.data* containg
row wise values of x, p, u, ρ, T
end EULER_AFZ

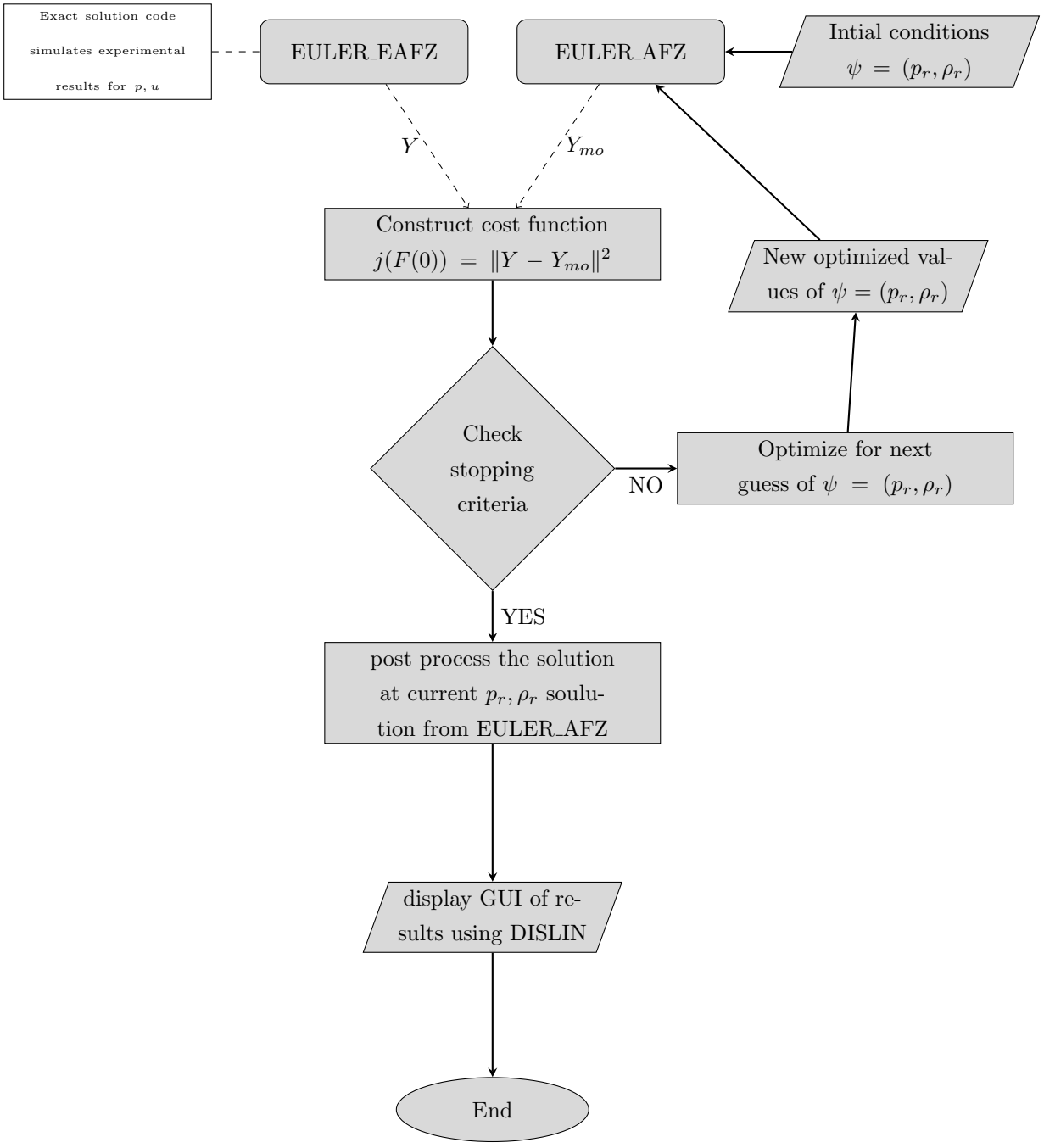
A.7 Exact solution solver- Program EULER_EAFZ




Do for i=1,no_{nodes}
$x = (x_{initial} + \frac{dx}{2}) + (i-1).dx$
if $x < x1$
YES
write in *exact.solution* X, P_l, ρ_l, U_l
NO
if $x1 < x < x2$
YES
$Ue = \frac{2}{\gamma+1}(a_l + \frac{x-S_{loc}}{s_t})$, $a_e = a_l - \frac{(\gamma-1)Ue}{2}$, $P_e = p_l(\frac{a_e}{a_l})^{\frac{2\gamma}{\gamma-1}}$, $\rho_e = \rho_l(\frac{p_e}{p_l})^{\frac{1}{\gamma}}$; write in *exact.solution* X, P_e, ρ_e, U_e
NO
if $x2 < x < x3$
YES
$\rho_2 = \rho_l(\frac{P_1}{P_l})^{\frac{1}{\gamma}}$, $P_2 = P_1 = p_r\frac{2\gamma Ms^2}{\gamma+1} - \frac{\gamma-1}{\gamma+1}$ $U1 = U2 = \frac{2}{\gamma+1}(Ms - \frac{1}{Ms})$ write in *exact.solution* X, P_2, ρ_2, U_2
NO
if $x3 < x < x4$
YES
rite in *exact.solution* X, P_1, ρ_1, U_1


