Poznan University of Technology

Doctoral Thesis

Experimental Research of Dual-Throat Fluidic Thrust Vectoring Nozzle

Author:
MSc. Eng. Wiktor Hoffmann

Supervisor:
DSc. Eng. Magda Joachimiak

Auxiliary supervisor:
PhD. Eng. Bartosz Ziegler



A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering

in the:

Faculty of Environmental Engineering and Energy
Institute of Thermal Energy

June 2026

Declaration of Authorship

I, MSc. Eng. Wiktor HOFFMANN, declare that this thesis titled, 'Experimental Research of Dual-Throat Fluidic Thrust Vectoring Nozzle' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Streszczenie

W rozprawie doktorskiej przedstawiono badania eksperymentalne dwuprzewężeniowej dyszy typu Dual-Throat Nozzle (DTN), przeznaczonej do sterowania wektorem ciągu silnika odrzutowego bez poruszania elementami dyszy. W badanej koncepcji odchylenie głównego strumienia uzyskuje się przez doprowadzenie strumienia wtórnego do pierwszego przekroju krytycznego odpowiednio ukształtowanej dyszy. Rozwiązanie to jest istotne z punktu widzenia rozwoju układów napędowych o mniejszej złożoności mechanicznej, krótkim czasie odpowiedzi i zwiększonej niezawodności.

Celem pracy było doświadczalne określenie odpowiedzi dynamicznej osiowosymetrycznej dyszy DTN oraz zmierzenie częstotliwości i amplitudy ciśnienia na ściance dyszy DTN, podczas pracy ze stałymi parametrami czynnika w przekrojach wlotowych. Przegląd literatury wykazał, że dotychczasowe badania dysz DTN dotyczyły głównie optymalizacji geometrii, skuteczności wektorowania oraz pola przepływu w warunkach stacjonarnych, a jedyne badania niestacjonarne były prowadzone z wykorzystaniem analiz numerycznych. Brakuje natomiast danych eksperymentalnych opisujących odpowiedź przejściową tego typu dyszy. Praca uzupełnia tę lukę przez wyznaczenie charakterystycznych parametrów dynamicznych, w tym czasu opóźnienia t_0 oraz stałej czasowej τ, dla reprezentatywnego przypadku pracy układu. Nowość pracy polega na eksperymentalnym wyznaczeniu czasu odpowiedzi dynamicznej przy niestacjonarnym charakterze przepływu wewnątrz dyszy.

Badania przeprowadzono w krótkotrwałym przepływie uzyskanym w tunelu typu tuba Ludwiega. Zarejestrowano przebiegi ciśnienia w wybranych punktach pomiarowych na ściankach dyszy oraz parametry w przekrojach wlotowych. Ze względu na obecność pulsacji ciśnienia opracowano procedurę przetwarzania danych, pozwalającą oddzielić średnią odpowiedź przejściową od składowych niestacjonarnych. Do opisu dynamiki zastosowano model inercyjny pierwszego rzędu z opóźnieniem, umożliwiający powtarzalne wyznaczenie parametrów t_0 i τ.

Wyniki wykazały, że wtrysk boczny powoduje mierzalną zmianę rozkładu ciśnienia na ściankach dyszy, potwierdzając możliwość sterowania kierunkiem przepływu. Jednocześnie zarejestrowane sygnały zawierały wyraźne składowe pulsacyjne, obecne zarówno przed, jak i po uruchomieniu wtrysku wtórnego. Oznacza to, że opis pracy dyszy DTN wyłącznie za pomocą wartości średnich jest niewystarczający, szczególnie przy ocenie jej przydatności w układach sterowania wektorem ciągu.

Porównanie wyników eksperymentalnych z analizą numeryczną pozwoliło ocenić zgodność przewidywanego i mierzonego rozkładu ciśnienia oraz wskazać ograniczenia modelowania przepływu w złożonej geometrii DTN. Uzyskane wyniki potwierdziły przydatność zaproponowanej metodyki do analizy dynamicznej dysz oraz mogą stanowić podstawę dalszych prac nad modelowaniem, optymalizacją i sterowaniem układami DTN.

Abstract

This doctoral dissertation presents an experimental study of a Dual-Throat Nozzle (DTN) designed for thrust vector control in a jet propulsion system without moving nozzle components. In the investigated concept, deflection of the primary jet is achieved by supplying a secondary flow to the first critical section of a properly shaped nozzle. This solution is relevant to the development of propulsion systems with reduced mechanical complexity, short response time, and increased reliability.

The aim of the work was to experimentally determine the dynamic response of an axisymmetric DTN and to measure the frequency and amplitude of wall pressure fluctuations during operation with constant flow parameters at the inlet sections. The literature review showed that previous studies of DTN nozzles focused mainly on geometry optimization, vectoring performance, and the flow field under steady-state conditions, while the only unsteady studies were based on numerical analyses. Experimental data describing the transient response of this type of nozzle are therefore lacking. This work addresses this gap by determining the characteristic dynamic parameters, including the delay time t_0 and the time constant τ, for a representative operating case. The novelty of the work lies in the experimental determination of the dynamic response time under the non-stationary character of the internal nozzle flow.

The experiments were carried out under short-duration flow conditions obtained in a Ludwieg tube facility. Pressure time histories were recorded at selected measurement points on the nozzle walls, together with parameters at the inlet sections. Due to the presence of pressure pulsations, a data processing procedure was developed to separate the mean transient response from non-stationary components. The dynamics were described using a first-order inertial model with time delay, enabling repeatable determination of the parameters t_0 and τ.

The results showed that secondary injection causes a measurable change in the wall pressure distribution, confirming the possibility of controlling the flow direction. At the same time, the recorded signals contained distinct pulsating components, present both before and after the activation of secondary injection. This indicates that describing DTN operation solely by mean values is insufficient, especially when assessing its applicability in thrust vector control systems.

Comparison of the experimental results with the numerical analysis made it possible to assess the agreement between predicted and measured pressure distributions and to identify limitations of flow modeling in the complex DTN geometry. The obtained results confirmed the usefulness of the proposed methodology for dynamic nozzle analysis and may serve as a basis for further work on the modeling, optimization, and control of DTN systems.

Acknowledgements

In this place I would like to thank the Institute of Thermal Energy of Poznan University of Technology, with special recognition to MSc. Eng. Tomasz Krakowski for thorough support during this project.

Contents

Declaration of Authorship iii

Streszczenie v

Abstract vii

Acknowledgements ix

Table of contents xi

List of figures xiii

List of tables xvii

Nomenclature xix

1 Introduction to the thesis topic 1
- 1.1 Historical background and past efforts . . . 1
- 1.2 State of the Art . . . 4
- 1.3 Motivation . . . 14
- 1.4 Aim and objectives of the thesis . . . 15

2 Background information and theory 17
- 2.1 Thrust vector control - definition . . . 17
- 2.2 Thrust vectoring quality evaluation . . . 18
- 2.3 Flow in DTN . . . 21
- 2.4 Current dynamic experimental studies on DTN . . . 25
- 2.5 Implications of the Literature Review . . . 28

3 Geometry and CFD Analysis 31
- 3.1 Nozzle geometry . . . 31
- 3.2 Boundary conditions . . . 32
- 3.3 Model and meshing . . . 34
- 3.4 Physical and numerical setup . . . 36
- 3.5 Results . . . 37

4 Experimental setup & instrumentation 45
- 4.1 Operating Principle and Design of the Transonic Ludwieg tube facility . . . 45

4.2 Manufacturing and Modular Design of the DTN Model 49
4.3 Mass Flow Control in the Primary and Secondary Lines 51
4.4 Measurement System and Instrumentation 52
4.5 Experimental Test Procedure 63

5 Data analysis and the experimental results 67
5.1 Preliminary overview of the results 67
5.2 Data analysis methods 68

6 Experimental results 77
6.1 Scope of the experimental results 77
6.2 Flow conditions 77
6.3 Time analysis results 78
6.4 Wall pressure distribution & comparison with CFD results 87
6.5 Non-stationary flow 88

7 Conclusion & future directions 93

A DTN test section geometry 97

B Force Decoder 103

C CFD Pressure & FFT code 107

D LabVIEW structures 115

E TDMS Reader 119

F Voltage to pressure conversion rate calculator 121

G Raw-to-Precut Processing code 123

H Data post-processor 127

I FFT Analysis 147

J Raw and optimized values of wall pressure and secondary mass flow 153

K Wall Plots 163

Bibliography 171

List of Figures

1.1 Schematic of a flow field on SVC [85] 4
1.2 Sketch of typical SVC [53] 5
1.3 Sprint missile FTV methods: a) hot gas injection [7], b) freon injection [63] 5
1.4 Schematic of Counter Flow Thrust Vectoring [96] 6
1.5 Schematic Co-flow Thrust Vectoring [2] 6
1.6 Schematic Throat Skewing Vectoring Control [62] 7
1.7 Schematic of DTN thrust vector control [16] 8
1.8 Comparison of DTN's experimental thrust vectoring efficiency and thrust efficiency to TSC and SVC methods [32] 8
1.9 DTN characteristic dimensions 10

2.1 Computational total pressure flow contours show the decreased effective area (A_{vc}) from the vena contracta effect at the upstream minimum area (A_{min}) [31]. 19
2.2 Nozzle internal performance for an 13-60 geometry, operating over a range of NPR with 3% fluidic injection [32]. Open symbols experimental data. Solid symbols computational data. 20
2.3 Variable area nozzle operating curve [32] 22
2.4 Schematic of the Walter HWK 109-509A-1 rocket motor in Messerschmitt Me 163 Komet [4] 23
2.5 Sketch of theoretical flow through DTN [15] 24
2.6 Pressure fluctuations in axisymmetric flow[45] 26
2.7 Pressure amplitude periodogram [45] 27

3.1 Secondary flow injection span 32
3.2 Flow domain and far field dimensions 35
3.3 Mesh 35
3.4 Mesh – close view of the nozzle region 36
3.5 DTN thrust vector angle δ_p over range of NPR with MFR=3%. Open symbols experimental data. Solid symbols computational data. [32] 39
3.6 Calculated mean surface pressure distribution. 40
3.7 FFT amplitudes of the wall-pressure signals obtained for the vectorized CFD case. 41
3.8 Instantaneous Mach number fields before the fully developed vectoring state. 42
3.9 Instantaneous Mach number fields after the onset of the vectorized flow state. 43

4.1 Scheme of the expansion process in the Ludwieg tube [79]. 46

4.2 Ludwieg tube at the Aviation Engines Laboratory of the Poznan University of Technology. . . . 47
4.3 Ruptured BoPET diaphragm after a test run. . . . 48
4.4 Blade used to initiate diaphragm rupture. . . . 48
4.5 DTN test section with Ludwieg tube Facility 49
4.6 Overview of the DTN modular test section 50
4.7 Orifice plate cross section 51
4.8 Discharge coefficients for different shapes of orifice plates [22]. . . . 52
4.9 Comparison of signals recorded by PCB piezoelectric and ENDEVCO piezoresistive transducers installed face to face during the same run [76]. . 53
4.10 Measurement points 56
4.11 Shock tube setup used to evaluate the dynamic response of pressure transducer mounting configurations. . . . 57
4.12 Influence of connecting tube length and diameter on the measured pressure build up time (100 samples correspond to 0.002 s). . . . 57
4.13 Custom five channel instrumentation amplifier used for the Kulite XTL-190 pressure transducers. . . . 61
4.14 Calibration of the pressure measurement chain using a reference pressure transducer. . . . 61
4.15 LabVIEW FPGA control sequence 64
4.16 Example measured values 65

5.1 Pressures p_{12} and p_{22} and secondary mass flow $\dot{m}_s$. Vertical line indicates start of secondary flow. . . . 68
5.2 Parameters of the first-order system with transport delay 73

6.1 Raw, smoothed and fitted values of wall pressure at point p_{11} and secondary mass flow $\dot{m}_s$ 82
6.2 Raw, smoothed and fitted values of wall pressure at point p_{12} and secondary mass flow $\dot{m}_s$ 82
6.3 Raw, smoothed and fitted values of wall pressure at point p_{13} and secondary mass flow $\dot{m}_s$ 83
6.4 Raw, smoothed and fitted values of wall pressure at point p_{14} and secondary mass flow $\dot{m}_s$ 83
6.5 Raw, smoothed and fitted values of wall pressure at point p_{15} and secondary mass flow $\dot{m}_s$ 84
6.6 Raw, smoothed and fitted values of wall pressure at point p_{22} and secondary mass flow $\dot{m}_s$ 84
6.7 Raw, smoothed and fitted values of wall pressure at point p_{24} and secondary mass flow $\dot{m}_s$ 85
6.8 Raw, smoothed and fitted values of wall pressure at point p_{25} and secondary mass flow $\dot{m}_s$ 85
6.9 Wall pressure distribution before and after secondary injection, normalized by the primary stagnation pressure p_3. Error bars denote the standard deviation of p_i/p_3 in the corresponding time window. . . . 88
6.10 Calculated variation of mass flow rate [94] 90
6.11 Frequency domain analysis of the wall pressure signals. . . . 91

A.1 DTN test section assembly 97

A.2 DTN test section base and nozzle divergent part 98
A.3 Primary throat and secondary injection slot insert part 99
A.4 DTN convergent and secondary (exit) throat part 100
A.5 DTN - Ludwieg tube adapter 101

D.1 LabVIEW host user interface 115
D.2 LabVIEW host structure VI 116
D.3 LabVIEW FPGA structure VI 117

J.1 Raw and optimized values of secondary mass flow w_s over time and dimensionless time 154
J.2 Raw and optimized values of wall pressure at point p_{11} over time and dimensionless time 155
J.3 Raw and optimized values of wall pressure at point p_{12} over time and dimensionless time 156
J.4 Raw and optimized values of wall pressure at point p_{13} over time and dimensionless time 157
J.5 Raw and optimized values of wall pressure at point p_{14} over time and dimensionless time 158
J.6 Raw and optimized values of wall pressure at point p_{15} over time and dimensionless time 159
J.7 Raw and optimized values of wall pressure at point p_{22} over time and dimensionless time 160
J.8 Raw and optimized values of wall pressure at point p_{24} over time and dimensionless time 161
J.9 Raw and optimized values of wall pressure at point p_{25} over time and dimensionless time 162

List of Tables

1.1 Selected aircrafts, missiles and rockets first flight/launch, thrust vectoring methods and engine type (LRE - liquid rocket engine, SRE - solid rocket engine & AB - air breathing engine) 4

3.1 CFD boundary conditions for the DTN simulation. 33
3.2 Air properties used in the Fluent model. 36
3.3 SST k-ω turbulence model settings. 37

4.1 Transducers parameters. 53
4.2 Measurement chain parameters. 55
4.3 Comparison of theoretical and experimentally determined voltage to pressure dividers for the pressure measurement channels. 62

6.1 First-order delay fit parameters. 79
6.2 Comparison of the wall pressure response parameters relative to the secondary mass flow response (from the shortest). 81
6.3 Comparison of the present experimental response time with the numerical results of Ferlauto and Marsilio [28, 29]. 87
6.4 Dominant FFT peak frequency and amplitude before and after secondary injection. 90

Nomenclature

A_t	nozzle throat area	m^2 or mm^2
A_e	nozzle exit area	m^2 or mm^2
$A_{\min}$	geometrical minimum cross-sectional area	m^2 or mm^2
A_{vc}	effective cross-sectional area	m^2 or mm^2
C_d	discharge coefficient	$-$
$C_{d,p}$	discharge coefficient of the primary nozzle	$-$
$C_{f,g,\mathrm{sys}}$	system resultant thrust ratio	$-$
c	speed of sound	m/s
c_k	Savitzky–Golay filter coefficient	$-$
d	pressure tap channel diameter	m or mm
d_t	nozzle throat diameter	m or mm
d_e	nozzle exit diameter	m or mm
dA	differential area element	m^2
f	frequency	Hz
f_n	resonance frequency of the n-th acoustic mode	Hz
f_1, f_3, f_5, f_7	selected quarter-wave resonance frequencies	Hz
F_A	measured axial force	N
F_N	measured normal force	N
F_S	measured side force	N
F_r	resultant thrust	N
$F_{i,p}$	ideal isentropic thrust of the primary nozzle	N
$F_{i,s}$	ideal isentropic thrust of the secondary nozzle	N
F_x	axial force component	N
F_y	lateral force component in the symmetry plane	N
F_z	out-of-plane force component	N
h_{ut}	height of the primary-nozzle upstream throat	mm
h_{dt}	height of the primary-nozzle downstream throat	mm
i, k, n	integer indices	$-$
K	response amplitude, $K = y_\infty - y_0$	depends on signal
l	length of the primary-nozzle cavity	mm
L	physical pressure tap channel length	m or mm
L_{eff}	effective acoustic length of the pressure tap channel	m or mm
L_{ref}	reference length used for non-dimensional time	m
M	Mach number	$-$
m	Savitzky-Golay half-window size	$-$
$\dot{m}_{i,p}$	ideal mass flow rate of the primary nozzle	kg/s
$\dot{m}_{i,s}$	ideal mass flow rate of the secondary nozzle	kg/s
$\dot{m}$	actual mass flow rate	kg/s
$\dot{m}_p$	actual mass flow rate of the primary jet	kg/s

$\dot{m}_s$	actual mass flow rate of the secondary jet	kg/s
N	number of samples	$-$
$\mathbf{n}$	unit normal vector	$-$
p	local static pressure	bar or Pa
p_0	ambient or nozzle exit static pressure	bar or Pa
p_a	atmospheric pressure	bar or Pa
p_t	total pressure upstream of the nozzle throat	bar or Pa
$p_{t,j}$	average total pressure of the primary jet	bar or Pa
$p_{t,s}$	total pressure of the secondary jet	bar or Pa
p_i	pressure measured at the i-th wall-pressure point	bar or Pa
P_k	Fourier coefficient	depends on signal
R	specific gas constant	$J/(kg\,K)$
R^2	coefficient of determination	$-$
r_i	residual of the fitted model	same as y
S	Strouhal number	same as $-$
t	time	s
t_0	delay time of the fitted response	s
t_{63}	characteristic response time, $t_{63} = t_0 + \tau$	s
t^*	non-dimensional time	$-$
T	absolute temperature	K
T_0	reference or initial temperature	K
T_1, T_2	measured temperatures in the Ludwieg tube system	K
$T_{0,2}$	secondary-flow stagnation temperature	K
$T_{t,j}$	average total temperature of the primary jet	K
$T_{t,s}$	total temperature of the secondary jet	K
u	characteristic velocity	m/s
V	voltage	V
$\mathbf{V}$	velocity vector	m/s
W	nozzle width	mm
x, y, z	Cartesian coordinates	$-$
$y(t)$	analyzed time-dependent signal	depends on signal
y_i	measured discrete value of the analyzed signal	depends on signal
$\hat{y}_i$	fitted value of the analyzed signal	depends on signal
y_0	initial quasi-steady value of the analyzed signal	depends on signal
y_∞	final quasi-steady value of the analyzed signal	depends on signal
y^+	non-dimensional wall distance	$-$
ΔP	pressure difference	bar or Pa
Δt_0	delay difference	s
δ_p	resultant pitch thrust-vector angle	$^\circ$
δ_{xy}	thrust-vector angle in the symmetry plane	$^\circ$
η	resultant pitch thrust-vectoring efficiency	$^\circ/\%$
κ	specific heat ratio	$-$
μ	dynamic viscosity	$Pa\,s$
μ_t	eddy viscosity	$Pa\,s$
ρ	density	kg/m^3
τ	time constant of the fitted response	s
τ^*	non-dimensional time constant	$-$
Θ_1	upstream divergent cavity ramp angle	$^\circ$
Θ_2	downstream convergent cavity ramp angle	$^\circ$
Θ	secondary-flow injection angle	$^\circ$

φ	secondary-flow injection span angle	$^\circ$

ARX	**A**uto**R**egressive with e**X**ogenous input
BDTN	**B**ypass **D**ual-**T**hroat **N**ozzle
BoPET	**B**iaxially **o**riented **P**oly**e**thylene **T**erephthalate
CAD	**C**omputer-**A**ided **D**esign
CFD	**C**omputational **F**luid **D**ynamics
DTN	**D**ual-**T**hroat **N**ozzle
FFT	**F**ast **F**ourier **T**ransform
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray
FTV	**F**luidic **T**hrust **V**ectoring
MFR	**M**ass **F**low **R**atio
NPR	**N**ozzle **P**ressure **R**atio
PC	**P**olynomial **C**haos
RANS	**R**eynolds-**A**veraged **N**avier-**S**tokes
SLS	**S**pace **L**aunch **S**ystem
SRB	**S**olid **R**ocket **B**ooster
SST	**S**hear **S**tress **T**ransport
SVC	**S**hock **V**ector **C**ontrol
TDMS	**T**echnical **D**ata **M**anagement **S**treaming
TVC	**T**hrust **V**ector **C**ontrol
URANS	**U**nsteady **R**eynolds-**A**veraged **N**avier–**S**tokes
VI	**V**irtual **I**nstrument
TSC	**T**hroat **S**kewing **C**ontrol
VTOL	**V**ertical **T**ake-**O**ff and **L**anding

To my children — may you never lose your curiosity about the world, the courage to ask questions, and the perseverance to seek answers.

Chapter 1

Introduction to the thesis topic

1.1 Historical background and past efforts

Thrust vectoring is a technique that allows for the manipulation of the direction of an aircraft's thrust, and therefore flight direction. This section aims to provide a detailed account of the key milestones and developments in thrust vectoring from its early conceptualization to its incorporation into modern aircraft and aerospace systems. By examining the historical evolution of thrust vectoring,it is possible to gain some insights into the encountered challenges, the corresponding technological and methodological breakthroughs, and the subsequent impact on the fields of aviation and aerospace engineering.

Thrust vectoring was the first to be widely described by Robert H. Goddard as a solution to the problems of maintaining flight trajectories regardless of lateral wind gusts and controlling rockets in the high layers of the atmosphere and beyond. The first paper [35] presents the concept of using gas-dynamic interceptors, also known as jet tabs, mechanically coupled to a gyroscopic system. Further work led to the design and flight testing of a rocket with a gimbaled combustion chamber and nozzle assembly in 1936 [36].

The onset of World War II provided impetus for further experimentation with thrust vectoring. Hermann Oberth and Wernher von Braun conducted significant research with the A-4 (Aggregat 4 and V-2 rocket) [48], equipped with exhaust vanes connected with external rudders in the tail section. These early experiments, though limited in scope, demonstrated the potential of thrust vectoring to improve maneuverability in aircraft and rockets. It is worth mentioning here that the V-2 rocket, in a modified configuration, was the first to cross The Kármán line. [86].

The post-war era witnessed notable advancements in aviation and aerospace technologies, including thrust vectoring. The Bell X-14 (first flight on 19 February 1957)

[10], which became the first jet powered aircraft to perform vertical take-off and landing (VTOL), incorporated thrust vectoring gimbaled nozzles. Note that the vectored thrust provided not only lifting power, but also maneuverability and stability. The first aircraft introduced into service with thrust vectoring was the Harrier family of fighters, which used pivoting nozzles to shorten run up or outright vertical takeoff [24]. Today, thrust vectoring in fighter aircraft is implemented in the form of pivoting nozzles that can increase maneuverability or, in cooperation with additional equipment, enable vertical takeoff [11] [37] [90].

Space exploration has contributed to, and even required, the development of thrust vectoring technique. As space agencies developed rockets and spacecraft, they encountered the need for more precise control during critical phases of flight. Successful launches, orbital maneuvers, and re-entry procedures are difficult to accomplish without thrust vectoring and require additional engines such as Vernier thrusters. During the launch, thrust vectoring is vital for maintaining the rocket's trajectory and stability. It allows the vehicle to follow a precise ascent path and make adjustments in response to dynamic conditions such as wind or engine performance variances. In orbit, spacecraft use thrust vectoring for various maneuvers, including orbit insertion, orbit transfers, and docking with other spacecraft or space stations. The precision offered by thrust vectoring is essential for the safety and success of these operations. For missions beyond Earth's orbit, thrust vectoring enables trajectory corrections and fine tuning of the spacecraft's path. This is crucial for missions to other planets or celestial bodies, where small errors can lead to mission failure. The Solid-Rocket-Boosters (SRB) are equipped with a pair of hydraulic gimbal servo actuators that enable the nozzle to adjust vertically and horizontally. This capability facilitates thrust vectoring, contributing to the whole vehicle's control across all three axes: roll, pitch, and yaw [21]. It should be noted here that the use of a gimbaled nozzle on solid fuel engines is not a common practice, due to technological problems arising from high gas temperature and pressure. Liquid fueled rocket and space shuttle engines are made almost always as complete assemblies of nozzles, combustion chambers, and pumps, suspended from actuators that fix the angle of the thrust vector [64] [55] [84]. The fact that the Space Launch System (SLS), launched in 2022, uses the same solutions as the space shuttles is a testament to their effectiveness [65].

Most missiles, regardless of the fuel and type used (ballistic, air-to-air, surface-to-air, etc.), use exhaust jet vanes as a method of thrust vectoring. The disadvantage of this solution is the reduction in propulsion efficiency, but the simplicity of the design required in these applications and the resulting ease of implementation are the dominant factor [59] [92] [20] [19]. However, in some cases where operating conditions have determined the complete elimination of control surfaces, jet vanes have also been abandoned in favor

of thrust vectoring through secondary fluid injection into the nozzle. Details of such solutions will be presented in the following chapters. The concept of fluid injection-induced thrust vectoring was first considered between the mid 50 and mid 60 as a solution for rocket nozzle systems [1] [91]. Fluidic Thrust Vectoring (FTV) nozzles are an propulsion system that is used to control the thrust direction by injecting a second stream of fluid. Because there are no mechanical actuators or moving parts involved, the FTV nozzle is inherently lighter and less complex than traditional mechanical thrust vectoring systems. This can lead to weight savings, which is important for aircraft and rockets that require higher performance. With fewer moving parts (valve components), the FTV nozzle is less prone to mechanical failure, improving the overall reliability of the propulsion system. FTV offers rapid response times, allowing for quick and precise adjustments in the direction of thrust during flight maneuvers. The absence of mechanical actuators reduces the aircraft's radar cross section and heat signature, contributing to improved stealth characteristics. FTV provides enhanced maneuverability, allowing aircraft and rockets to perform complex aerial maneuvers efficiently.

The historical background of thrust vectoring demonstrates the step-by-step evolution of this technique from its conceptualization to its applications in modern aviation and aerospace. Key developments in jet nozzle technique, advancements during the space race, and military applications have all contributed to thrust vectoring becoming an important element for enhancing the performance and control of aircraft and rockets. The knowledge gained from understanding the historical journey of thrust vectoring provides the foundation for further exploration. Table 1.1 shows the selected aircrafts, rockets, and missiles that use thrust vectoring. It can be noted that, in the context of mechanical methods, no substantial technological breakthroughs have been achieved over the years. In the case of missiles, jet vanes are the dominant method, in carrier rockets, gimbaled engines, and in jet aircraft, movable nozzles.

TABLE 1.1: Selected aircrafts, missiles and rockets first flight/launch, thrust vectoring methods and engine type (LRE - liquid rocket engine, SRE - solid rocket engine & AB - air breathing engine)

date	**aircraft**	**type**	**engine type**	**vectoring method**
1937	Goddard's L-7 [36]	experimental rocket	LRE	gimbaled engine
1942	V-2 [48]	ballistic missile	LRE	exhaust jet vanes
1953	PGM-11 Redstone [59]	ballistic missile	LRE	exhaust jet vanes
1957	SS-1b Scud-A [92]	ballistic missile	LRE	exhaust jet vanes
1957	Bell X-14 [10]	experimental aircraft	AB	movable nozzle
1964	LGM-30F [70]	ballistic missile	SRE	secondary fluid injection
1965	Sprint [63]	surface-to-air missile	SRE	secondary fluid injection
1967	Harrier jump jet [24]	experimental aircraft	AB	movable nozzle
1967	Saturn V [64]	lift launch vehicle	LRE	gimbaled engine
1977	UGM-96 Trident I [74]	ballistic missile	SRE	gimbaled nozzle
1981	Space Shuttle SRBs [21]	solid rocket booster	SRE	gimbaled nozzle
1981	Space Shuttle [55]	lift launch vehicle	LRE	gimbaled engine
1987	UGM-133A Trident II [26]	ballistic missile	SRE	gimbaled nozzle
1988	McDonnell Douglas F-15 STOL/MTD [11]	experimental aircraft	AB	movable nozzle
1990	Lockheed Martin F-22 Raptor [37]	fighter	AB	movable nozzle
1993	PSLV [54]	lift launch vehicle	SRE	secondary fluid injection
1996	9K720 Iskander [34]	ballistic missile	SRE	jet vanes
1996	Sukhoi Su-37 [38]	fighter	AB	movable nozzle
1997	McDonnell Douglas X-36 [27]	experimental aircraft	AB	movable nozzle
2000	Sukhoi Su-30MKI [39]	fighter	AB	movable nozzle
2006	Lockheed Martin F-35 Lightning II [90]	VTOL fighter	AB	movable nozzle
2008	AIM-9X Sidewinder [20]	air-to-air missile	SRE	jet vanes
2015	MPATGM [18]	anti-tank guided missile	SRE	jet vanes
2017	QRSAM [19]	surface-to-air missile	SRE	jet vanes
2018	SpaceX Falcon Heavy [84]	lift launch vehicle	LRE	gimbaled engine
2022	Space Launch System (SLS) [65]	lift launch vehicle	LRE	gimbaled engine

1.2 State of the Art

1.2.1 Various solutions of FTV nozzles

Shock Vector Control (SVC) is a technique that involves injecting a secondary fluid flow into the divergent section of a convergent-divergent nozzle. This injected flow creates a disturbance that leads to the formation of an oblique shock wave. This shock wave is generated because of the low-pressure area that exists downstream of the injection port. When the oblique shock wave interacts with the primary flow, it causes the flow to be deflected in a supersonic manner, resulting in thrust vectoring [5]. Details of the flow field are shown in Figure 1.1. An illustrative diagram of the SVC can be seen in Figure 1.2.

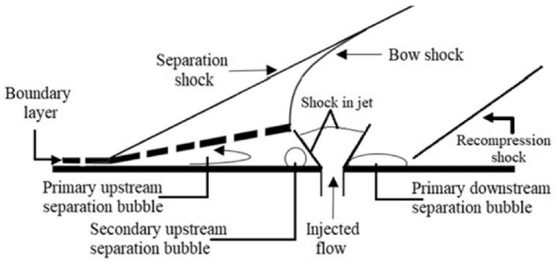


FIGURE 1.1: Schematic of a flow field on SVC [85]

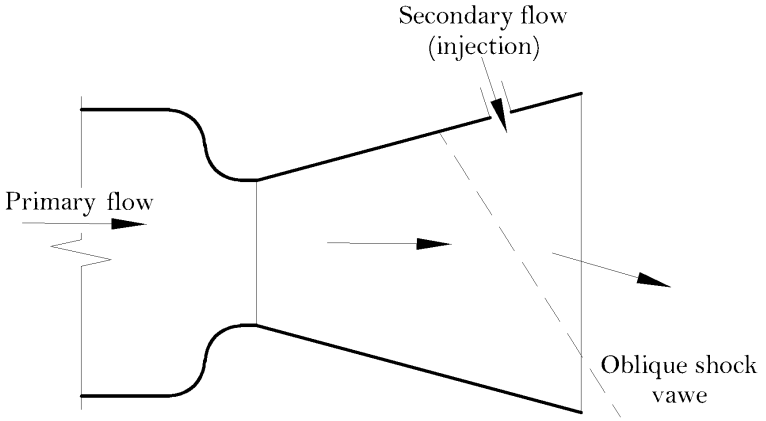


FIGURE 1.2: Sketch of typical SVC [53]

SVC got several adaptations in solid-fuel missiles, where no jet vanes or gimbaled engine were possible. An example of the application was Sprint missiles, which used SVC by injecting hot gas into the divergent part of the nozzle [63]. The effectiveness of this solution is evidenced by the maneuverability of this missile at speeds up to 3,400 meters per second (Mach 10). It proved to be a problem to design a valve that would operate at more than 200 bar of pressure and temperatures of more than 3,500 K, so switching valves were used that directed the gas into the nozzle or out of the missile Figure 1.3 a). Creating a secondary injection flow through bypass from the combustion chamber was also proposed, in the 1964 by Sekhon [82]. Later work led to replacement of the bypass from the combustion chamber with freon gas tanks, which was also injected into the divergent part of the nozzle 1.3 b).

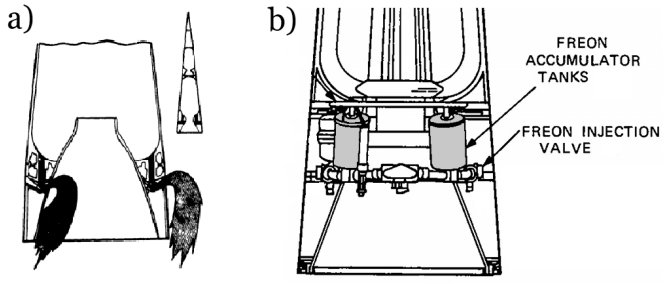


FIGURE 1.3: Sprint missile FTV methods: a) hot gas injection [7], b) freon injection [63]

The Counterflow Thrust Vectoring technique employs a secondary fluid flow that moves counter to the primary flow direction. Typically, this technique uses suction to generate a secondary fluid flow, resulting in asymmetry in the primary flow. Counterflow Thrust Vectoring is capable of attaining significant thrust vectoring angles using suction points located between the nozzle's rear edge and an aft collar [72]. Figure 1.4 illustrates the Counterflow Thrust Vectoring setup.

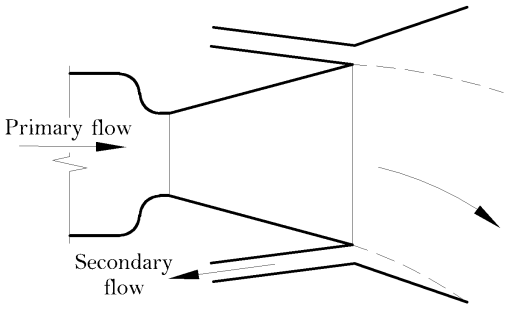


FIGURE 1.4: Schematic of Counter Flow Thrust Vectoring [96]

Similarly to the Counterflow Thrust Vectoring method, the Co-flow Thrust Vector Control approach also uses a secondary fluid flow, leveraging the Coanda effect to assist in thrust vectoring. This technique employs tangential injection to manipulate the direction of the main flow. Here, an additional airstream is merged with the primary flow, adhering to the Coanda effect principles, which steer the primary flow along a curved trajectory. The introduced secondary flow is noticeable on the side closest to the surface, generating a pressure differential perpendicular to the jet flow's central axis. This results in a localized area of low pressure, enhancing the entrainment rate on the side opposite the Coanda surface-bound secondary jet. As a result, thrust vectoring occurs as a result of the altered direction of the primary jet's flow. The efficiency of co-flow thrust vectoring can be further improved by adding another curved surface at the nozzle's rear and introducing a secondary jet that flows parallel to the Coanda surface. Figure 1.5 depicts the schematic for co-flow control in thrust vectoring [87]. The key distinction between co-flow and counterflow thrust vectoring is that the first is based on injection of momentum, whereas the second is based on momentum extraction to manipulate the primary flow.

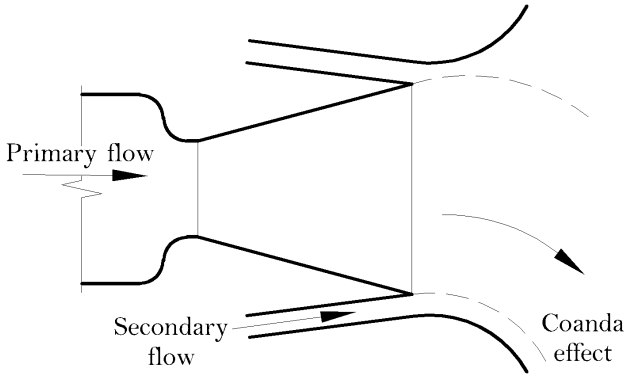


FIGURE 1.5: Schematic Co-flow Thrust Vectoring [2]

The Throat Skewing Control (TSC) concept, also known as Throat Shifting, is based on asymmetric secondary-fluid injection in the throat region of the nozzle. The injected flow modifies the effective throat flow field and changes the direction of the primary jet, producing thrust vectoring without the need for mechanical deflection of the nozzle walls. In contrast to Shock Vector Control (SVC), where secondary injection into the supersonic divergent section intentionally generates shock waves and separation regions, TSC acts mainly by altering the throat flow structure. As a result, the vectoring mechanism does not require the formation of additional shocks in the supersonic part of the nozzle, which can reduce thrust losses associated with shock-induced separation. Figure 1.6 illustrates the TSC schematic. The main design parameters of TSC nozzles, including vectoring angle, thrust coefficient, thrust losses, injector geometry, and flow conditions, have been analyzed in previous studies. The influence of jet-area control on throttling performance has also been examined to ensure that thrust vectoring operation does not significantly degrade nozzle efficiency [61].