NO
if $x \geq x4$
YES
write in *exact.solution*
X, P_r, ρ_r, U_r
NO
End

A.8 Inverse Solution solver- Program INVERSE_ISTP



Appendix B

Algorithms

B.1 Gauss Newton Algorithm

Algorithm 1: Gauss Newton Method

input : $Y = [y_1, y_2, y_3, ...y_{dim(Y)}]$ *experimental points,* ε *step size*
Let $k = 0, \psi^0$ *be the starting guess for parameter vector* ψ

while *stopping criteria is not met* **do**
- $k = k + 1$
- ***Run*** *Forward model to obtain* Y_{mo}
- *Error Vector* $E = Y - Y_{mo}$
- **for** *parameters* $i = 1, dim(\psi)$ **do**
  - $\psi_i^k = \psi_i^k(1 + \varepsilon)$
  - ***Run*** *Forward model to obtain* Y_{mo}^+
  - **for** *measurements* $j = 1, dim(Y)$ **do**
    - *sensitivity Matrix* $S_{ji} = \frac{Y_{mo}^+ - Y_{mo}}{\varepsilon \psi_i^k}$
- *Solve* $\delta\psi = -[S^T S]^{-1} S^T E$
- $\psi_i^k = \psi_i^k + \delta\psi$

B.2 Simplex Algorithm

Algorithm 2: Simplex Method

input : $(dim(\psi)+1) = 3$ *random values in parameter space*
Run *Forward model to obtain cost function* j *at the points*

while *stopping criteria is not met* **do**
- $k = k + 1$
- ***Sorting:*** *sort the points in descending order* $j(u) < j(v) < j(w)$
- *Mid Point vector* $M = \frac{1}{2}[u + v]$
- ***Reflection*** $[R] = 2[R] - M$
- **if** $j(R) < j(v)$ **then**
  - Perform case (i) { either reflect or extend };
- **else**
  - Perform case (ii){ either contract or shrink };
- ***Begin*** *case(i)*
- **if** $j(u) < j(R)$ **then**
  - *Replace* W *with* R;
- **else**
  - ***Expansion*** $E = 2R - M$;
  - **if** $j(E) < j(w)$ **then**
    - *Replace* w *with* E;
  - **else**
    - *Replace* w *with* R;
- ***End*** *case(i)*
- ***Begin*** *case(ii)*
- **if** $j(R) < j(w)$ **then**
  - *Replace* W *with* R;
- **else**
  - ***Contraction:*** $C1 = \frac{1}{2}[R + M]; C2 = \frac{1}{2}[W + M]$;
  - **if** $j(C) < j(w)$ **then**
    - *Replace* w *with* C;
  - **else**
    - ***Shrinkage:*** $u = M; w = \frac{1}{2}[w + v]$;
- ***End*** *case(ii)*

B.3 Particle swarm optimization

Algorithm 3: Particle Swarm Algorithm

input : $C1, C2, \omega$ *Scalar constant*
for *particle* $k = 1, maxparticles$ **do**
  Initialize position $x_k^p = [Y_1, Y_2, .., Y_{dim(\psi)}]$
  P.best *particles best position* $p.best_k = x_k^p$
  G.best *global best position* $j(G) > j(p.best_k)$
  Initialize velocity $V_k^p = [\delta Y_1, \delta Y_2, .., \delta Y_{dim(\psi)}]$

while *stopping criteria is not met* **do**
  for *particle* $k = 1, maxparticles$ **do**
    Generate two random numbers based on normal distribution $(u(0,1))$ $R1, R2$
    $V_k^p = \omega V_k^p + C1R1(p.best_k - x_k^p) + C2R2(G - x_k^p)$
    $x_k^p = x_k^p + V_k^p$
    update P.best *if* $j(x_k^p < j(p.best_k))$ ***update G.best*** *if* $j(x_k^p < j(G))$

B.4 Steepest Gradient Method

Algorithm 4: Steepest Gradient Method

input : $Y = [y_1, y_2, y_3, ...y_{dim(Y)}]$ *experimental points*,m, ε *Scalar constant*
$k = 0$,*initialize* $\phi = [\phi_1, \phi_2, .., \phi_{dim(\psi)}]$ *Vector with random guess values*
Run *Forward model to obtain* Y_{mo}
$j_{old} = \frac{1}{2}[Y - Y_{mo}]^2$

while *stopping criteria is not met* **do**
  $k = k + 1$
  $\alpha = \frac{1}{mk}$
  for *parameter* $i = 1, dim(\psi)$ **do**
    $\psi_i = \phi_i + \varepsilon$
    Run *Forward model to obtain* Y_{mo}
    $j_{new} = \frac{1}{2}[Y - Y_{mo}]^2$
    $\nabla(\psi_i) = \frac{j_{new} - j_{old}}{\varepsilon}$
  for *parameter* $i = 1, dim(\psi)$ **do**
    $\phi_i = \phi_i - \alpha \frac{\nabla(\psi_i)}{\|\nabla(\psi)\|}$
  $\psi = \phi$
  Run *Forward model to obtain* Y_{mo}
  $j_{old} = \frac{1}{2}[Y - Y_{mo}]^2$