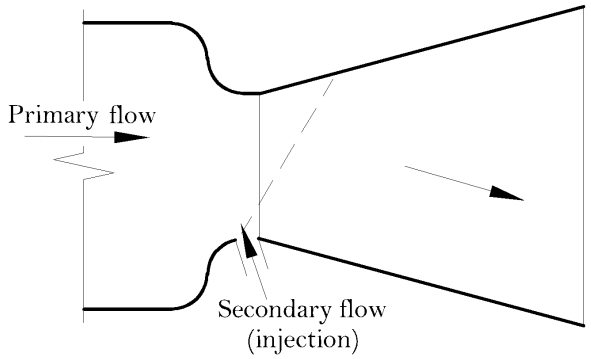


FIGURE 1.6: Schematic Throat Skewing Vectoring Control [62]

A refinement of the TSC thrust vectoring system has led to the development of a Dual Throat Nozzle (DTN), initially named Recessed Cavity Nozzle, which builds upon the Throat Shifting nozzle concept and employs separation control. The DTN features two minimum-area sections, one upstream and one downstream, with a recessed cavity situated within the convergent-divergent-convergent nozzle structure. When secondary fluid is injected near the upstream minimum area, the local blockage and pressure disturbance modify the effective throat flow field, skew the sonic plane, and promote asymmetric flow separation inside the recessed cavity close to the injection region. This asymmetric separation changes the pressure distribution within the nozzle and deflects the primary jet, producing thrust vectoring [16, 30]. Figure 1.7 provides an illustrative schematic of the DTN thrust-vectoring mechanism.

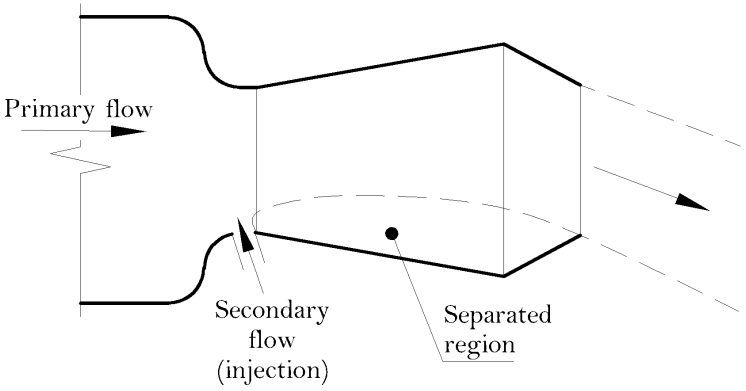
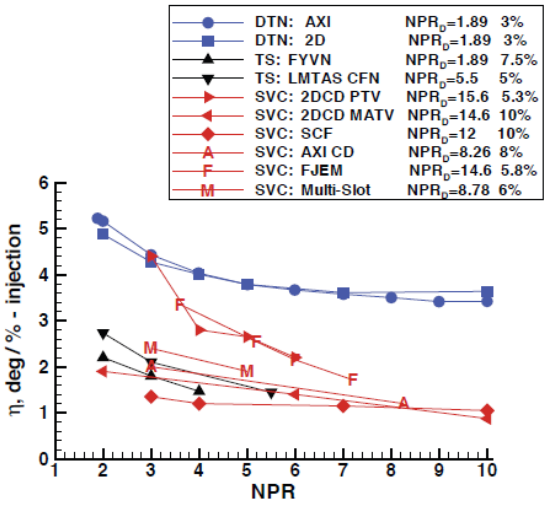
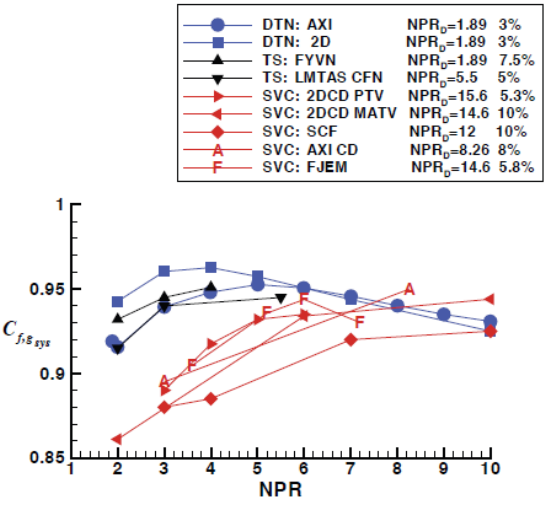


FIGURE 1.7: Schematic of DTN thrust vector control [16]



(a) Thrust Vectoring Efficiency



(b) Thrust Efficiency

FIGURE 1.8: Comparison of DTN's experimental thrust vectoring efficiency and thrust efficiency to TSC and SVC methods [32]

One of the most important parameters characterizing thrust nozzles is their Nozzle Pressure Ratio (NPR), which can be defined as the ratio of the total pressure upstream of the throat p_t to the ambient or nozzle exit static pressure p_0, what can be denoted as $NPR = \frac{p_t}{p_0}$. The research done was paired with previous studies on TSC and SCV methods. For an NPR of more than 1 to 10, to vector the thrust by a certain angle, DTN requires $50 - 25\%$ secondary flow compared to what TSC and SVC require. DTN also shows significantly higher efficiency during vectoring, but only for NPR lower than 5. Comparison of different efficiency and thrust efficiency can be seen in the Figure 1.8.

1.2.2 Research on DTN

The first and most important Computational Fluid Dynamics (CFD) studies and flow analyses through DTN were carried out by the NASA Langley Research Center and included stationary flow through rectangular and axisymmetric nozzles [15]. In 2005, they conducted an experiment that examined the effects of cavity length and secondary injection angle on rectangular flow [30].

The shape of the nozzle is described by characteristic dimensions: Θ_1 and Θ_2, as the angles of the inner divergent and convergent parts, l as the cavity length between the throats: primary throat with diameter d_t and secondary (exit) throat with diameter d_e. These diameters also correspond to the area of the throats A_t and A_e. In addition, the secondary flow injection angle is given as Θ_s. All dimensions are shown in the figure 1.9.

The DTN achieved a maximum pitch thrust vector angle of 15° at NPR = 4. The maximum resultant pitch thrust vectoring efficiency, achieved in NPR = 4, was 6.1° per 1 % of injection at 1 % secondary injection with a corresponding thrust coefficient of 0.968 indicating a decrease in thrust efficiency of less than 0.5 % compared to the case without vectoring. For the experiments described, CFD analyses were also performed to study the effect of geometric parameters of the nozzle, such as cavity divergence angle, cavity convergence angle, and upstream throat height, in the NPR range of 2 to 7 and the injection ratio of up to 3% [16]. The injection ratio can be also described as Mass Flow Ratio (MFR) which can be defined as the ratio of secondary mass flow $\dot{m}_s$ to the sum of both primary and secondary mass flows $\dot{m}_p + \dot{m}_s$, what can be denoted as:

$$MFR = \frac{\dot{m}_s}{\dot{m}_p + \dot{m}_s} \text{ or } MFR = \frac{\dot{m}_s}{\dot{m}_p + \dot{m}_s} \cdot 100\%. \tag{1.1}$$

The principal findings emerging from these initial investigations are as follows:

- Computational results indicate that increasing the cavity divergence angle beyond 10°is detrimental to the efficiency of thrust vectoring, while increasing the cavity convergence angle from 20°to 30°improves the efficiency of thrust vectoring at NPR greater than 2.
- Pulsed injection is not more efficient than steady injection for DTN.
- DTN has achieved levels of thrust vectoring efficiency greater than any other FTV concept reported in the open literature.
- The DTN technique does not compromise the thrust ratio of the system, unlike SVC methods, which incur large thrust penalties by creating the shock required for the turn.
- CFD inaccurately predicted the pressure losses that occur in the large regions of separated flow of the cavity
- CFD can predict the trends of thrust vectoring efficiency,
- Increasing the cavity convergence angle improved thrust vectoring efficiency, but at the expense of the discharge coefficient.
- The best vectoring configuration achieved thrust vectoring efficiencies of about 4 to 6% per degree secondary injection, with resultant pitch thrust-vector angles up to 12°, and minimal thrust losses at secondary injection flow rates of 3% or less, but had a detrimental effect on discharge coefficient that could cause engine operability problems if not addressed.

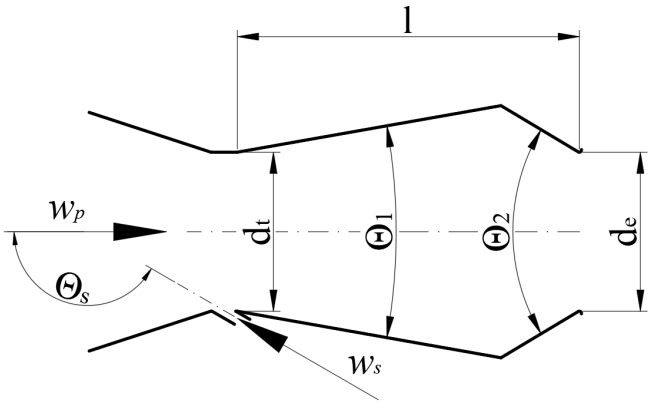


FIGURE 1.9: DTN characteristic dimensions

In 2006, the same group of researchers published additional results showing the impact of various parameters and more dependencies more widely. The article [31] shows

the effect of the secondary injection rate at a NPR of 4. The effects of modifying cavity divergence angle Θ_1, convergence angle Θ_2 and cavity shape on internal nozzle performance were also investigated, as were effects of injection geometry and injection channel shape (hole or slot). That leads to following conclusions:

- A curved cavity apex provided improved thrust ratios at some injection rates.
- Decreasing cavity divergence angle Θ_1 had a negative impact and increasing cavity convergence angle Θ_2 had a positive impact on thrust vector angle and thrust efficiency.
- Overall nozzle performance suffered without secondary injection.
- Injection holes were more efficient than the injection slot in the injection rate range, but the slot generated larger thrust vector angles for injection rates less than 4°of the primary mass flow $\dot{m}_p$.

Chen Q. et al. [12] investigated the non-uniqueness of steady-state solutions in nozzle flows with uncertain initial conditions. The study used generalized Polynomial Chaos (PC) methods to assess how variations in the initial flow field affect the resulting steady-state solution, with particular attention given to the predicted shock wave position. Although individual flow realizations contain discontinuities associated with the shock, the corresponding PC expansion modes were shown to vary smoothly in the spatial coordinate (x). This distinction is important because the shock position is a discontinuous quantity in the stochastic space, even when the expansion modes remain spatially smooth. The authors demonstrated that the probability density function of the shock location can be predicted accurately when the variance of the uncertain initial conditions is small. However, for larger uncertainty levels, the convergence of the PC expansion becomes slower and a larger number of expansion terms is required to obtain acceptable accuracy. This behavior results from the non-smooth dependence of the shock position on the random input variables. The study therefore highlights both the usefulness and the limitations of PC methods for predicting shock wave locations in nozzle flows affected by uncertain initial conditions.

In the article [8] Bellandi E. and Slippey A. noted that the above mentioned studies did not take into account analytical methods to describe the flow through DTN. CFD analysis and hydraulic analogy were used to verify the analytical solutions. A new DTN wall contours was proposed, but the analytic methods mentioned above could not be quantitatively verified.

Japanese and Korean studies from 2010 [83] consisted of CFD analysis of a rectangular nozzle in the range of NPR up to 10 and the proportion of secondary flow from 0°to 10°. Based on the studies, it was concluded that:

1. Increasing secondary flow by 6°, increases the values of the thrust vector angle, but a further increase in secondary flow does not change the values of the thrust vector angle.

2. Secondary flow affects The nozzle discharge coefficient which can have a negative impact on engine operation.

3. The system thrust coefficient is highest for $NPR = 5$, while further out, it decreases as NPR increases. Also at $NPR = 5$, DNT generates the best thrust vectoring.

In 2016 in 2017 Ferlauto M. and Marsilio R. published the results of numerical analyses of the dynamic characteristics of DTN [28] [29]. The study focusing on the dynamic response of a DTN under both open and closed-loop control scenarios. It emphasizes the importance of investigating the nozzle system's dynamic response and control strategies for reliable thrust vectoring, where closed-loop model predictive control, based on "AutoRegressive with eXogenous input" (ARX) models, has been successfully implemented. Thrust vectoring is facilitated through secondary flow injections that induce local flow separations and asymmetric pressure distributions, effectively vectoring the primary jet flow. The research analyzes nozzle performances and thrust vector angles across a broad spectrum of nozzle pressure ratios and secondary flow injection rates, presenting the potential of FTV technique for precise and efficient aircraft control.

In 2014, researchers Gu R. and Vu J. [41] proposed the Bypass Dual Throat Nozzle concept (BDTN), in which secondary flow comes directly from the combustion chamber. Such a solution seems logical, but the Sprint missile experiments [63], already described above, showed that moving from a cold air experiment to a full-scale real-engine application will reveal problems with the control valves, which will not be able to handle high pressure, high temperature, and high flow rate simultaneously. The square-shaped nozzle case experiment and calculations carried out confirm the conclusions of previous publications, but a quantitative summary of the results of the experiment and CFD analysis is missing. It is worth noting the Schlieren images, which show some phenomena, but in an unprocessed form allow only a subjective evaluation.

Subsequent experimental investigations of the BDTN conducted by Gu R. and Xu J. [40] were designed to determine its dynamic characteristics and response time. The experiments revealed static pressure oscillations due to the non-stationary nature of the flow. The decision to apply a low-pass filter and effectively eliminate this phenomenon from the analysis is methodologically debatable. Although not explicitly described in the

text, photographic documentation clearly shows that the pressure signals were transmitted to the transducers through long tubing or lines, which further compromises the reliability of the reported results.

In 2020, tests were carried out in the same wind tunnel on an axisymmetric nozzle [95]. These experiments were conducted with reference to two-dimensional numerical simulations published in 2017 [94]. The study identified a phenomenon termed the "DTN starting problem" and emphasized the necessity of performing three-dimensional numerical analyses to achieve an accurate and comprehensive understanding of the underlying flow physics.

Numerical investigations of Wu K. and Kim H. [101] [100] revealed that the bypass mass flow is significantly influenced by both friction and geometry choking, each playing distinct roles in mass flow regulation. Notably, as the contraction area ratio of the bypass passage increases, there is a continuous decrease in both the vectoring angle and bypass mass flow ratio, while the thrust coefficient, thrust efficiency, and total pressure loss concurrently rise. The research underscores the growing research value of FTV nozzles and positions DTN technique as important for advanced aircraft maneuverability. Furthermore, the study examines the effects of nozzle pressure ratio, injection to mainstream momentum flux ratio, and the setup angle of the slot injector on systemic performance, aiming to refine control effectiveness. It also points out the asymmetry in the flow field of the 3-D rectangular DTN, underscoring the importance of considering the full domain to accurately capture flow field characteristics, thereby offering useful data for engineering designers in the field. The confusion in the work presented was the use of geometric bypass flow throttling during the CFD anlysis, instead of setting a boundary condition, which would increase reliability.

In 2021 Estakhrsar M. and Moghaddam H. [42] presented results of experiment and numerical simulation of 3-D turbulent airflow in square BDTN. The results appear to be in line with previous predictions, but the experiment conducted raises concerns due to its attempt to learn flow angles from the Schlieren image.

In 2021, Salimi M. et al published results of a computational investigation into the performance of a fuel-injected planar DTN[77]. Utilizing numerical investigation methods, the study employed three-dimensional compressible reacting simulations to analyze the effects. Key findings revealed that slot injection of methane fuel at a secondary injection rate of 9°of the primary flow rate achieved a maximum vector angle. The study further delineated that light fuels significantly enhance thrust ratio, vector angle, and vectoring efficiency, whereas heavy fuels contribute to a better discharge coefficient. This was the first published study of DTN with fuel injection

The study of Maruyama Y. et al [57] assesses the performance of a DTN using numerical analysis and direct simulation with Monte Carlo method. The results mostly agree with previous studies, but the computational method used showed a larger vectoring angle at NPR = 2. The inaccuracies in the Gu R. and Xu J. articles ([40]) mentioned here were also pointed out. The possibility of moving the injection point farther into the divergent part of the nozzle was raised, but according to the analyses, it does not give any benefit. The most important observation is that widening of the divergent part to a value of 1.50 of area of the first throat to stop unintended thrust oscillations during zero secondary jet flow.

1.3 Motivation

The analysis of thrust vectoring methods presented in the preceding sections indicates that DTN represents a promising FTV solution, primarily due to its high vectoring effectiveness and the absence of moving mechanical components within the nozzle itself. Despite nearly two decades of research on DTN, the existing body of literature does not exhaustively address all aspects of the concept, and further systematic investigation is required before it can be deemed suitable for practical implementation in aircraft propulsion systems.

The majority of published investigations concentrate on steady-state performance, thrust vectoring efficiency, and discharge coefficients, as well as on selected flow structures obtained from numerical simulations. Significantly less attention has been paid to the dynamic response of the nozzle following the initiation of secondary injection. This constitutes an important limitation, because the practical application of a DTN requires not only the attainment of a prescribed thrust vector angle, but also a detailed understanding of how rapidly the flow field responds to the control input.

In particular, there remains a lack of experimental data characterizing the dynamic response of a DTN, specifically the response delay and time constant for a prescribed nozzle pressure ratio (NPR). These parameters are critical for the design and implementation of thrust vector control systems, especially because direct in-flight measurement of the thrust vector angle is difficult or infeasible. Consequently, control of the thrust vector orientation must rely on pre-identified nozzle characteristics and validated dynamic response models.

Another critical research gap pertains to the inherently unsteady nature of the flow within the DTN. The interactions among the primary flow, secondary injection, separated regions, shock structures, and wall pressure fluctuations can exert a substantial influence on the stability of the thrust-vectoring process.

1.4 Aim and objectives of the thesis

The primary objective of this thesis is to experimentally investigate the transient and non-stationary behavior of a Dual-Throat fluidic thrust-vectoring nozzle and to identify selected dynamic parameters that are relevant for the characterization and potential control of operation.

Based on the research questions formulated at the outset of the study, the following specific and detailed objectives were established:

1. **Determination of the DTN time constant and delay time.**
   This objective concerns the characterization of the dynamic response of the nozzle following the activation of the secondary injection. Specifically, the temporal evolution of the pressure signal at selected wall pressure taps is analyzed to determine the delay time and the time constant of the DTN under the operating conditions investigated. These parameters quantify the dynamic response of the nozzle flow field to the control input and form the basis for subsequent control oriented modeling.

2. **Experimental determination of wall pressure frequency and amplitude, during quasi-steady operation of DTN.**
   This objective concerns the characterization of pressure fluctuations arising during both vectorized and non-vectorized operation of the DTN. The investigation includes the identification of pressure oscillations under constant inlet-flow conditions, including the determination of their spectral (frequency) content and amplitude. It is essential to evaluate the degree to which the flow in the DTN can be regarded as quasi-steady, and to determine whether inherently unsteady phenomena must be explicitly incorporated into the analysis.

Chapter 2

Background information and theory

2.1 Thrust vector control - definition

The thrust vector control (TVC) of a jet engine is the deviation of the engine jet from the direction corresponding to the cruise flight mode. This is to provide directional control. Instead of relying solely on aerodynamic control surfaces, like ailerons or rudders, TVC provides the ability to alter the direction of the thrust itself [35].

Directly, the thrust vectoring process is defined by the normal force F_N. The resultant pitch thrust vector angle δ_p is the angle between the vectors of the normal and axial sequence F_A, what is designated as

$$\delta_p = tan^{-1}\frac{F_N}{F_A}, deg. \tag{2.1}$$

The resultant thrust is the geometric sum of the axial and normal force is designated as

$$F_r = \sqrt{F_A^2 + F_N^2}, N. \tag{2.2}$$

It should be noted here that measuring the thrust vector angle is difficult to achieve under laboratory conditions, with the test section rigidly attached to the driver tube or tank, and will not be performed as part of this study. All results described in the literature are based on forces determined in CFD analyses. An actual measurement would only be possible on a test bench equipped with a free motion test section or on an actual engine.

2.2 Thrust vectoring quality evaluation

2.2.1 Introduction to ideal isentropic and actual parameters

Ideal and actual parameters should be taken into account for quality evaluation.

From primary flow (equivalent of non-vectorized engine):

- $F_{i,p}$: ideal isentropic thrust of primary flow (equivalent of ideal non-vectorized thrust),
- $\dot{m}_{i,p}$: ideal primary mass flow,
- $\dot{m}_p$: actual primary mass flow.

Please note that the actual mass flow is lower, among other reasons, because the effective critical cross section is much smaller than the geometrical cross section (vena contracta effect) 2.1.

From secondary flow aka fluid injection:

- $F_{i,s}$: ideal isentropic thrust of fluidic injection flow (which can be compared to an independent thruster or engine running perpendicular to the main motor,
- $\dot{m}_{i,s}$: ideal fluidic injection mass flow,
- $\dot{m}_s$: actual fluidic injection mass flow.

2.2.2 System discharge coefficient

The relationship between ideal and actual mass flow can be expressed by the discharge coefficient C_d, which is the dimensionless ratio of the actual discharge to the ideal discharge. This parameter proves beneficial in calculating the irreversible losses related to a specific equipment within a fluid system, or the "friction" that the equipment creates against the flow.

$$C_d = \frac{\dot{m}}{\dot{m}_i}, \tag{2.3}$$

where ideal mass flow $\dot{m}_i$, in compressible flow conditions, is defined as

$$\dot{m}_i = A\, p_t \sqrt{\frac{\kappa}{RT_t}} \left(\frac{2}{\kappa+1}\right)^{\frac{\kappa+1}{2(\kappa-1)}}. \tag{2.4}$$

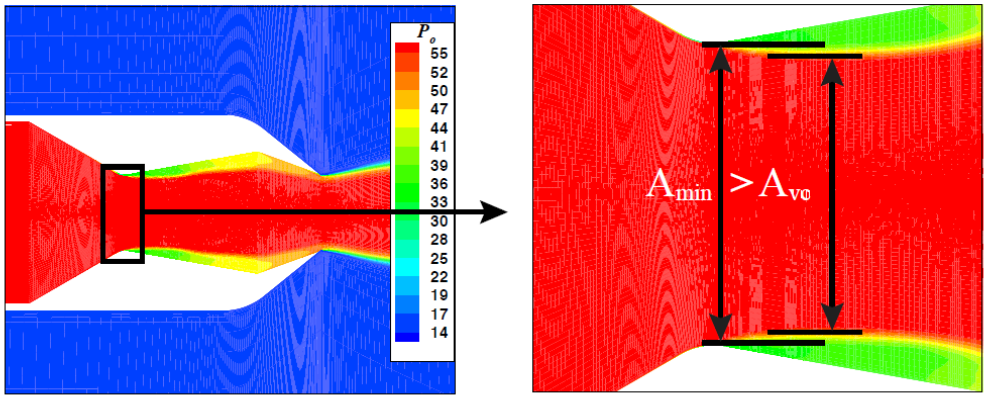
where A is the throat area, p_t is the upstream stagnation pressure, T_t is the upstream stagnation temperature, κ is the specific heat ratio, and R is the specific gas constant of

air. The equation is valid for choked flow conditions, i.e. when the pressure ratio is equal to or lower than the critical pressure ratio.

For any type of FTV, the system discharge coefficient $C_{d,p}$ or $C_{d,prim}$ (also known as discharge coefficient of primary nozzle) is a number that represents the ratio of the actual flow rate of the primary nozzle $\dot{m}_p$ to its theoretical or ideal mass flow $\dot{m}_{i,p}$, considering the influence of secondary jets. Mathematically, it is defined as

$$C_{d,p} = \frac{\dot{m}_p + \dot{m}_s}{\dot{m}_{i,p}}. \tag{2.5}$$

The primary mechanism responsible for the reduction in actual mass flow rate (as well as the associated pressure losses) is the vena contracta effect. This phenomenon entails a decrease in the actual cross-sectional area within the nozzle or orifice, denoted as A_{min}, arising from the finite inertia of the fluid. The impact of this effect is particularly pronounced in choked flow conditions, for which the mass flow rate is strongly governed by the local cross-sectional area. Figure 2.1 presents a representative CFD visualization of this behavior, in which the reduced effective area is indicated as A_{vc}.



(a) Full view of primary flow through nozzle geometry (white). (b) Enlarged view of upstream throat.

FIGURE 2.1: Computational total pressure flow contours show the decreased effective area (A_{vc}) from the vena contracta effect at the upstream minimum area (A_{min}) [31].

System discharge coefficient depends on the geometric dimensions of the nozzle and the NPR. For example, for a flat DTN with secondary flow and $NPR = 3$, one manages to obtain $C_{d,p} = 0.87 - 0.89$ [31]. For an axisymmetric DTN, it manages to achieve $C_{d,p} = 0.78 - 0.83$, depending on NPR [32], what is shown in the Figure 2.2. Above $NPR = 3$, system discharge coefficient is almost constant and equals to $C_{d,p} \approx 0.83 \pm 0.01$. These values will be used later to evaluate the results of the experiment.

The system discharge coefficient for the DTN nozzle is an extremely important parameter because it allows to know the actual primary flow mass flow $\dot{m}_p$, which is difficult to measure during a tube or tank dump experiment. Knowing this parameter, which is

actually constant for the nozzle if the NPR is greater than the critical pressure ratio (about $NPR = 1.89$ for a perfect gas) allows for a proper comparison of results.

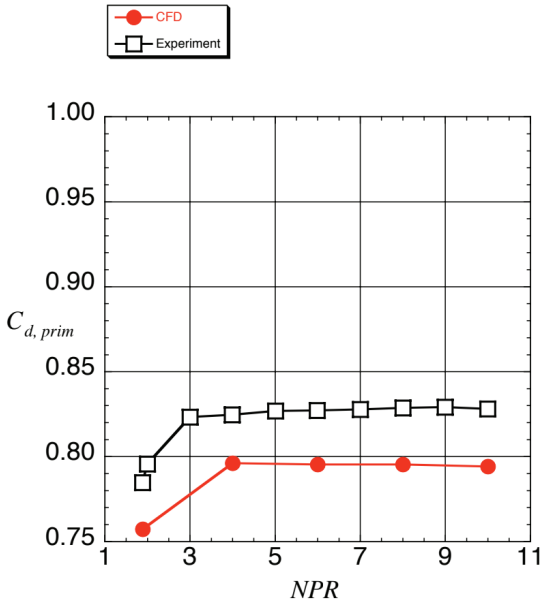


FIGURE 2.2: Nozzle internal performance for an 13-60 geometry, operating over a range of NPR with 3% fluidic injection [32]. Open symbols experimental data. Solid symbols computational data.

2.2.3 System thrust ratio

The system thrust ratio $C_{fg,sys}$ is used to assess the overall propulsive efficiency of FTV systems. This parameter allows the thrust-vectoring performance to be evaluated not only in terms of deflection angle, but also in terms of the thrust penalty associated with generating the vectoring effect. System thrust ratio is described as ratio of the actual resultant thrust F_r (Eq.2.2) to the sum of the: ideal isentropic thrust of the primary flow $F_{i,p}$ and the ideal isentropic thrust to the secondary (injected) flow $F_{i,s}$. As already mentioned, this is a simple comparison of the actual resultant thrust to two ideal, isentropic, motors (thrusters) running perpendicular to each other, which can be written as follows

$$C_{fg,sys} = \frac{F_r}{F_{i,p} + F_{i,s}}. \tag{2.6}$$

This ratio is also dependent on the geometry, as well as the secondary flow rate. For example, for a flat DTN and $MFR = 0.03$, $C_{fg,sys} = 0.95 - 0.96$. On the other hand, for an axisymmetric nozzle, with the same flow rate $C_{fg,sys} = 0.76 - 0.77$.

2.2.4 Thrust vectoring efficiency

The resultant pitch thrust vectoring efficiency, denoted as η, represents the efficiency of a FTV nozzle in achieving a desired pitch vectoring angle. It is expressed in degrees per 1% percent of secondary injection (2.7). It is worth noting here that this parameter is actually independent of DTN, but strongly related to the secondary flow ratio and thus the thrust vector angle δ_p [32]. For DTN, the greatest changes occurring in the range up to 0.05 of MFR. Beyond this, the vectoring angle does not increase and the efficiency decreases slightly.

$$\eta = \frac{\delta_p}{\frac{\dot{m}_s}{\dot{m}_s+\dot{m}_p}} \cdot 100\%. \tag{2.7}$$

2.3 Flow in DTN

2.3.1 Non-vectorized flow

In the non-vectorized operating mode, no secondary flow is supplied to the injection slot of DTN and the flow field should remain nominally axisymmetric. For a DTN geometry in which the primary and secondary throat areas are equal and their ratio $A_e/A_t = 1$, the nozzle may be regarded, in a first approximation, as a converging nozzle with an internal recessed cavity. Such a configuration is suitable mainly for relatively low NPRs, for which high expansion efficiency is not the dominant design requirement.

However, the internal flow structure of a DTN is more complex than that of a simple converging nozzle. The presence of two minimum area sections and the cavity between them leads to additional losses. In particular, the effective flow area at the upstream throat is reduced by mentioned vena contracta effect, while separated and recirculating flow regions may develop inside the cavity. These phenomena reduce the overall thrust performance compared with an ideal converging nozzle of the same nominal throat area. Therefore, even in the absence of secondary injection, the DTN geometry introduces a performance penalty associated with internal separation, mixing losses, and non-uniform acceleration of the primary flow.

Another important feature of the non-vectorized DTN flow is its non-stationary character. Previous mentioned numerical and experimental investigations indicate that the flow inside the cavity may exhibit pressure and velocity fluctuations even without secondary injection. These oscillations are related to the interaction between the separated

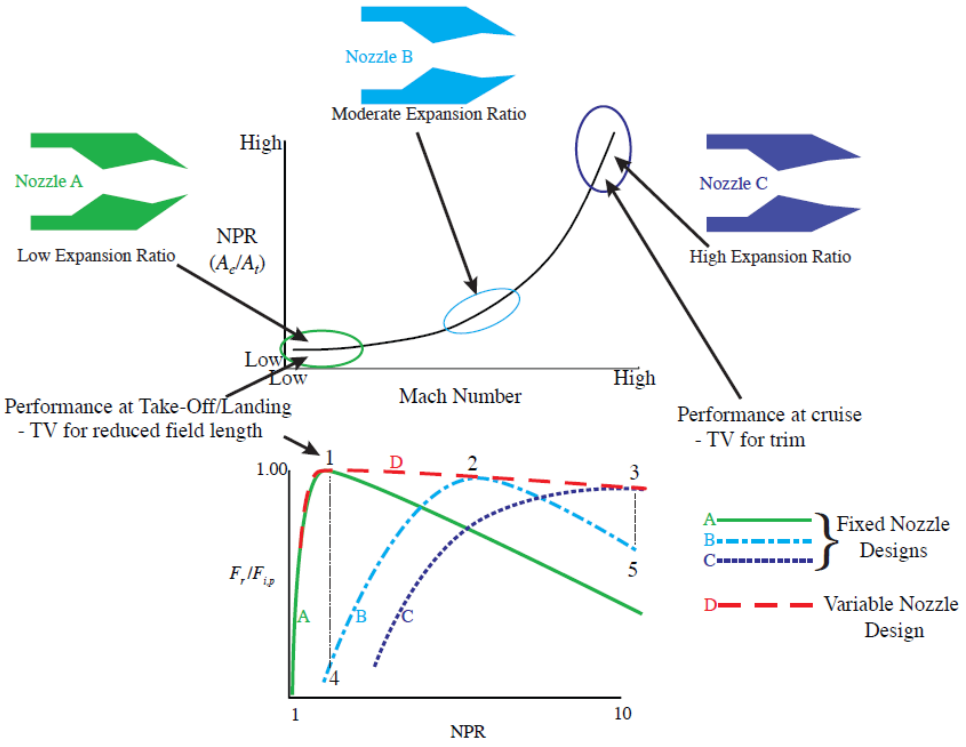


FIGURE 2.3: Variable area nozzle operating curve [32]

flow region, the shear layer, and the downstream throat. This issue is particularly important in the present work, because such fluctuations may affect wall-pressure measurements and complicate the interpretation of the dynamic response of the nozzle. The non-stationary behavior of the flow is discussed in more detail in the following sections.

For engines operating at higher NPR, a fixed-area DTN with $(A_e/A_t = 1)$ is not optimal from the standpoint of propulsive efficiency. In such conditions, a variable-area nozzle exit would theoretically allow the system to recover part of the expansion efficiency during non-vectorized operation, while retaining the DTN configuration during maneuvers requiring thrust vectoring. Nevertheless, a quantitative assessment of the mass, complexity, reliability, and performance trade offs associated with such a solution requires a more advanced stage of technology development. The influence of variable geometry on nozzle operation at different NPR values was conducted by Flamm et al. [32] and it's illustrated in Fig. 2.3. The figure shows two graphs: the first illustrates a proposal to increase the nozzle area as Mach number increases, which allows thrust vectoring efficiency to be maintained during low speed maneuvers. The second graph shows the relationship between DTN thrust efficiency and NPR for various nozzle exit diameters. It is clearly visible here that an excessively small nozzle diameter dramatically reduces efficiency at higher NPRs.

A historical example of a propulsion system in which such a compromise was acceptable is the Walter HWK 109-509A-1 rocket motor used in the Messerschmitt Me 163

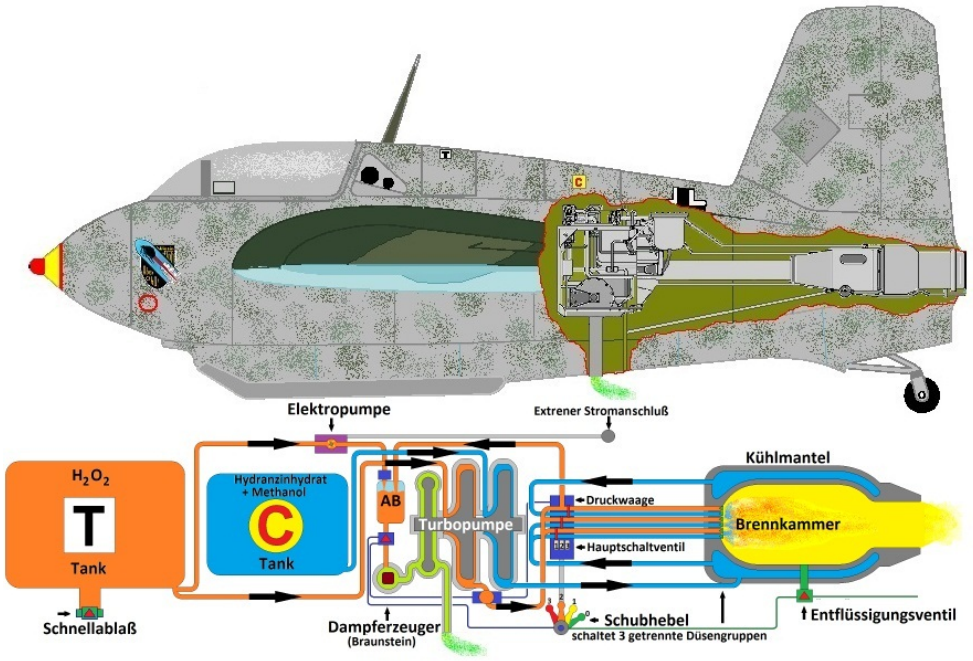


FIGURE 2.4: Schematic of the Walter HWK 109-509A-1 rocket motor in Messerschmitt Me 163 Komet [4]

Komet [66]. The engine (Figure 2.4) was designed under severe technological limitations related to propellant handling, combustion chamber cooling, turbo pump operation, and compact installation inside a very small airframe. As a result, NPR had to remain relatively limited compared with an ideal high expansion rocket nozzle. In such a case, where the available nozzle length and expansion ratio are constrained and the operating time is short, the loss of expansion efficiency introduced by a DTN would be less critical than in a conventional high efficiency propulsion system. At the same time, the ability to generate thrust vectoring forces without large external moving surfaces would make the DTN concept particularly attractive for this class of compact rocket powered aircraft.

2.3.2 Flow with vectoring

The general mechanism of vectorized flow in a DTN is illustrated schematically in Figure 2.5. In the vectorized operating mode, thrust deflection is achieved by introducing a secondary flow into the upstream throat region. The injected jet locally disturbs the primary flow field and alters the effective throat area. Consequently, the sonic region ceases to be symmetric with respect to the nozzle axis. Instead, the location of the minimum effective flow area is displaced and skewed, resulting in an asymmetric acceleration of the primary stream.

The physical mechanism responsible for thrust vectoring in a DTN configuration differs fundamentally from that typically observed in classical SVC nozzles. In a DTN,

the secondary jet does not primarily induce vectoring by generating a strong oblique shock in the divergent section of a conventional convergent–divergent nozzle. Instead, the injected flow modifies the flow field in the throat region and promotes a controlled flow separation within the recessed cavity located between the two throats. Under these conditions, the primary flow tends to remain attached to the wall opposite the injection side, while a separated flow region develops in the vicinity of the injection side. This asymmetry gives rise to a non-uniform wall pressure distribution, which in turn produces a lateral component of the resultant thrust force.

The magnitude of the vectoring effect depends mainly on the secondary to primary mass flow ratio, NPR, injection angle, injection span, and cavity geometry. Prior investigations have demonstrated that the DTN concept can achieve comparatively high vectoring efficiency at low to moderate NPR values, while incurring smaller thrust penalties than several alternative FTV configurations [16, 31]. However, these advantages are strongly coupled to internal flow losses, reductions in discharge coefficient, and the detailed topology of the separated flow within the cavity.

Due to the complex interaction among the injected jet, the separated cavity flow, and the downstream throat, the sonic region in a vectorized DTN cannot be represented as a single, well defined sonic plane. Consequently, a one-dimensional nozzle model is inadequate for capturing the underlying vectoring mechanism. Accurate assessment of DTN operation therefore necessitates detailed local flow field characterization, including measurements of wall pressure distributions and their unsteady evolution following the onset of secondary injection.