B.5 Conjugate gradient Method

Algorithm 5: conjugate Gradient Method

input : $Y = [y_1, y_2, y_3, ...y_{dim(Y)}]$ *experimental points,*ε *Scalar constant*
$k = 0$,*initialize* $\phi = [\phi_1, \phi_2, .., \phi_{dim(\psi)}]$ *Vector with random guess values*
$\psi_0 = \phi$
Run *Forward model to obtain* Y_{mo}
$j_{old} = \frac{1}{2}[Y - Y_{mo}]^2$
Calculate $\nabla j(\psi_0)$
Direction of descent $d_{old} = -\nabla j(\psi_0)$

while *stopping criteria is not met* **do**
  $k = k + 1$
  if $k = 1$ **then**
    $\beta = 0$;
  else
    $\beta = \frac{\nabla j(\psi^k)\nabla j(\psi^k)}{\nabla j(\psi^{k-1})\nabla j(\psi^{k-1})}$;
  for *parameter* $i = 1, dim(\psi)$ **do**
    $\psi_i = \phi_i + \varepsilon$
    Run *Forward model to obtain* Y_{mo}
    $j_{new} = \frac{1}{2}[Y - Y_{mo}]^2$
    $\nabla j_i = \frac{j_{new} - j_{old}}{\varepsilon}$
  with $\psi = \phi$ *calculate* $\nabla j(\psi^k)$
  $d_{new} = -\nabla j(\psi^k) + \beta d_{old}$
  $\psi = \phi$
  Run *Forward model to obtain* Y^-_{mo}
  $\psi = \phi + \epsilon \frac{d_{new}}{\|d_{new}\|}$
  Run *Forward model to obtain* Y^+_{mo}
  $Y' = \frac{Y^+ - Y^-}{\sqrt{\sum(\psi - \phi)}}$
  $\alpha = -\frac{Y'(Y^- - Y)}{Y'Y'}$
  $\psi^{k+1} = \phi + \alpha d_{new}$
  $\phi = \psi$
  $d_{old} = d_{new}$
  calculate gradient $\nabla j(\psi^k)$

B.6 Newton Rapsons Algorithm

Algorithm 6: Newton Rapsons Algorithm for calculating M_s

input : $k = 0$; $M_s^k = 1.0$ initial guess value for Mach number
input : ε convergence criteria
calculate the scalar value of compatibility equation $f(M_s^k)$

while *stopping criteria is not met* **do**
  $k = k + 1$
  calculate the scalar value of derivative compatibility equation $f'(M_s^k)$
  $M_s^k = M_s^{k-1} - \frac{f(M_s^k)}{f'(M_s^k)}$ *calculate the scalar value of compatibility equation* $f(M_s^k)$
  if $f(M_s^k) \leq \varepsilon$ **then**
    Exit iterations;
  else
    continue the iteration ;

B.7 Algorithm for code EULER_AFZ

Algorithm 7: Algorithm for program EULER_AFZ

Load *Modules MODGEN , DISLIN , USERDATA*
Read *Grid File from GRIDGEN*
Initialize *grid points with initial value of primitive variables* p, ρ, u
Initialize *grid points with states* $U1 = f(p, \rho, u)$, $U2 = f(p, \rho, u)$, $U3 = f(p, \rho, u)$

while *Time* $t \leq stop_{time}$ **do**
  calculate δt *with* $\delta t = f(CFL, u, a)$
  $t = t + \delta t$
  calculate gradients $(\frac{\partial u}{\partial x}, \frac{\partial \rho}{\partial x}, \frac{\partial p}{\partial x})$ *at each node*
  Reconstruct *the gradients using Venkatakrisnan limiting*
  calculate fluxes $F = f(U)$ *using Van Leers scheme*
  calculate states at next time step $t = t + \delta t$ *using* 4^{th} *order Runge Kutta scheme*
  Update *all the states* (U) *and primitives* (p, ρ, u) *with new values*
output the results in a text file

Appendix C

EULER Solutions- Exact and Numerical

C.1 Analytic solution for Euler equation

Analytical solution of the flow inside the shock tube follows the mathematics and physics that has been discussed in chapter 2. Flow is divided into four uniform sections i.e., the right (R) and the left (L) region maintaining constant parameters (velocity, density, temperature and pressure) that existed as initial conditions and regions in between them denoted by region (1) and (2).

To analyze these regions in (x, t) plane is considered see Figure C.1. All waves at time $t = 0$ are centered at the diaphragm ($x = x_0, \quad t = 0$). Shock and the contact discontinuity propagate with constant velocities moving right and are displayed as lines in (x, t) plane. Expansion waves stretches in zone (E) inside which the flow parameters change linearly.

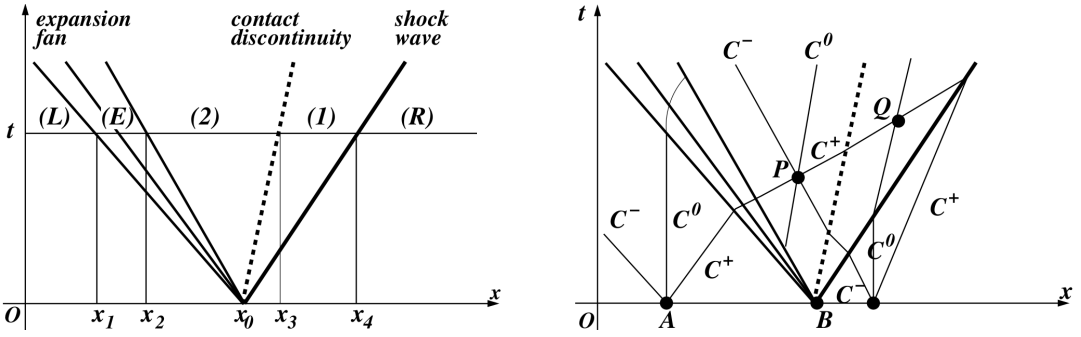


FIGURE C.1: Diagram showing (x, t) plane of the analytical solution for shock tube (left). Characteristics used in calculating the analytical solution (right).