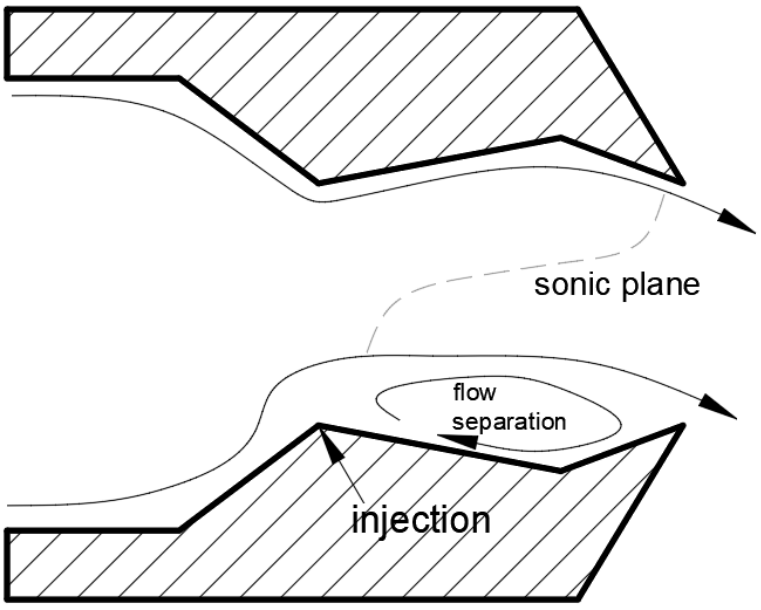


FIGURE 2.5: Sketch of theoretical flow through DTN [15]

2.4 Current dynamic experimental studies on DTN

The first attempt to identify the timing of DTN responses was made by Gu R. and Xu J. [40]. They conducted experimental tests for a flat bypass DTN (BDTN) with a fast wall pressure measurement (10 kS/s/ch) and a cyclically open secondary flow valve. The results after spectral analysis showed a strong oscillation of pressures in the region of $3\,kHz - 4\,kHz$. A low-pass filter was applied to 'clean up' the signal. The final conclusion shows that the stream deflection rate is between $30\,deg/s$ and $50\,deg/s$, depending on the NPR. The mentioned pressure oscillation indicates the non-stationary nature of the flow field, which was not recognized in earlier experiments due to a sampling frequency too low [28] [29]. Unfortunately, a number of considerations make it necessary to approach the presented data with great scepticism:

- Photographic documentation of the experimental setup reveals the presence of an exceptionally long pressure hose. between measured point and transducers (on the order of a few meters), which is of great importance, as the length of the hose and its volume influence the shape and rise time of the pressure signal.
- The use of a low-pass filter is questionable because, as shown in the spectral analysis, the amplitudes of the filtered signal are greater than half the amplitude of the signal used to reach the conclusions.
- The terminology used in that study is not fully consistent with the usual distinction between dynamic pressure and time-dependent static pressure, which makes the interpretation of the reported response parameters less straightforward.

Ferlauto M. and Marsilio R. performed a series of URANS analyses [28] [29] and validated them with the results obtained by Flamm, Deere et al. Calculations were carried out to find the response time of the system, for future open- or closed-loop control systems. The input parameter was a step change in pressure and the response was a change in vector angle over time. Specific time values were set, but it was found that DTN flow field is its intricate and nonlinear dynamic response, which arises from the significant interactions between shifting shocks, boundary layers, and regions of separated flow. The calculated time response values are of the order of ms, while the Gu R. and Xu J. experiment showed times of whole seconds [40]. Wang Y. Xu J. et al. [94] performed a numerical study on the axisymmetric flow through the DTN. Non-stationary analyses showed a pulsation of the fluid mass flux in the outlet section in the range of $0.26\,{}^{kg}/_{s} - 0.32\,{}^{kg}/_{s}$. Symmetric fluid injection into the cavity between the throats was

proposed as a solution to the flow stability.

Previous published studies by the author of this dissertation on flow instability through DTNs [45] have confirmed the phenomena described above. Experimental tests performed on a transonic wind tunnel stand with a 3D-printed DTN nozzle and high speed pressure transducers, allowed a more detailed analysis of the nozzle's behavior during operation. The results showed that the amplitude and frequency of pressure fluctuations can significantly affect the performance of DTN. The phenomenon of turbulence and flow instability had a significant impact on the experimental results. Improved measurement methods and non-stationary analysis contribute to a better understanding and optimization of these technologies. Figure 2.6 shows the pressure waveform over time. Figure 2.7 shows the periodogram with the dominant frequency corresponding to the pressure pulsation on the nozzle walls marked.

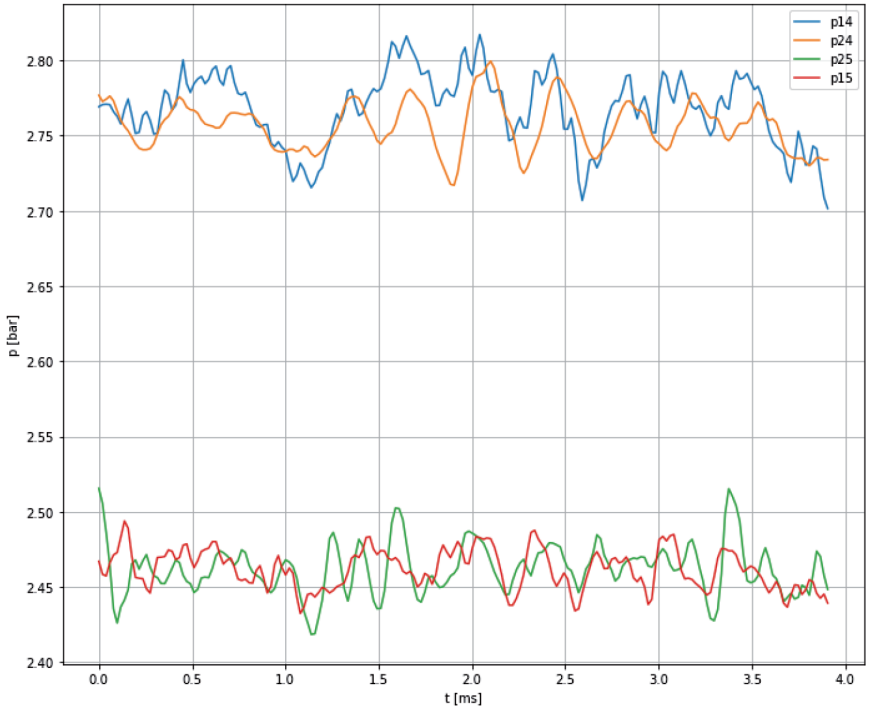


FIGURE 2.6: Pressure fluctuations in axisymmetric flow[45]

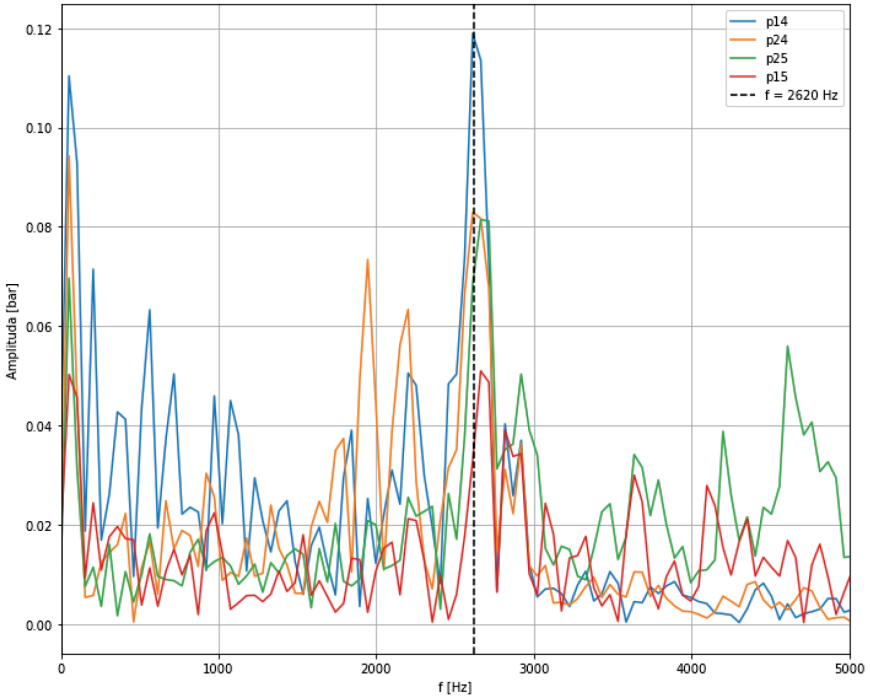


FIGURE 2.7: Pressure amplitude periodogram [45]

2.5 Implications of the Literature Review

The literature review leads to several conclusions relevant to the present study:

- DTN nozzles are among the most efficient FTV concepts reported in the literature, particularly at low and moderate NPR values. Their advantage is mainly associated with the ability to obtain relatively large vectoring angles with a small secondary mass flow rate.
- The propulsive performance of a fixed geometry DTN deteriorates at higher NPR values. This is primarily related to the limited expansion capability of the nozzle and to additional losses generated by the DTN geometry, including separation, vena contracta effects, and mixing losses in the recessed cavity.
- The DTN concept is therefore most suitable for propulsion systems operating with relatively low nozzle expansion ratios. Ramjet propulsion systems are a natural candidate for such applications. For turbine jet engines operating over a wider NPR range, the use of variable exit area geometry may be necessary to reduce the efficiency penalty in non-vectorized operation.
- The influence of the main geometric parameters of DTN nozzles has been extensively investigated. Previous studies indicate that the cavity divergence angle, cavity convergence angle, cavity length, injection angle, and injection span affect both vectoring efficiency and thrust performance. However, these parameters are not independent, and the optimum geometry depends on the selected operating range and performance criterion.
- The most frequently used performance parameters include the thrust vector angle, thrust vectoring efficiency, discharge coefficient, and system thrust ratio. In the literature, these quantities are usually evaluated together for specified boundary conditions, because a high vectoring angle alone does not necessarily indicate favourable overall nozzle performance.
- Published studies indicate that the DTN flow field may be non-stationary even in the absence of secondary injection. Pressure oscillations, flow separation, and interactions between the shear layer and the downstream throat can influence both the mean flow field and the dynamic response after the onset of secondary injection.
- Although numerical analyses have provided substantial insight into the steady and unsteady behavior of DTN nozzles, experimental data on the dynamic response of axisymmetric DTN configurations remain limited. In particular, there is a lack of studies in which the transient wall pressure response is measured directly under short duration flow conditions.

- The available experimental studies on DTN response time provide useful reference information, but their quantitative reliability is limited by methodological issues. These include the use of long pressure lines, low-pass filtering of physically significant pressure fluctuations, and insufficient separation between the mean transient response and flow induced oscillations.
- No published study was found in which a Ludwieg tube facility was used as the driver section for experimental investigation of DTN dynamic response. This indicates that the applicability of such a short duration test facility to DTN studies remains insufficiently documented.

These conclusions define the research gap addressed in the present dissertation. The existing literature confirms the potential of DTN nozzles for efficient fluidic thrust vectoring, but also shows that their internal flow is strongly affected by separation, losses, and non-stationary phenomena. Therefore, the present work focuses on the experimental identification of the dynamic response of an axisymmetric DTN under short duration flow conditions. Particular attention is given to the influence of secondary injection on the wall pressure distribution, the characteristic response time of the nozzle, and the role of pressure pulsations in the interpretation of experimental data. In this context, the use of a Ludwieg tube facility and high speed pressure measurements provides a basis for assessing whether the transient behavior of the DTN can be described in a repeatable and physically meaningful way.

Chapter 3

Geometry and CFD Analysis

3.1 Nozzle geometry

In the experiments in this study, an axisymmetric DTN was chosen. The axisymmetric shape is more commonly found in propulsion systems. From the point of view of the experiment, the rectangular nozzle should be more convenient, due to the quasi 2D flow field and the possibility of following Schlieren image inside, however, rectangular nozzles have a slightly lower efficiency. For technological reasons, the nozzle used includes only conical elements. The study considered only the extreme case, where the ratio of the both throats is $A_e/A_t = 1$, as this is closest to the concept of "pure" DTN.

The influence of DTN geometrical parameters was studied in detail by Flamm, Deere, and co-workers circa 2006 [17, 32]. Their CFD analyses were performed for nozzle pressure ratios ranging from 1.89 to 10 and for a secondary mass flow ratio of approximately 3% relative to the primary flow. This level of secondary injection was sufficient to achieve a thrust vector angle of approximately 15°, while the maximum vectoring capability of the DTN concept was reported to be about 20°. Complementary experimental tests were performed for $NPR = 1.89$ and secondary injection mass flow ratios ranging from 1% to 11%.

The geometry adopted in this thesis was based on the reference Deere-Flamm axisymmetric DTN configuration. In the original studies, the corresponding configuration was denoted as *"202"* in the CFD analysis and as *"13-60"* in the experimental investigation. This configuration was selected because it is one of the best documented and most directly comparable DTN geometries available in the literature. Following the reference design, the same non-dimensional geometric proportions were retained, including the relative throat dimensions, cavity length, wall angles, and secondary injection slot arrangement. The geometry was then scaled according to the selected critical diameter,

which was chosen to match the capabilities of the test facility, corresponding to a primary mass flow of approximately 0.5kg/s.

The reference experimental studies also reported a small jet deflection, of about 4°, even in the absence of secondary injection. This behavior was attributed to a slight geometric asymmetry caused by the presence of the injection slot. To eliminate this possible source of bias, the geometry used in the present study was modified by introducing two symmetrically placed slots. One slot was used for secondary injection, while the opposite slot remained inactive and served only to preserve geometric symmetry in the thrust vectoring plane.

The dimensions are as follows:

first throat diameter	$d_t = 30\,mm$,
exit throat diameter	$d_e = 30\,mm$,
upstream divergent cavity ramp angle	$\Theta_1 = 20\,deg$,
downstream convergent cavity ramp angle	$\Theta_2 = 60\,deg$,
cavity length	$l = 64\,mm$,
secondary flow injection angle	$\Theta = 150\,deg$,
Injection span angle Fig. 3.1	$\phi = 60\,deg$.

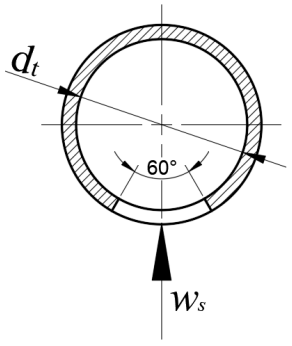


FIGURE 3.1: Secondary flow injection span

3.2 Boundary conditions

The CFD boundary conditions were defined to reproduce the operating conditions of the experimental DTN working in the Ludwieg tube facility. The working medium was dried compressed air. Since the nozzle discharged to the surroundings, the external domain was defined as a pressure outlet corresponding to atmospheric discharge. Air was treated as an ideal gas and the energy equation was enabled, which is necessary for the considered compressible transonic flow with local choking and shock structures.

The main operating parameter was the total pressure in the driver section upstream of the primary nozzle inlet $p_{0,3}$, defined as

$$p_{0,3} = NPR \cdot p_a, \tag{3.1}$$

where p_a is the ambient pressure. The secondary injection mass-flow rate $\dot{m}_s$ was calculated from the prescribed mass flow ratio MFR and the primary mass flow rate $\dot{m}_p$ as follows:

$$\dot{m}_s = \frac{MFR \cdot \dot{m}_p}{100 - MFR} \tag{3.2}$$

where $\dot{m}_p$ is the primary mass flow rate and $\dot{m}_s$ is the secondary injection mass flow rate.

For the representative case of $NPR = 2.85$, the primary flow is choked in the critical cross-section. Assuming an ideal gas, the theoretical, ideal, primary mass flow rate may be estimated from

$$\dot{m}_{i,p} = \frac{A_t p_{0,3}}{\sqrt{T_{0,3}}} \sqrt{\frac{\kappa}{R}} \left(\frac{\kappa + 1}{2} \right)^{-\frac{\kappa+1}{2(\kappa-1)}}. \tag{3.3}$$

For the adopted critical diameter and inlet temperature $T_{0,3} = 300\,K$, this gives a primary mass flow rate of approximately $\dot{m}_p = 0.445\,kg/s$ at $NPR = 2.85$. Therefore, a secondary injection level of $MFR = 3\%$ corresponds to $\dot{m}_s = 0.014\,kg/s$ and the same temperature.

Boundary condition set, used in the Fluent model, is summarized in Table 3.1. Pressures were introduced in Fluent as gauge values in bar, consistently with the setup shown in the solver interface.

Table 3.1: CFD boundary conditions for the DTN simulation.

Boundary	Type	Value
Primary inlet	Pressure inlet	$1.85\,barG$
Primary inlet	Supersonic/initial gauge pressure	$1\,barG$
Primary inlet	Total temperature	$300\,K$
Primary inlet	Flow direction	normal to boundary
Primary inlet	Turbulence intensity	5%
Primary inlet	Turbulent viscosity ratio	10
Secondary inlet	Pressure inlet	$4.35\,barG$
Secondary inlet	Total temperature	$300\,K$
Outlet/far field	Pressure outlet	$0\,barG$
Outlet/far field	Backflow total temperature	$300\,K$
Outlet/far field	Backflow direction	normal to boundary
Outlet/far field	Backflow turbulence intensity	5%
Outlet/far field	Backflow turbulent viscosity ratio	10
Symmetry planes	Type	symmetry
Solid surfaces	Type	no-slip wall

The inactive secondary slot, introduced to preserve geometrical symmetry of the model, was not used as an injection boundary in the vectoring case.

3.3 Model and meshing

The computational domain was prepared as a half model using a vertical symmetry plane. This reduced the numerical cost while preserving the relevant axisymmetric flow features and the symmetry of the thrust vectoring plane. The external far-field region was extended sufficiently far from the nozzle exit to limit the influence of the outlet boundary on the internal nozzle flow. The adopted far-field length was $14 \cdot d_e = 420\,mm$, while the far-field diameter was $13 \cdot d_e = 390\,mm$. The domain dimensions are shown in Figure 3.2.

The mesh was generated as a quasi-structured grid and then converted to a polyhedral mesh for the Fluent calculation. The near wall region was refined to resolve the boundary layer with the first cell height selected for approximately $y^+ = 1$, which resulted in a first layer thickness of $1.51772 \cdot 10^{-6}\,m$. Additional refinement was introduced in the throat regions, in the DTN cavity, near the secondary injection slot, and downstream of the nozzle exit, where high pressure and velocity gradients were expected. Rounded edges and local O-grid topology were used in the nozzle region to improve cell quality near sharp geometrical features [49].

The final Fluent mesh report listed 7 742 480 nodes and 4 398 204 mixed interior faces. The mesh contained separate zones for the primary inlet, secondary injection slot, outlet, symmetry boundaries, nozzle walls, collector walls, orifice walls, and internal fluid regions. Non-overlapping interfaces were used where the slot and neighbouring fluid domains were connected. The complete mesh is shown in Figure 3.3, while the local refinement in the nozzle region is shown in Figure 3.4.

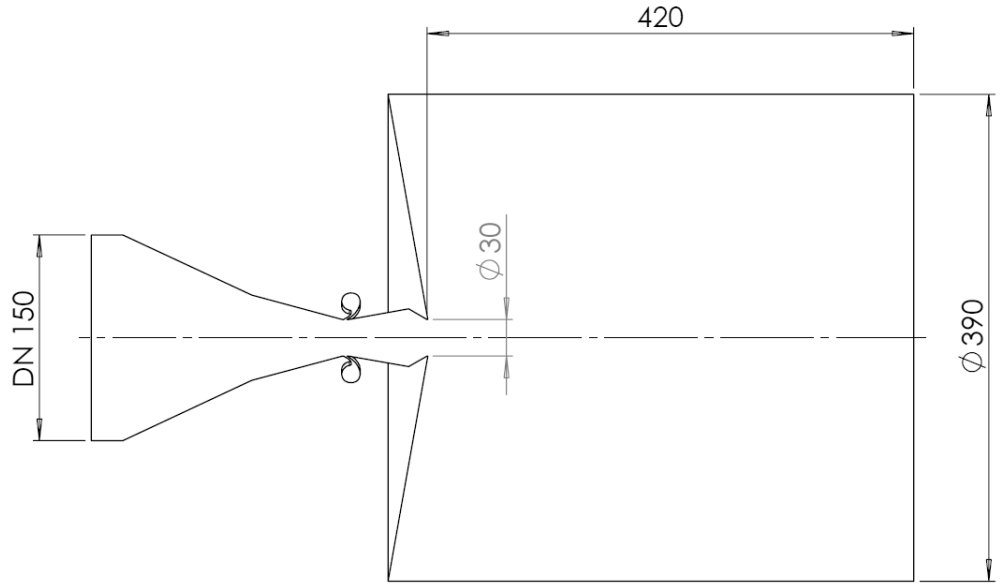


FIGURE 3.2: Flow domain and far field dimensions

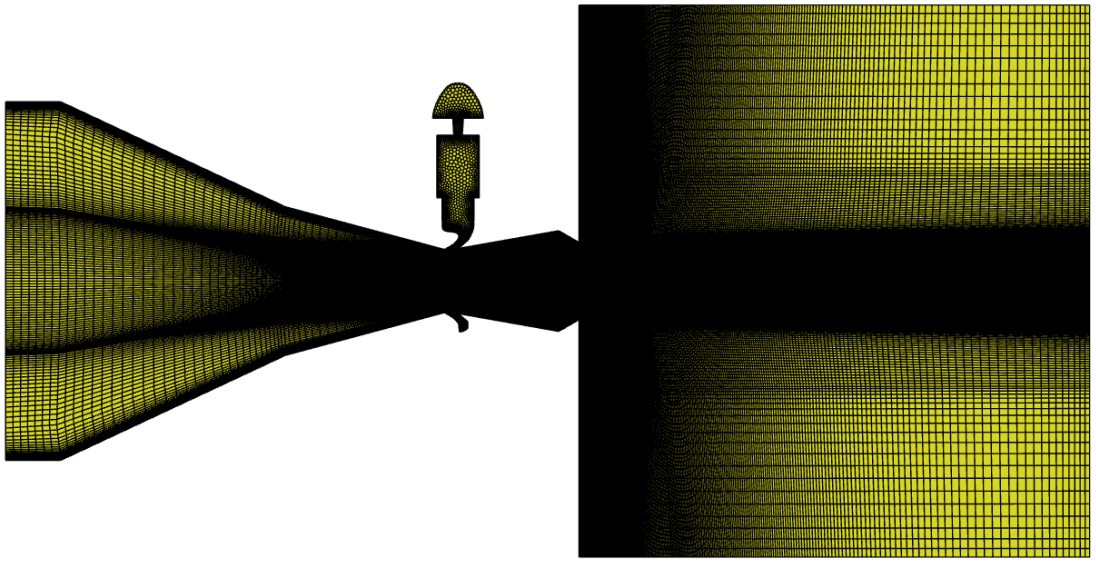


FIGURE 3.3: Mesh

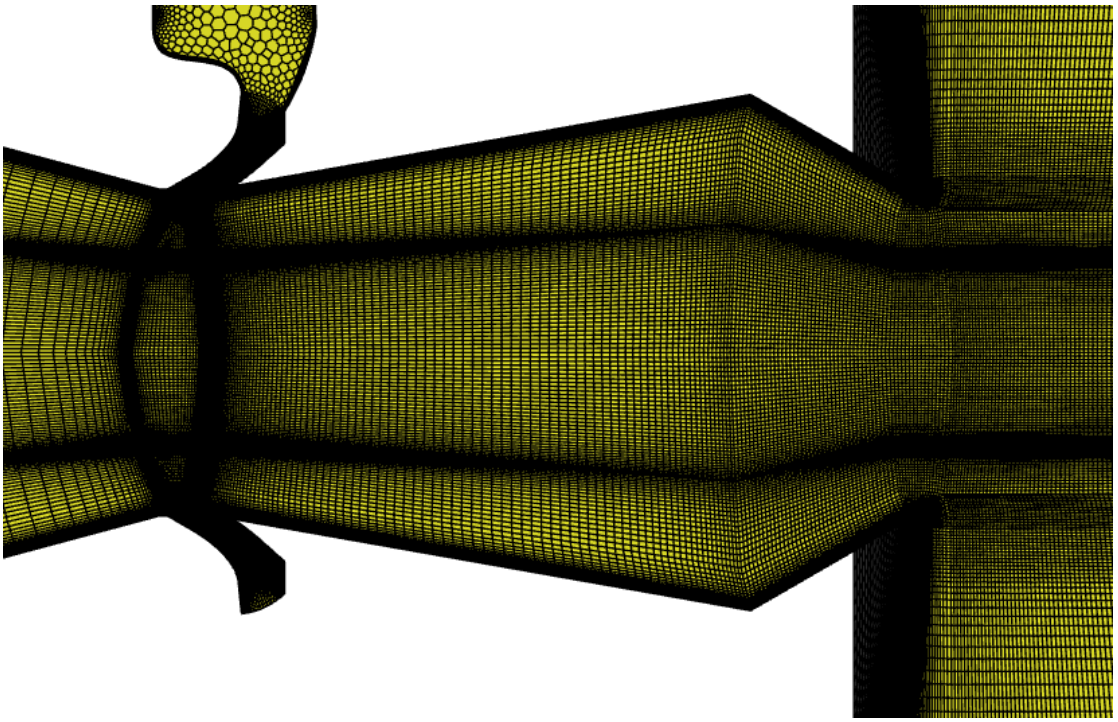


FIGURE 3.4: Mesh – close view of the nozzle region

3.4 Physical and numerical setup

The compressible Reynolds-averaged Navier-Stokes (RANS) equations [99] were solved using ANSYS Fluent [3]. A pressure based transient solver with absolute velocity formulation was used. The energy equation was enabled and air density was calculated from the ideal gas law. The adopted material properties are listed in Table 3.2.

TABLE 3.2: Air properties used in the Fluent model.

Property	Value
Density model	ideal gas
Specific heat, c_p	$1006.43\,J/(kg\,K)$
Thermal conductivity	$0.0242\,W/(m\,K)$
Dynamic viscosity	$1.7894 \cdot 10^{-5}\,kg/(m\,s)$
Molecular weight	$28.966\,kg/kmol$
Specific gas constant	$287\,J/(kg\,K)$
Specific heat ratio	$\kappa \approx 1.4$

Turbulence was modeled with the two-equation $k-\omega$ SST model [60]. This model was selected because it is commonly used for wall-bounded transonic flows with adverse pressure gradients, shock boundary layer interaction, and possible separation. The production limiter was enabled, while low-Re corrections, curvature correction, Kato-Launder [47]

production correction, and transition modeling were disabled. The near wall treatment was set to the correlation option. The model constants were left at the Fluent default values, as summarized in Table 3.3.

TABLE 3.3: SST k-ω turbulence model settings.

Setting	Value
Turbulence model	SST k-ω
Near wall treatment	correlation
Production limiter	enabled
Low-Re corrections	disabled
Curvature correction	disabled
Kato-Launder production correction	disabled
Transition model	none
β_i inner	0.075
β_i outer	0.0828
β^*_∞	0.09
α_∞	0.52
a_1	0.31

Second-order spatial discretization was used for the transported flow variables. The transient calculation was initialized from the expected inlet and outlet pressure levels and subsequently advanced in time until the monitored pressure and mass flow histories reached a repeatable state. Residuals and integral quantities, including inlet and outlet mass flow rates and wall pressure signals, were monitored during the calculation to assess numerical stability and convergence within each time step.

3.5 Results

3.5.1 Resultant Force and Thrust Vector Estimation

The resultant force generated by the nozzle was estimated from the CFD solution using a dedicated Python post-processing script listed in Appendix B. The script reads the local pressure, density, velocity components and cell volume values exported from Fluent. The local cell volume was converted into an equivalent outlet section area using the exported layer thickness of 0.58 mm.

The force was calculated as the sum of the pressure contribution and the momentum flux contribution

$$\mathbf{F} = \int_A p\mathbf{n}\, dA + \int_A \rho(\mathbf{V}\cdot\mathbf{n})\mathbf{V}\, dA. \tag{3.4}$$

The calculation was performed for a half-domain model with symmetry in the xy plane. Therefore, the force components in the x and y directions were doubled to represent the complete geometry. The out-of-plane component cancels by definition after mirroring and was therefore not included in the final resultant force.

The outlet section contained 12 767 cells. The effective area was 1.79×10^{-4} m^2 for the half-domain and 3.59×10^{-4} m^2 for the complete domain. For the half-domain, the axial pressure contribution was 7.71 N, while the corresponding axial momentum contribution was 25.07 N. The half-domain total force components were

$$F_{x,1/2} = 32.77 \text{ N}, \qquad F_{y,1/2} = 9.37 \text{ N}. \tag{3.5}$$

After applying the symmetry factor, the force components for the complete nozzle were

$$F_x = 65.54 \text{ N}, \qquad F_y = 18.73 \text{ N}. \tag{3.6}$$

The resultant force magnitude was therefore

$$F_r = \sqrt{F_x^2 + F_y^2} = 68.17 \text{ N}. \tag{3.7}$$

The thrust vector angle in the symmetry plane was calculated as

$$\delta_{xy} = \delta_p = \tan^{-1}\left(\frac{F_y}{F_x}\right), \tag{3.8}$$

which gave

$$\delta_p = 15.95^\circ. \tag{3.9}$$

This confirms that the flow field obtained in the CFD simulation produces a significant lateral force component and a clear deflection of the resultant thrust vector. This result is also consistent with the literature [32], where the angle was calculated to be approximately $\delta_p \approx 15^\circ$ (Figure 3.5).

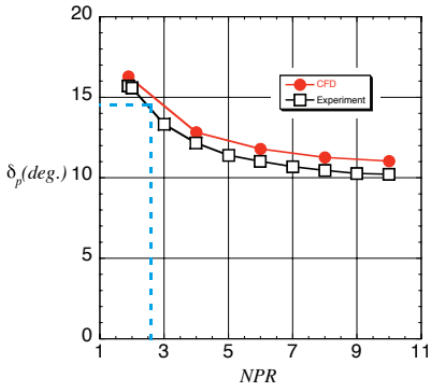


FIGURE 3.5: DTN thrust vector angle δ_p over range of NPR with MFR=3%. Open symbols experimental data. Solid symbols computational data. [32]

The thrust vectoring efficiency was also estimated using the resultant vectoring angle obtained from the CFD force balance. For the calculated angle of $\delta_p = 15.95°$ and the MFR of $MFR = 3.1\%$, the vectoring efficiency was

$$\eta = \frac{\delta_p}{MFR} = \frac{15.95}{3.1} = 5.15\ °/\%. \tag{3.10}$$

This means that, under the flow conditions analyzed, each one percent of the secondary mass flow generated approximately 5.15° thrust vector deflection.

3.5.2 Wall pressure distribution

Figure 3.6 presents the calculated mean wall pressure distribution obtained from the CFD solution. Calculations were performed using Python code.C. The pressure values were averaged over a selected quasi-steady time window and normalized by the reference pressure p_3. Separate distributions are shown for the upper and lower nozzle walls, while the nozzle contour is included as a geometrical reference to indicate the location of individual pressure points along the divergent section. This representation provides a compact description of the spatial pressure loading predicted by the numerical model and will be used later for direct comparison with the experimentally measured wall pressure distribution.

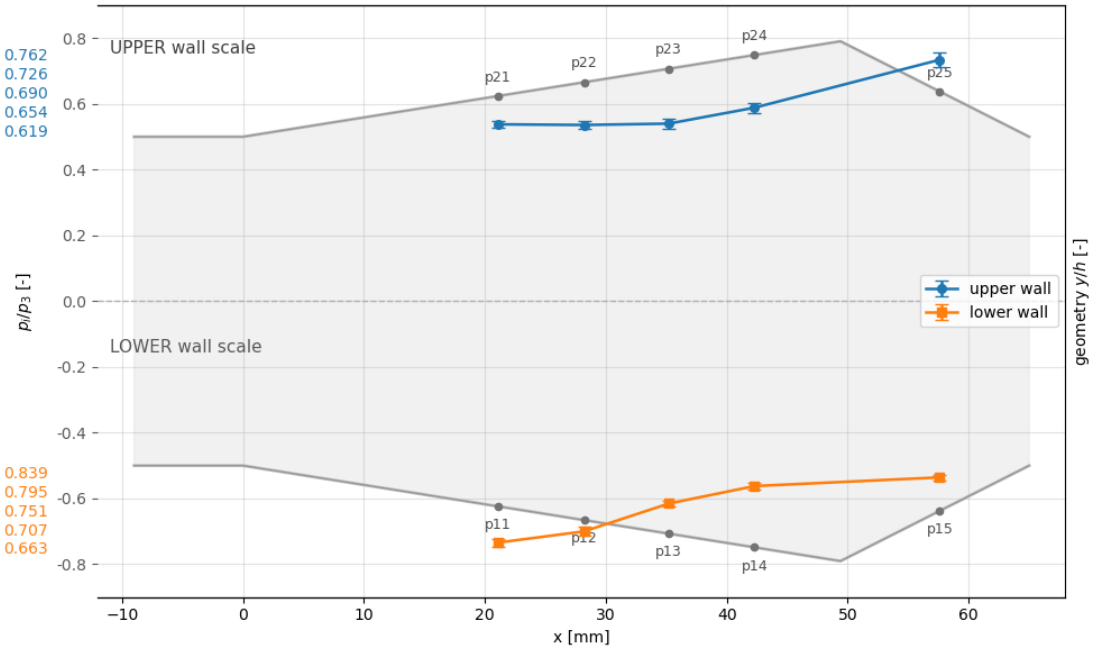


FIGURE 3.6: Calculated mean surface pressure distribution.

3.5.3 Unsteady vectorized flow and spectral analysis

The vectorized operating mode was accompanied by clearly visible unsteady flow structures in the nozzle cavity and in the external jet. To quantify this behavior, the wall pressure signals obtained from the CFD solution were analyzed in the frequency domain using the Fast Fourier Transform (FFT) in python code C. The FFT is an efficient algorithm for computing the discrete Fourier transform of a sampled signal, introduced in its modern computational form by Cooley and Tukey [14]. For a discrete pressure signal p_n sampled at constant time intervals, the transform may be written as

$$P_k = \sum_{n=0}^{N-1} p_n \exp\left(-i\frac{2\pi kn}{N}\right), \tag{3.11}$$

where P_k is the complex Fourier coefficient corresponding to the k-th frequency component, and N is the number of samples. In the present analysis, the mean value was removed from each signal before applying the FFT, so that the obtained spectra describe only the fluctuating component of the wall pressure.

Figure 3.7 shows the FFT amplitudes of the wall pressure signals for the vectorized case in the time window of $t = 0.004$ s. The spectra indicate that the pressure field is not steady during this interval. The amplitude occurs at low frequencies, with a dominant contribution close to the first resolved frequency range of the analyzed window. Additional weaker components are visible at higher frequencies, but their amplitudes decrease rapidly. Due to the relatively short duration of the analysis, the frequency resolution is

limited by the window length. Therefore, the FFT should be interpreted mainly as an indication of the presence and relative intensity of pressure pulsations, rather than as a precise identification of a single characteristic frequency.

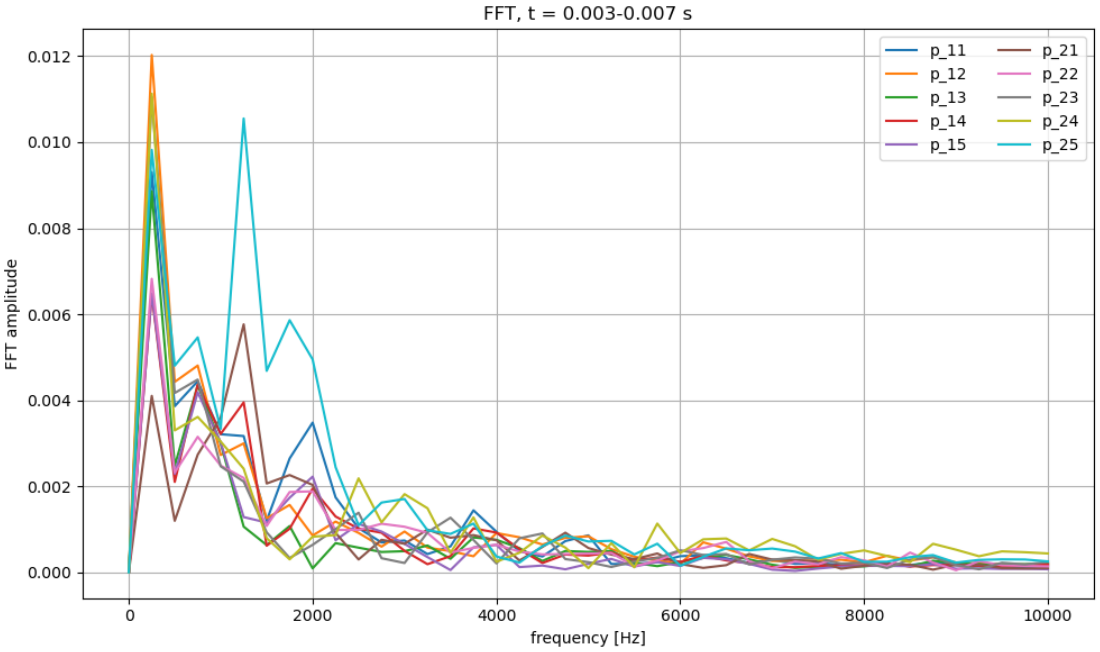


FIGURE 3.7: FFT amplitudes of the wall-pressure signals obtained for the vectorized CFD case.

The instantaneous Mach number fields in Figures 3.8 and 3.9 provide a physical interpretation of the spectral results. Before the fully developed thrust vectoring state, shown in Figure 3.8, the jet remains only weakly deflected and the internal flow structure changes between consecutive time instants. This phenomenon has been noted previously in the literature [32] and is undesirable. Alternating regions of higher and lower Mach number appear downstream of the second throat, indicating oscillatory behavior of the shear layer and shock related structures. After the onset of vectoring, Figure 3.9 shows a more pronounced asymmetric jet deflection. The high Mach number core is displaced toward one side of the nozzle, while the separated region inside the cavity evolves in time. These time dependent changes in the separated flow, shear layer and external jet explain the pressure fluctuations observed at the wall pressure locations and confirm that the vectorized DTN flow is inherently unsteady in the analyzed operating condition.