Dimensionless parameters in regions (R) and (L) are

$$\text{Region (R):}\quad u_r = 0,\quad p_r = 1/\gamma,\quad \rho_r = 1,\quad T_r = 1/\gamma,\quad a_r = 1 \tag{C.1}$$

$$\text{Region (L):}\quad u_r = 0,\quad p_l,\quad \rho_r,\quad T_r,\quad a_r\quad \text{(given usually)} \tag{C.2}$$

As Euler equation was proven to be hyperbolic in nature (refer section 2.5), there is a need to take into account propagation of information along the characteristic lines, hence the following mathematics is applied across the discontinuities

1. Discontinuity due to normal shock standing between regions (R) and (1) is governed by Rankine Hugoniot Equations (for example, see, Hirsch [1]):

$$\frac{p_1}{p_r} = \frac{2\gamma}{\gamma+1}M_s^2 - \frac{\gamma-1}{\gamma+1} \tag{C.3}$$

$$\frac{\rho_r}{\rho_1} = \frac{2}{\gamma+1}\frac{1}{M_s^2} - \frac{\gamma-1}{\gamma+1} \tag{C.4}$$

$$u_1 = \frac{2}{\gamma+1}\left(M_s^2 - \frac{1}{M_s^2}\right) \tag{C.5}$$

   Here u_s is the dimensionless speed of Normal shock and M_s is the corresponding Mach number.

2. Contact discontinuity standing between regions (1) and (2) is only discontinuity of Temperature and density, pressure and velocity remain unaffected by this discontinuity. So for this

$$u_2 = u_1,\quad p_1 = p_2 \tag{C.6}$$

3. To have a link of parameters in regions (2) and (L), consider a point $P(x,t)$ in region (2) the characteristic lines passing through this point can be seen in Figure C.1. It can be observed that only two characteristic curves C^0 and C^+ run across the expansion fans in order to exchange information from region (L). Using $u_l = 0$ and the expression for invariant for the two curves C^0 and C^+, the following relation is obtained (for more details, see, [21])

$$\frac{\rho_2}{\rho_l} = \left(\frac{p_2}{p_l}\right)^{1/\gamma},\quad u_2 = \frac{2}{\gamma-1}(a_l - a_2) \tag{C.7}$$

4. By combining all the equations mentioned above, *compatibility equation* for M_s can be obtained (for more details, see, [21])

$$M_s - \frac{1}{M_s} = a_l\frac{\gamma+1}{\gamma-1}\left\{1 - \left[\frac{p_r}{p_l}\left(\frac{2\gamma}{\gamma+1}M_s^2 - \frac{\gamma-1}{\gamma+1}\right)\right]^{\frac{\gamma-1}{2\gamma}}\right\} \tag{C.8}$$

   Generally some iterative algorithm (Newton-Rapsons method) is used in order to solve the compatibility equation for the values of M_s. The value of M_s obtained by the iterative algorithm can be used to obtain all parameter values in the region space (1) and (2).

For analytical solution abscissa values x_1, x_2, x_3, x_4 in Figure C.1 at any time t should be known. In order to do so we proceed as follows:

- *calculation for x_1 and x_2*: The left running characteristic curve C^- originating from point B forms the left boundary for expansion fan area (E) (see Figure 2.4), this curve belongs to region (L) i.e. slope of curve will be given by $dx/dt = a_l$. Similarly the right running characteristic curve C^+ that also originates from point B forms the right boundary for expansion fan area (E), this curve belongs to region (1) i.e. slope of curve will be given by $dx/dt = u_2 - a_2$. Hence values of x_1 and x_2 are given as

$$x_1 = x_0 - a_l t, \quad x_2 = x_0 + (u_2 - a_2)t \tag{C.9}$$

- *calculation for x_3*: As it was mentioned earlier the contact discontinuity travels with a constant speed $u_1 = u_2$, so

$$x_3 = x_0 + u_2 t \tag{C.10}$$

- *calculation for x_4*: Since the normal shock also propagates with a constant dimensionless velocity $u_s = M_s$, so

$$x_4 = x_0 + M_s t \tag{C.11}$$

In order to complete the analytical flow analysis, mathematics flow inside the expansion fan area needs to be developed: considering a point $m(x,t)$ in the region (E), here $x_1 \leq x \leq x_2$. As this point belongs to characteristic curve C^- originating form B, slope at the point will be given by

$$\frac{(x - x_0)}{t} = u_e - a_e$$

where u_e and a_e are the velocity and speed of sound in the expansion fan area (E). Using characteristic curve C^+ that passes from region (L),

$$a_l = a_e + \frac{\gamma - 1}{2} u_e$$

Combining the two relations flow solution in region (E) can be obtained

$$u_e = \frac{2}{\gamma + 1}\left(a_l + \frac{x - x_0}{t}\right) \tag{C.12}$$

$$a_e = a_l - (\gamma - 1)\frac{u_e}{2} \tag{C.13}$$

$$p_e = p_l \left(\frac{a_e}{a_l}\right)^{\frac{2\gamma}{\gamma - 1}} \tag{C.14}$$