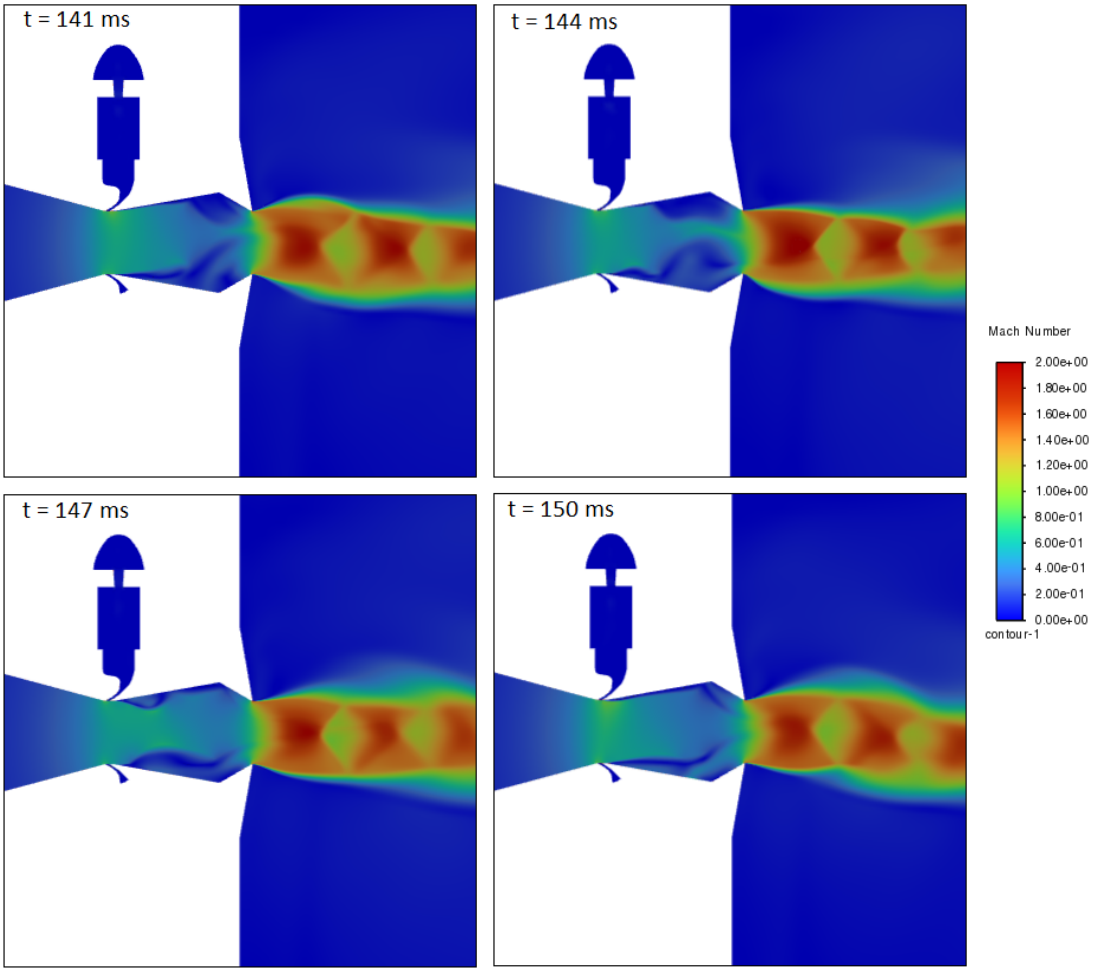


FIGURE 3.8: Instantaneous Mach number fields before the fully developed vectoring state.

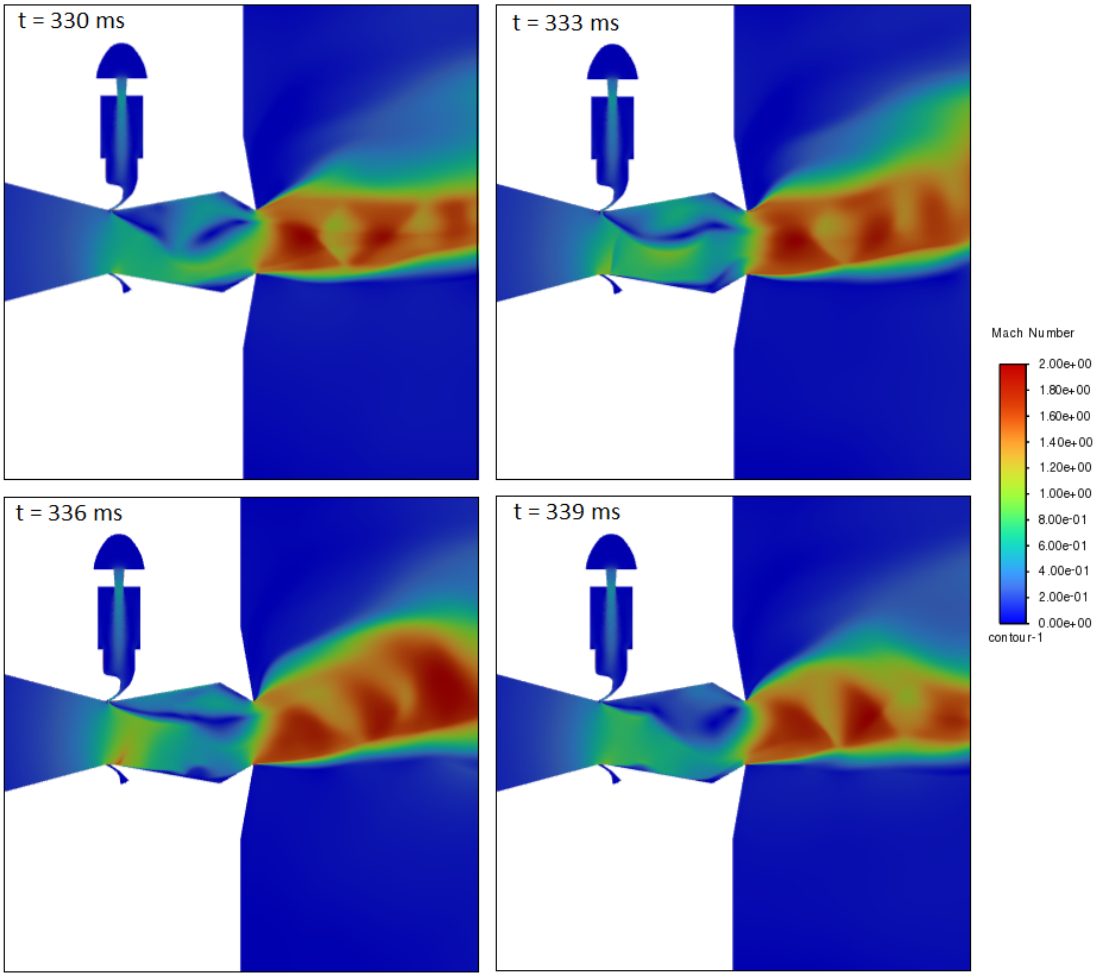


FIGURE 3.9: Instantaneous Mach number fields after the onset of the vectorized flow state.

Chapter 4

Experimental setup & instrumentation

4.1 Operating Principle and Design of the Transonic Ludwieg tube facility

4.1.1 Operating Principle

The experimental campaign was carried out in a short duration transonic Ludwieg tube facility designed to deliver compressed air to the investigated DTN model. The Ludwieg tube concept was originally introduced by Hubert Ludwieg [56] and relies on the use of a long pressurized tube as a finite duration source of stagnation conditions. In contrast to a continuous wind tunnel, a Ludwieg tube does not maintain the flow indefinitely. Instead, it provides a short and repeatable interval of quasi steady compressible flow.

Before each run, the driver tube is filled with compressed air to the prescribed initial pressure. After the fast acting valve is opened, the gas expands toward the downstream section of the facility. If the imposed pressure ratio is higher than the critical pressure ratio, the flow becomes choked at the limiting throat, and the mass flow rate is governed mainly by the throat area and by the instantaneous upstream stagnation pressure and temperature. The valve opening generates an expansion wave that travels upstream through the driver tube. Behind this wave, the pressure and temperature decrease, but the gas properties remain approximately uniform for a finite time interval. The useful test time is therefore limited to the period between the establishment of quasi steady flow and the return of the reflected expansion wave from the closed end of the driver tube. After this reflected wave reaches the nozzle, the upstream supply conditions change and the flow can no longer be treated as stationary [79]. This operating principle of the

facility is illustrated schematically in Fig. 4.1, where graph shows traveling expansion wave. This operating mode is suitable for the present study because the characteristic dynamic response times of the pressure field in the DTN are shorter than the available quasi steady flow interval.

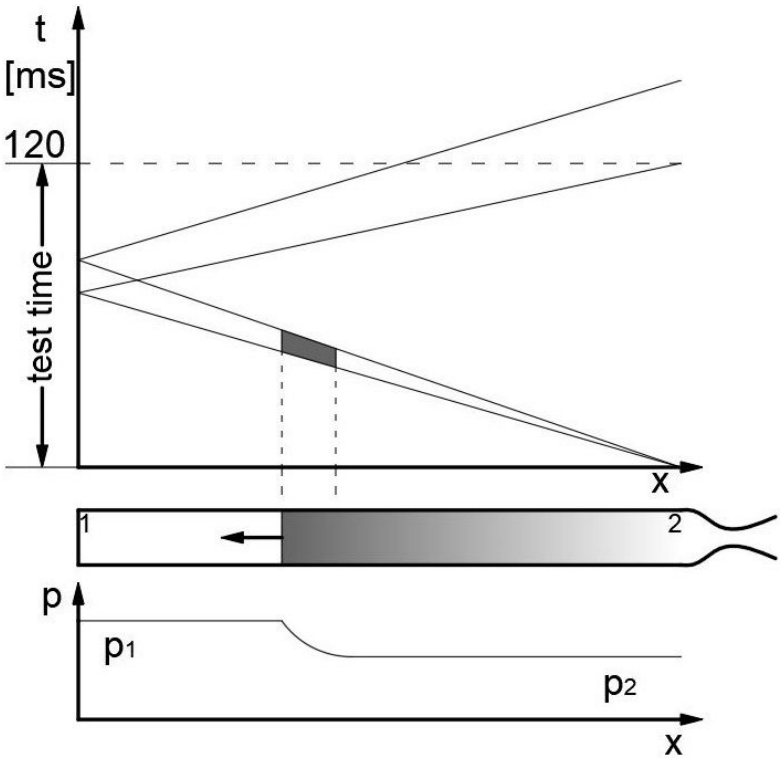


FIGURE 4.1: Scheme of the expansion process in the Ludwieg tube [79].

4.1.2 Facility Design and Construction

The Ludwieg tube used in this study was developed as a modular laboratory facility for investigations of compressible internal flows, nozzle flows and transient pressure phenomena. The design assumptions and the initial construction concept were previously described in [44]. This Ludwieg tube is shown in Figure 4.2.

The main storage volume of the facility is formed by DN150 pipe segments connected into a driver tube. In the present configuration, the tube is folded by 180° using two Hamburg elbows. This arrangement reduces the required laboratory floor length while preserving the effective length of the driver section. Similar folded or double bent storage tube configurations are reported for other Ludwieg tube facilities [102]. Therefore, the use of bends was treated as an acceptable construction solution. The actual useful measurement interval was determined from the recorded pressure histories rather than assumed only from the nominal tube length.

Prior to each experiment, the driver tube was charged with compressed air to the specified initial pressure. The air temperature was then allowed to approach the ambient temperature in order to improve repeatability of the initial stagnation conditions. The operating conditions used in the present study were selected according to the requirements of the DTN test section and the available quasi steady measurement time.



FIGURE 4.2: Ludwieg tube at the Aviation Engines Laboratory of the Poznan University of Technology.

4.1.3 Fast Acting Valve

In the present facility, the role of the fast acting valve is performed by a notched diaphragm made of Biaxially Oriented PET foil (BoPET) with a thickness of 0.35 mm. Before each run, the diaphragm separates the pressurized driver tube from the downstream section of the apparatus. The diaphragm rupture is initiated mechanically by a razor blade mounted on a pneumatic actuator, as shown in Fig. 4.4. After rupture, the diaphragm opens rapidly and produces a repeatable start of the flow.

A ruptured diaphragm is shown in Fig. 4.3. To reduce uncontrolled tearing and shearing of diaphragm fragments against the pipe flange, a machined aluminium insert with a radius of 10 mm was installed at the diaphragm seat. The insert provides a rounded support edge and reduces the risk of fragment retention or partial obstruction of the flow path after opening.



FIGURE 4.3: Ruptured BoPET diaphragm after a test run.

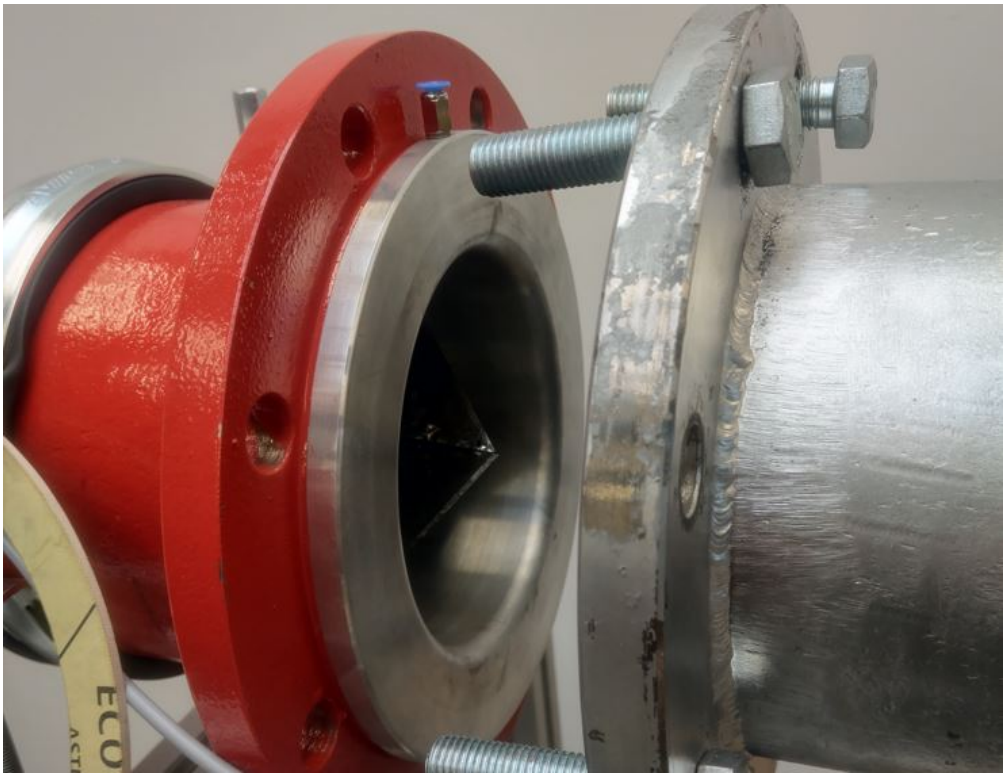


FIGURE 4.4: Blade used to initiate diaphragm rupture.

4.2 Manufacturing and Modular Design of the DTN Model

The test section contained the investigated DTN model and the pressure measurement channels integrated into the nozzle walls. The nozzle was manufactured as a modular assembly in order to allow modifications of the secondary injection geometry and to provide access to the internal pressure taps. The complete test section is shown in Figure 4.5.

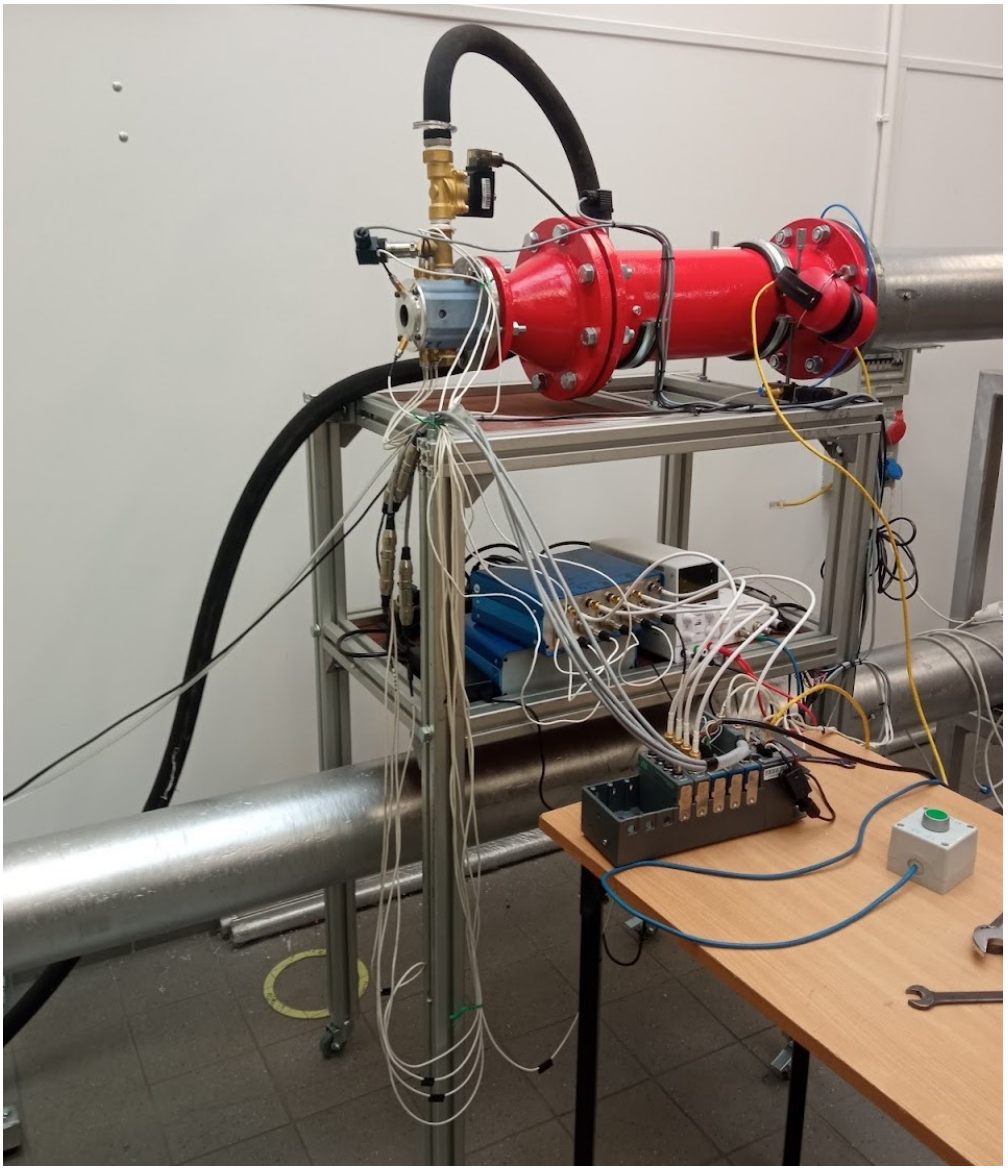


FIGURE 4.5: DTN test section with Ludwieg tube Facility

The converging and diverging parts of the primary nozzle were manufactured as two separate elements using stereolithography. A low shrinkage photopolymer resin was selected to reduce dimensional deviations after curing. The use of additive manufacturing made it possible to integrate internal pressure measurement channels directly into the

nozzle walls, which would have been difficult to manufacture using conventional machining. The downstream part of the divergent section was equipped with wall pressure taps used to measure the pressure distribution along the DTN contour.