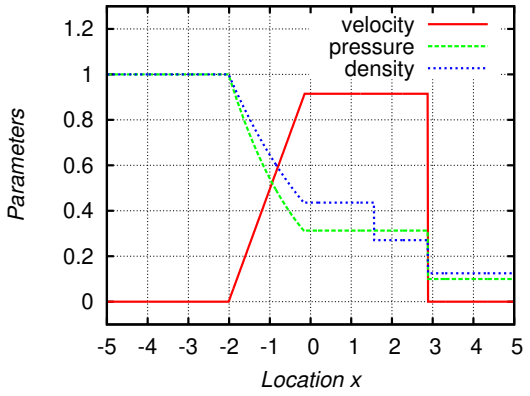


FIGURE C.2: Figure showing parameters (p, ρ, u) varying with location (x) in a shock tube at time t. Note that the figure is produced for initial conditions of $p_l = 1, \rho_l = 1, u_l = 0$ and $p_r = .1, \rho_r = .125, u_r = 0$ these are same conditions used by Sods shock tube experiment details of which are mentioned in the paper [22]

From the mathematics detailed in this section a FORTRAN code is developed in order to generate the exact solution for a shock tube (EULER_EAFZ). A flow chart explaining the this exact solution solver is given in Appendix A.7. Newton-Rapsons algorithm to solve the compatibility equation C.8 for values of M_s is defined in Appendix B.6. Some of the results produced by the exact solver are given in the Fig C.2.

C.2 Finite volume formulation for Euler equation

In FVM, computational domain is discretized by dividing it into number of non overlapping finite volumes. These finite volumes contain the control points (generally centroids) that holds information about states (**U**) and the faces of these volumes hold information of the fluxes (**F**). *Cell centered approach* has been chosen as method to chose the finite volumes, grid itself forms the finite volume and centroids of which form the control point Figure C.3 below depicts the terminology and the kind of finite volume used to solve the shock tube problem.

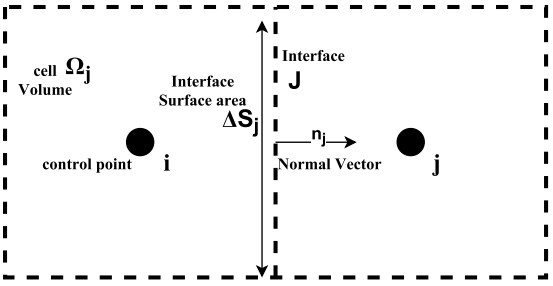


FIGURE C.3: Finite volume cell i sharing interface J with neighboring cell j

As FVM is an integral formulated scheme, integrating equation 2.6 over finite volume i

$$\int_{\Omega_i} \left(\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} \right) d\Omega = 0 \tag{C.15}$$

Using Gauss Divergence theorem to convert volume integral on flux term to surface integral,

$$\int_{\Omega_i} \frac{\partial U}{\partial t} d\Omega + \oint_{\Gamma_i} F.d\Gamma = 0, \tag{C.16}$$

introducing cell average quantity,

$$\overline{U_i} = \frac{1}{\Omega_i} \oint_{\Omega_i} U d\Omega, \tag{C.17}$$

Expressing surface integral in discretized form,

$$\frac{d\overline{U_i}}{dt} = -\frac{1}{\Omega_i} \sum_J F_{\perp J} \triangle S_J \tag{C.18}$$

Here $F_{\perp J}$ forms interracial flux through finite volume interface J. Above mentioned equation is now a ODE and can be solved using the well known Runge-Kutta method.

C.2.1 Flux discretization

There exists many schemes to evaluate the interracial fluxes $F_{\perp J}$, many of these schemes (Lax-Fridrich, Richtmyer, MacCormack, Steger-Warming, Van Leer, Godunov, Osher & Roe schemes) have been thoroughly investigated in article [38]. Each of these schemes have one or the other advantages. For this study the in order to develop the Euler solver, the Van Leer's scheme [39] is chosen in order to evaluate interracial fluxes $F_{\perp J}$. Van Leer scheme is an upwinding

scheme which work quite well in comparison to other schemes, i.e. low dispersion error and less dissipation of solution, the scheme performs quite well at the discontinuities (for details, see, [38]).

In order to calculate interfacial applying upwinding differencing at the interface J.

$$F_{\perp J} = F_{\perp L} + F_{\perp R}, \tag{C.19}$$

According to Van Leer's scheme flux vector splitting is given by the following conditions,

If $M_{\perp m} \leq -1$, then

$$F_{\perp m} = 0 \tag{C.20}$$

Else if $-1 < M_{\perp m} < 1$, then

$$F_{\perp m} = \begin{pmatrix} \dot{m_m} \\ \dot{m_m}[u + (2 - M_\perp)an_x/\gamma] \\ \dot{m_m}H \end{pmatrix}_m \tag{C.21}$$

Else if $M_{\perp m} \geq 1$, then

$$F_{\perp m} = \begin{pmatrix} \rho u_\perp \\ \rho u u_\perp + p n_x \\ (E + p)u_\perp \end{pmatrix}_m \tag{C.22}$$

Here $\dot{m}_m = \frac{\rho_m a_m (1+M_{\perp m})^2}{4}$ and m is either L or R corresponding left and right face.

At a particular time step fluxes are given by equations (C.20 – C.22), these fluxes provide the states U using equation C.18. In order to calculate the states U and primitive variables (ρ, u, p) at next time step Runger Kutta scheme is used to calculate the ODE equation C.18.

C.2.2 Runge-Kutta method for temporal discretization

Runge Kutta method is used in order to get the approximate solution of a ODE. Here in this case it is used to solve the temporal differential that arises in FVM equation C.18. RK scheme has been used for solving Euler equations and have been investigated by [40]. A 4^{th} order RK scheme is used to evaluate the term $\frac{dU}{dt}$. In general form n^{th} stage Runge Kutta scheme is as follows:

$$\begin{aligned} U^1 &= U^t \\ U^2 &= U^t + \delta t \alpha_{12} F^1 \\ U^j &= U^t + \delta t \sum_{k=1}^{j-1} \alpha_{kj} F^k \\ \\ U^{t+\delta t} &= U^t + \delta t \sum_{k=1}^{n} \beta_k F^k \end{aligned} \tag{C.23}$$

Here $F^k = f(U^k)$ Choice of β_k often applied is

$$\beta_i = 0 \quad for \quad i = 1, ..., K-1 \quad and \quad \beta_k = 1$$

For a 4^{th} order RK scheme, the coefficients are given by

$$\alpha_1 = \frac{1}{2} \quad \alpha_3 = \frac{1}{2} \quad \alpha_2 = 1$$
$$\beta_1 = \frac{1}{6} \quad \beta_2 = \beta_3 = \frac{1}{3} \quad \beta_4 = \frac{1}{6}$$

leads to

$$U^1 = U^t$$
$$U^2 = U^t + \frac{1}{2}\delta t F^1$$
$$U^3 = U^t + \frac{1}{2}\delta t F^2$$
$$U^4 = U^t + \delta t F^3$$

$$U^{t+\delta t} = U^t + \frac{\delta t}{6}(F^1 + 2F^2 + 2F^3 + F^4) \tag{C.24}$$

By knowing the value of δt we can easily predict the solution at time $t+\delta t$, as the Euler equation to be solved is transient in nature, the time step δt is a function of CFL number and is given by

$$\delta t = \frac{CFL \quad \triangle x}{max_{cells}(|u_j| + a_j)} \tag{C.25}$$

were $\triangle x$ is the cell size, u_j and a_j are the velocities and local sound speed in the cell j.

C.2.3 Artificial viscosity (slope limiting)

FVM method used to develop the Euler solver is a higher order FVM, hence it will not lead to monotone solutions. There is no reason to believe that an initial monotonic solution distribution can become non monotonic during solution evolution. In regions where the flow exhibits discontinuities or steep gradients, the reconstructed profiles produce suspicious oscillations. This can lead to loss in robustness of the solution methodology in an iterative time stepping procedure. The condition on the monotonicity of the reconstructed profile within a given finite volume requires that the reconstructed state should not exceed the maximum and minimum values specified by the cell average states in its neighboring finite volumes. Countering this problem can be achieved by introducing a slope limiter in the reconstruction process. In the present work, the limiting procedure proposed by Venkatakrishnan [23] is employed.

Consider cell i sharing its face with neighbor j. Let U_i and U_j be the cell averaged states in cell i and j respectively. Defining

$$\delta^+ = max_j(U_j - U_i); \tag{C.26}$$

$$\delta^- = min_j(U_j - U_i); \tag{C.27}$$

$$\delta = U_j - U_i \tag{C.28}$$

The condition on limiting according to venkatakrishnan is defined by
if $\delta \geq 0$ then

$$\delta^p = \delta^+ \quad \delta^n = \delta \tag{C.29}$$

if $\delta < 0$ then

$$\delta^p = \delta^- \quad \delta^n = \delta \tag{C.30}$$

With above definitions, limiting coefficient ϕ_J for the face J is given by

$$\phi_J = \frac{((\delta^p)^2 + \varepsilon)\delta^n + 2(\delta^n)^2\delta^p}{(\delta^p)^2 + (\delta^n)^2 + \delta^n\delta^p + \varepsilon} \tag{C.31}$$

Here ε is the limiting parameter with $\varepsilon \to \infty$ indicating no limiting. This is the value ϕ calculate for cell i, there exists a similar calculation procedure for value of ϕ for cell j. The final limiting coefficient is then given by

$$\phi_i = min_J(\phi_J) \tag{C.32}$$

using this the reconstructed gradients (limiting gradients) are computed as

$$(\nabla U)_{i,limited} = \phi_i(\nabla U)_i \tag{C.33}$$

C.2.4 Steps involved in FVM solution for Euler equation

Following steps summarize the steps involved to achieve FVM solution in Euler equations

1. *Reconstruction:* FVM state update formula (see equation 2.6) returns cell averaged states $\overline{U}$, resulting in loss of information of solution variation in the cell. This information is recovered using reconstruction procedure (see section C.2.3).

2. *Flux computing* Using the reconstructed gradients inviscid fluxes are computed, any suitable scheme is used to accomplish so (see section C.2.1).

3. *Solution evolution* Once fluxes are calculated solution is updated by using Runge Kutta time marching strategy (see section C.2.2).

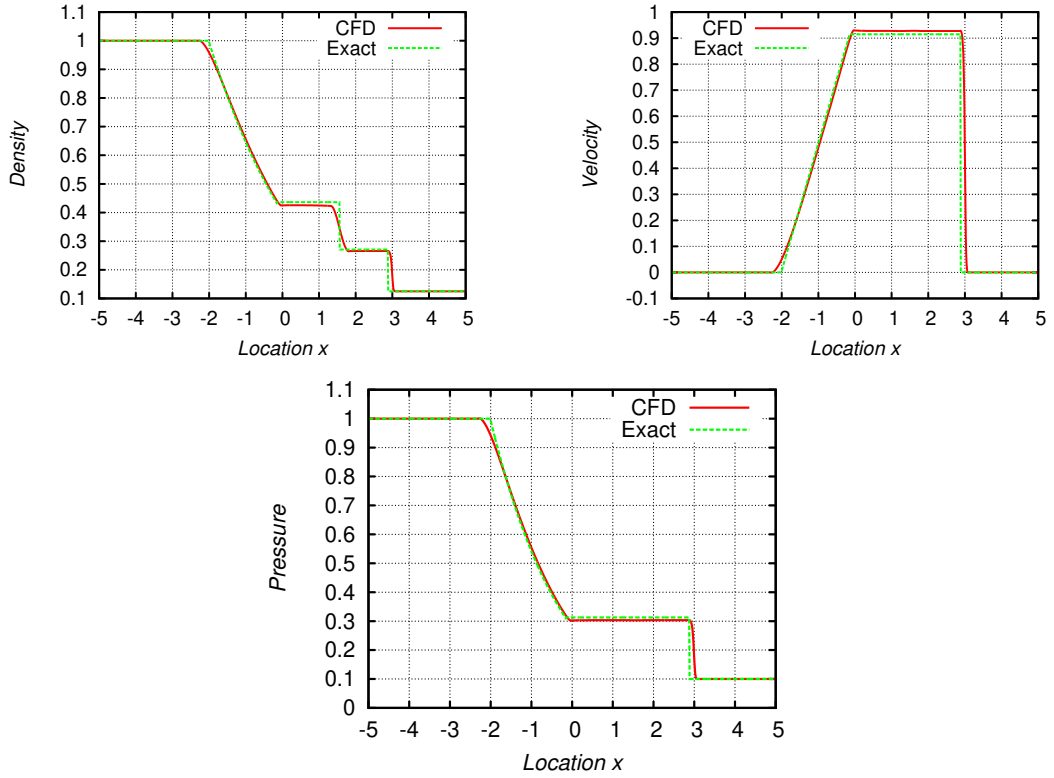


FIGURE C.4: Figure shows parameters (p, ρ, u) varying with location (x) in a shock tube at time $t > 0$ in comparison with the exact solution . Note that the figure is produced for initial conditions of $p_l = 1$,$\rho_l = 1$, $u_l = 0$ and $p_r = .1$, $\rho_r = .125$, $u_r = 0$ these are same conditions used by Sods shock tube experiment details of which are mentioned in the paper [22]

The above mentioned 3 step strategy is used to develop the Euler solver (EULER_AFZ), the algorithm of which is given in appendix B.7. The solver developed is a FVM solver that can handle unstructured grids. Figure C.4 gives some results from the solver used to solve the classic Sods Shock tube case. Note that the solver is a modular program, it needs MODGEN (FORTRAN code to generate variable containing module), USERDATA (GUI based FORTRAN code to interact with user for user input) and GRIDGEN (GUI based 1D unstructured grid generate to generate grid file) to produce the final solution for the shock tube problem. A flow chart explaining the procedure to use EULER_AFZ is shown in appendix A.2. Flow charts explaining Module generator MODEGEN, user data interface generator USERCFD , the grid generator GRIDGEN and FVM CFD code EULER_AFZ are given in Appendix A.3, A.4, A.5 and A.6 respectively.

Bibliography

[1] Charles Hirsch. Chapter 1 - the basic equations of fluid dynamics. In Charles Hirsch, editor, *Numerical Computation of Internal and External Flows (Second Edition)*, pages 27 – 64. Butterworth-Heinemann, Oxford, second edition edition, 2007. ISBN 978-0-7506-6594-0. doi: http://dx.doi.org/10.1016/B978-075066594-0/50041-2. URL http://www.sciencedirect.com/science/article/pii/B9780750665940500412.

[2] A. Dadone and B. Grossman. Progressive optimization of inverse fluid dynamic design problems. *Computers and Fluids*, 29(1):1 – 32, 2000. ISSN 0045-7930. doi: http://dx.doi.org/10.1016/S0045-7930(99)00002-X. URL http://www.sciencedirect.com/science/article/pii/S004579309900002X.

[3] Di Liu, Fu-Yun Zhao, Han-Qing Wang, Ernst Rank, and Guang-Xiao Kou. Inverse determination of building heating profiles from the knowledge of measurements within the turbulent slot-vented enclosure. *International Journal of Heat and Mass Transfer*, 55(17–18):4597 – 4612, 2012. ISSN 0017-9310. doi: http://dx.doi.org/10.1016/j.ijheatmasstransfer.2012.04.015. URL http://www.sciencedirect.com/science/article/pii/S001793101200258X.

[4] A.I. Korotkii and I.A. Tsepelev. Direct and inverse problems of high-viscosity fluid dynamics. *Automation and Remote Control*, 68(5):822–833, 2007. ISSN 0005-1179. doi: 10.1134/S0005117907050098. URL http://dx.doi.org/10.1134/S0005117907050098.

[5] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*. siam, 2005.

[6] V.Y. Arsenin A.N. Tikhonov. *Solutions of Ill-Posed Problems*. V.H. Winston and Sons, Washington, D.C., 1977.

[7] H.R. Orlande M.N. Ozisik. *Inverse Heat Transfer: Fundamentals and Applications*. Taylor and Francis Group, CRC Press, New York, NY, 2000.

[8] D. Maillet H.R. Orlande, O. Fudym and R M Cotta. *Thermal Measurements and Inverse Techniques*. Taylor and Francis Group, Boca Raton, Fl, 2011.

[9] D Knight, Q Ma, T Rossman, and Y Jaluria. Evaluation of fluid-thermal systems by dynamic data driven application systems-part ii. In *Computational Science–ICCS 2007*, pages 1189–1196. Springer, 2007.

[10] BD Henshall. *On some aspects of the use of shock tubes in aerodynamic research*. Citeseer, 1957.

[11] P. Vieille. Sur les discontinuites produites par la detente brusque de gas comprimes. *Comptes Rendus 129, 1228*, 68(5), 1899.

[12] William Payman and Wilfred Charles Furness Shepherd. Explosion waves and shock waves. vi. the disturbance produced by bursting diaphragms with compressed air. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 186(1006): 293–321, 1946.

[13] Wayland Griffith. Shock-tube studies of transonic flow over wedge profiles. *Journal of the Aeronautical Sciences (Institute of the Aeronautical Sciences)*, 19(4), 2012.

[14] J Lukasiewicz. *Shock tube theory and applications*. National Aeronautical Establishment, 1952.

[15] Abraham Hertzberg. A shock tube method of generating hypersonic flows. *Journal of the Aeronautical Sciences (Institute of the Aeronautical Sciences)*, 18(12), 2012.

[16] A Hertzberg. The application of the shock tube to the study of the problems of hypersonic flight. *Journal of Jet Propulsion*, 26(7):549–554, 1956.

[17] RK Hanson and DF Davidson. Recent advances in laser absorption and shock tube methods for studies of combustion chemistry. *Progress in Energy and Combustion Science*, 44:103–114, 2014.

[18] Stephen Downes, Andy Knott, and Ian Robinson. Towards a shock tube method for the dynamic calibration of pressure sensors. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 372(2023):20130299, 2014.

[19] II Glass. A theoretical and experimental study of shock-tube flows. *Journal of the Aeronautical Sciences (Institute of the Aeronautical Sciences)*, 22(2), 2012.

[20] Abraham Hertzberg. A shock tube method of generating hypersonic flows. *Journal of the Aeronautical Sciences (Institute of the Aeronautical Sciences)*, 18(12), 2012.

[21] Ionut Danaila, Pascal Joly, Sidi Mahmoud Kaber, and Marie Postel. *An introduction to scientific computing: Twelve computational projects solved with MATLAB*. Springer Science & Business Media, 2007.

[22] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of computational physics*, 27(1):1–31, 1978.

[23] Venkat Venkatakrishnan. Convergence to steady state solutions of the euler equations on unstructured grids with limiters. *Journal of computational physics*, 118(1):120–130, 1995.

[24] Andre Nikolaevich Tikhonov and Vasili Yakovlevich Arsenin. *Solutions of ill-posed problems*. Vh Winston, 1977.

[25] Jacques Hadamard. *Lectures on Cauchy's problem in linear partial differential equations*. Courier Corporation, 2014.

[26] James V Beck, Ben Blackwell, and Charles R St Clair Jr. *Inverse heat conduction: Ill-posed problems*. James Beck, 1985.

[27] Oleg M Alifanov. *Inverse heat transfer problems*. Springer Science & Business Media, 2012.

[28] P. Le Masson Y. Favennec. Lecture 9: Large scale optimization for function estimation. In *Thermal mesurements and inverse techniques,Advanced Spring School, Biarrtiz*. METTI, 2011.

[29] A. Holder, editor. *Mathematical Programming Glossary*. INFORMS Computing Society, http://glossary.computing.society.informs.org, 2006–14. Originally authored by Harvey J. Greenberg, 1999-2006.

[30] Xin-She Yang and Suash Deb. Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1(4):330–343, 2010.

[31] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[32] Philip E Gill, Walter Murray, and Margaret H Wright. *Practical optimization*. Academic press, 1981.

[33] Godfrey C Onwubolu and BV Babu. *New optimization techniques in engineering*, volume 141. Springer, 2013.

[34] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[35] James Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[36] Maurice Clerc. *Particle swarm optimization*, volume 93. John Wiley & Sons, 2010.

[37] Magnus Rudolph Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. 1952.

[38] H. Nishikawa. A comparison of numerical flux formulas for the euler equations. *Math 671 final assignment*, pages 120–130, 1998.

[39] Bram Van Leer. Flux-vector splitting for the euler equations. In *Eighth international conference on numerical methods in fluid dynamics*, pages 507–512. Springer, 1982.

[40] Antony Jameson, Wolfgang Schmidt, Eli Turkel, et al. Numerical solutions of the euler equations by finite volume methods using runge-kutta time-stepping schemes. *AIAA paper*, 1259:1981, 1981.