The primary throat region and the secondary injection slot were designed as an interchangeable insert. Initially, inserts with different slot widths were planned to be used to control the mass flow rate of the secondary jet. However, this method was later replaced by axisymmetric orifice plates installed downstream of the secondary flow valve. This solution allowed the secondary mass flow rate to be adjusted independently of the nozzle wall geometry, while keeping the injection slot geometry unchanged during the tests.

The secondary converging section and the secondary throat were manufactured from aluminium. These components also included pressure measurement holes, allowing the pressure upstream of the secondary throat and in the injection path to be monitored. The combination of a stereolithography printed primary nozzle and machined aluminium secondary flow components provided both geometric flexibility and sufficient mechanical stiffness in the regions subjected to direct secondary flow loading.

The section view of 3D CAD model of the test section is shown in Fig. 4.6.

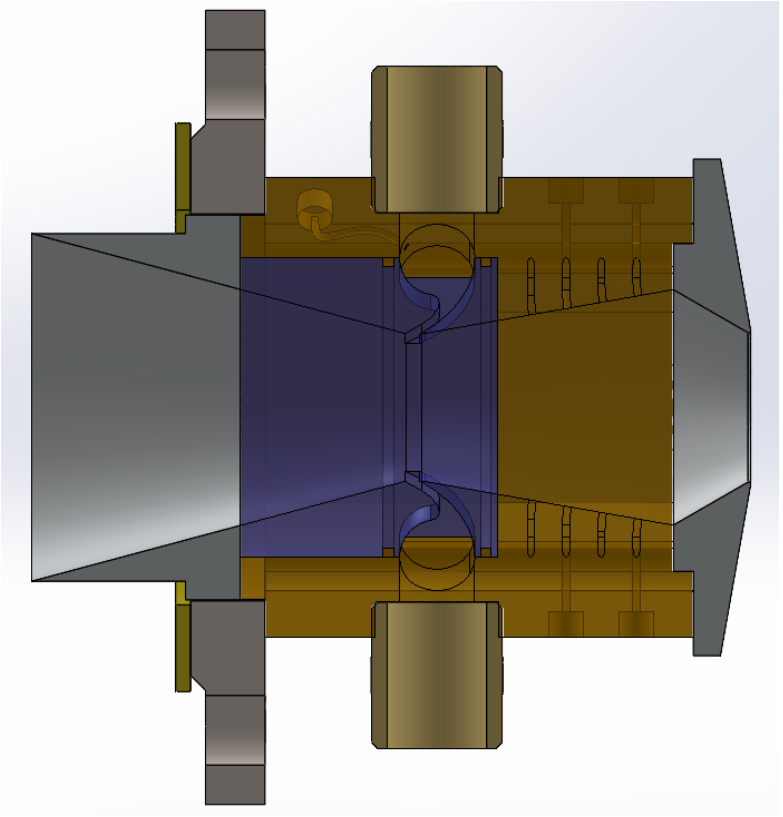


FIGURE 4.6: Overview of the DTN modular test section

4.3 Mass Flow Control in the Primary and Secondary Lines

Primary and secondary mass flow rates were controlled by separate flow paths. The primary flow $\dot{m}_p$ was supplied from the driver section of the Ludwieg tube, whereas the secondary flow $\dot{m}_s$ was supplied through an independent injection line equipped with a solenoid valve and interchangeable orifice plates.

The primary mass flow rate was governed by the stagnation pressure upstream of the nozzle, denoted as p_3. After the diaphragm rupture, this pressure stabilize at level in order of 0.2 bar lower than in the driver section of the Ludwieg Tube p_1. The mass flow limiting element in the primary line was the fixed critical cross section of the primary DTN throat, with a throat diameter of $D_t = 30$ mm. After opening of the fast acting valve, the gas accelerated through the primary nozzle. When the pressure ratio exceeded the critical value, the flow became choked in the throat and the primary mass flow rate was determined by the throat area, the upstream stagnation pressure and temperature and the gas properties.

The secondary mass flow rate was independently regulated by an orifice plate installed in the secondary supply line. The flow was initiated by a solenoid valve, while the interchangeable orifice plate determined the limiting cross section of the secondary line. This configuration allowed the secondary mass flow rate to be adjusted without modifying the geometry of the injection slot in the DTN model.

The orifice plates were manufactured from brass as flat, polished inserts. The holes were finished using a tapered reamer, which resulted in a 7.15° converging inlet geometry rather than a sharp edged cylindrical orifice. The cross section of the resulting orifice plate is shown in Fig. 4.7. For this type of inlet geometry, the discharge coefficient was adopted as $C_d = 0.925$, based on literature data for rounded and converging orifice geometries [22]. This value was used in the calculation of the secondary mass flow rate for all tested orifice plates. The graph of orifice discharge coefficient versus the angle of convergence is shown in Figure 4.8.

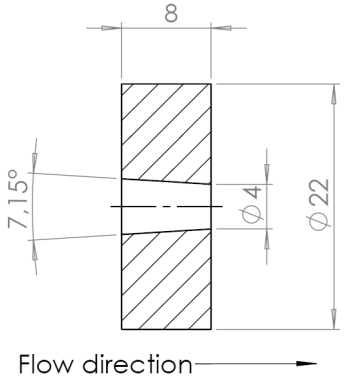


FIGURE 4.7: Orifice plate cross section

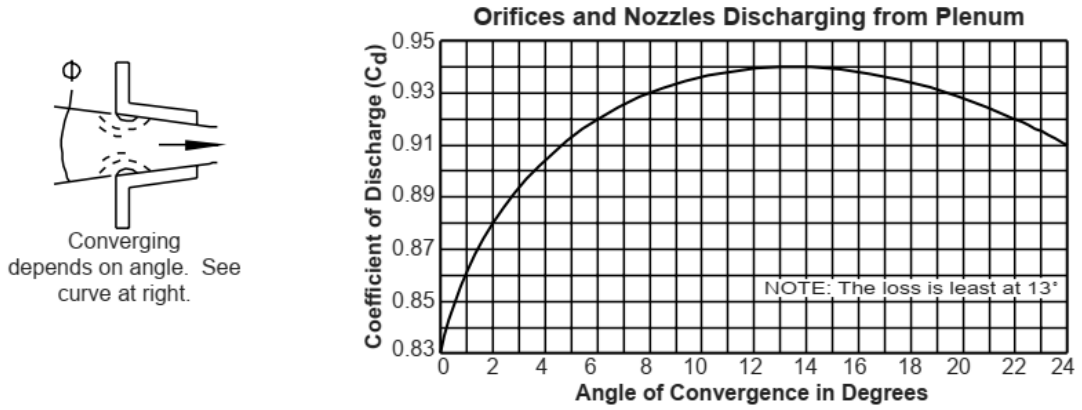


FIGURE 4.8: Discharge coefficients for different shapes of orifice plates [22].

4.4 Measurement System and Instrumentation

4.4.1 Selection of Pressure Transducers

The experimental analysis was based primarily on pressure measurements. The recorded quantities included the wall pressure distribution inside the DTN, the pressure upstream of the primary nozzle, and the pressures required to determine the primary and secondary mass flow rates. Direct thrust measurements were not performed, because the test section was rigidly connected to the Ludwieg tube facility and was not mounted on a force balance.

The investigated flow contained both transient and pulsating pressure components. Therefore, the selection of pressure transducers was governed not only by the required pressure range, but also by the dynamic response of the sensor. Measurements of rapid compressible flow phenomena, including shock wave motion and pressure oscillations, require transducers with sufficiently high diaphragm natural frequency. This requirement is typically met by piezoelectric or piezoresistive transducers with silicon diaphragms. Piezoelectric sensors are suitable for detecting rapid pressure changes, but they are less appropriate when the absolute pressure level must be measured. For this reason, piezoresistive transducers operating in a Wheatstone bridge configuration were used for the main pressure measurements [46]. A comparison between piezoelectric and piezoresistive pressure signals recorded in a transient flow experiment is shown in Fig. 4.9.

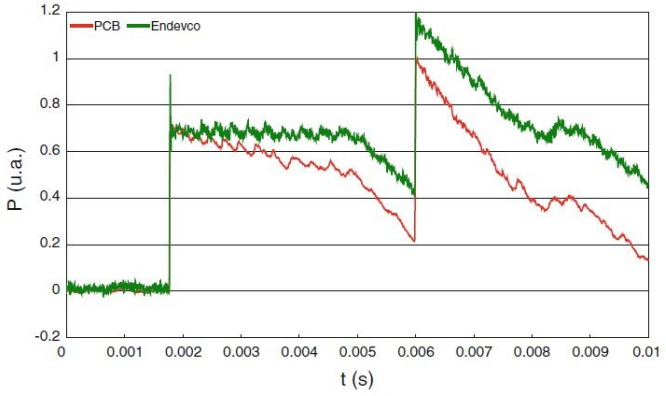


FIGURE 4.9: Comparison of signals recorded by PCB piezoelectric and ENDEVCO piezoresistive transducers installed face to face during the same run [76].

The wall pressure measurements were performed using MEAS XP5, MEAS XPM4, and Kulite XTL-190 miniature pressure transducers available in the laboratory. Their main parameters are listed in Table 4.1. The XP5 and XTL-190 transducers were used as the primary sensors for dynamic wall pressure measurements. The XPM4 transducer had a lower full scale pressure range and, in the applied measurement chain, provided a lower useful output level. Consequently, this channel required higher signal amplification, which increased the contribution of electronic noise to the recorded signal. For this reason, the XPM4 signal was treated with particular caution in the quantitative interpretation of the absolute pressure level, while it remained useful for identifying the timing and qualitative character of the pressure response.

TABLE 4.1: Transducers parameters.

Transducer model	Full scale	Natural Frequency	Full Scale Output
XPM4 [88]	2 bar G	249 kHz	30 mV
XP5 [89]	5 bar A	90 kHz	100 mV
XTL-190 [50]	7 bar A	380 kHz	100 mV

The useful frequency range of a pressure transducer is commonly estimated as approximately 20% of its natural resonant frequency. For the transducers used in this study, this gives a usable frequency range of at least 18 kHz, corresponding to the XP5 sensor with a natural frequency of 90 kHz [89]. The XPM4 and XTL-190 transducers have higher natural frequencies and therefore provide an additional dynamic response margin. Thus, the dynamic response of the selected transducers was sufficient for the time scales investigated in this work.

Industrial grade Belimo 22WP-115 and 22WP-116 pressure transducers were used for auxiliary and control measurements. These sensors measured the pressures associated with filling the Ludwieg tube and with monitoring the secondary flow supply system.

Since these quantities varied more slowly than the wall pressure signals, high frequency miniature transducers were not required for these measurement locations.

Before installation in the experimental facility, each pressure transducer was checked against a *WIKA CPT6200* reference pressure transducer [98]. This procedure was used to determine the calibration coefficients used during data reduction.

4.4.2 Location of Measurement Points

The measurement points were divided into four functional groups: ambient parameters, Ludwieg tube supply parameters, secondary flow parameters, and DTN wall pressure values. This division reflects the role of each signal in the experiment: control of the initial conditions, calculation of mass flow rates, and analysis of the pressure response inside the nozzle.

The ambient parameters were recorded in order to define the reference conditions for each run. These values were used in the calculation of air properties and in the conversion of gauge pressure readings to absolute pressure where required.

The Ludwieg tube supply parameters included the pressure in the driver tube upstream of the diaphragm and the air temperature measured near both ends of the tube. The driver tube pressure was used to control the filling process before each run. The temperature measurements were used to verify that the compressed air in the driver tube had returned to thermal equilibrium with the surroundings before the next shot. This procedure improved the repeatability of the initial stagnation conditions between individual runs.

The secondary flow parameters were measured in the injection line. Pressure measurements upstream and downstream of the interchangeable orifice plate were used to determine the secondary mass flow rate. The secondary flow pressure signal also provided the reference timing for the onset of injection, which was later used in the dynamic response analysis.

The main measurement set consisted of pressure taps located in the DTN model and a pressure measurement upstream of the primary nozzle. The upstream pressure p_3 was used to determine the NPR and the primary mass flow $\dot{m}_p$. The wall pressure taps were distributed along the lower and upper nozzle walls in order to capture the pressure redistribution caused by secondary injection. On the lower wall, the measurement points were located at p_{11}, p_{12}, p_{13}, p_{14}, and p_{15}. On the upper wall, the corresponding pressure measurements were taken at p_{22}, p_{24}, and p_{25}. The positions were selected to provide coverage of the cavity and downstream throat region, while respecting the geometric constraints imposed by the nozzle wall thickness, internal pressure channels, and available space for tubing and transducer connections.

All measurement points are shown in Fig. 4.10. The measured quantities, together with the corresponding transducer type and acquisition interface, are listed in Table 4.2.

TABLE 4.2: Measurement chain parameters.

name	**description**	**sensor/actuator**	**type**	**interface**	
T_0	Ambient temperature	Mercury termometer	-	manual	-
p_0	Atmospheric pressure	Aneroid barometer	-	manual	-
T_1	Temperature 1	thermocouple	type-K	1314	T1
T_2	Temperature 2	thermocouple	type-K		T2
p_7	Pressure in $\dot{m}_s$ manifold	22WP-116	G	NI 9234	CH0
p_1	Ludwieg tube pressure	22WP-115	G		CH1
p_3	Pressure before nozzle	2WP-115	G		CH2
p_6	Pressure before orifice	22WP-116	G		CH3
p_{11}	Pressure x=21.1, (lower wall)	XPM4 II	A	NI 9237	CH0
p_{12}	Pressure x=28.2, (lower wall)	XP5	G		CH1
p_{13}	Pressure x=35.2, (lower wall)	XP5	G		CH2
p_{14}	Pressure x=42.3, (lower wall)	XP5	G		CH3
p_{22}	Pressure x=21.1, (upper wall)	XTL-190	A	NI 9234	CH0
p_{24}	Pressure x=42.3, (upper wall)	XTL-190	A		CH1
p_{25}	Pressure x=57.6, (upper wall)	XTL-190	A		CH2
p_{15}	Pressure x=57.6, (lower wall)	XTL-190	A		CH3
S_1	Membrane bursting actuator	pneumatic	-	NI 9482	CH1
S_2	Secondary flow valve	AB510	-		CH2
S_0	Filling valve	AB510	-	manual	-
S_4	Dumping valve	ball valve	-	manual	-

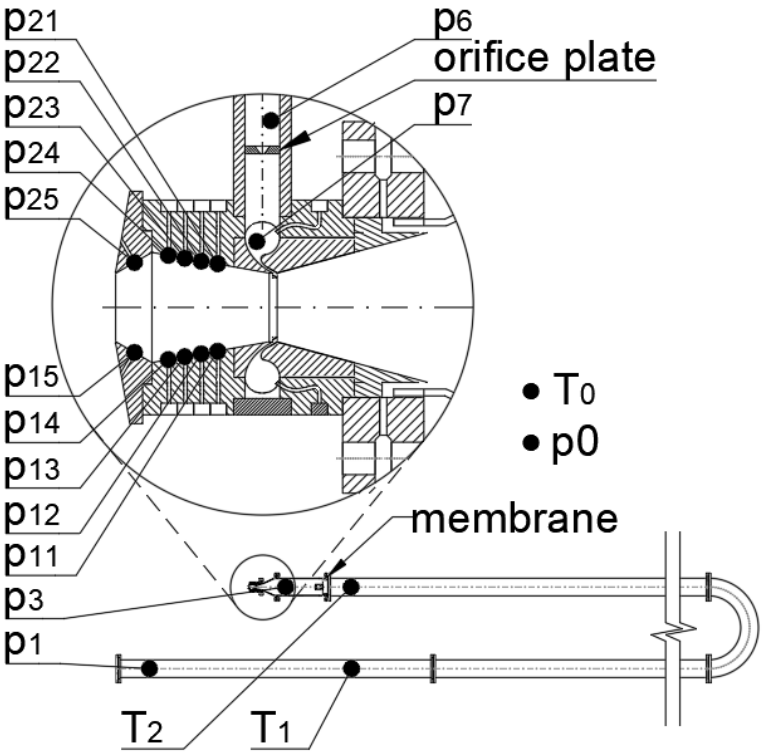


FIGURE 4.10: Measurement points

4.4.3 Dynamic Response of Pressure Tap Channels

From the standpoint of dynamic pressure measurement, the most favorable solution would be to mount each pressure transducer directly flush with the nozzle wall. In the present geometry, however, this approach was not practical. First, the active diaphragm diameter of the miniature transducers was of the order of several millimeters, which would result in spatial averaging over a wall region rather than a local pressure measurement. Second, the available space in the DTN model was insufficient to install several transducers directly in the wall at the required axial spacing. Therefore, the transducers were mounted outside the test section and connected to the wall pressure locations by pressure tap channels with a diameter of approximately 0.8 mm.

This configuration introduces additional dynamic effects into the measurement chain. The pressure signal is delayed by the finite propagation time in the channel, high frequency pressure fluctuations may be attenuated, and the channel may exhibit acoustic resonance. The first two effects were evaluated in a separate shock tube experiment. Three pressure measurement configurations were compared: a transducer mounted directly in the wall, a transducer connected by a 110 cm long tube with an inner diameter of 2 mm, and a transducer connected by a 10 cm long tube with an inner diameter of 7 mm. The shock tube arrangement is shown schematically in Fig. 4.11, and the recorded pressure responses are presented in Fig. 4.12.

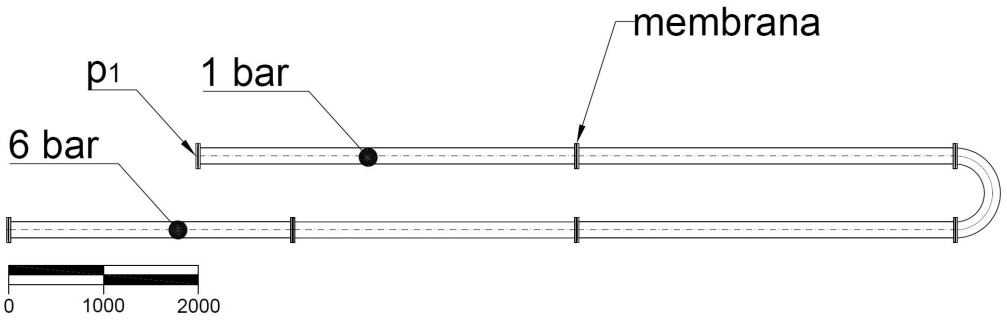


FIGURE 4.11: Shock tube setup used to evaluate the dynamic response of pressure transducer mounting configurations.

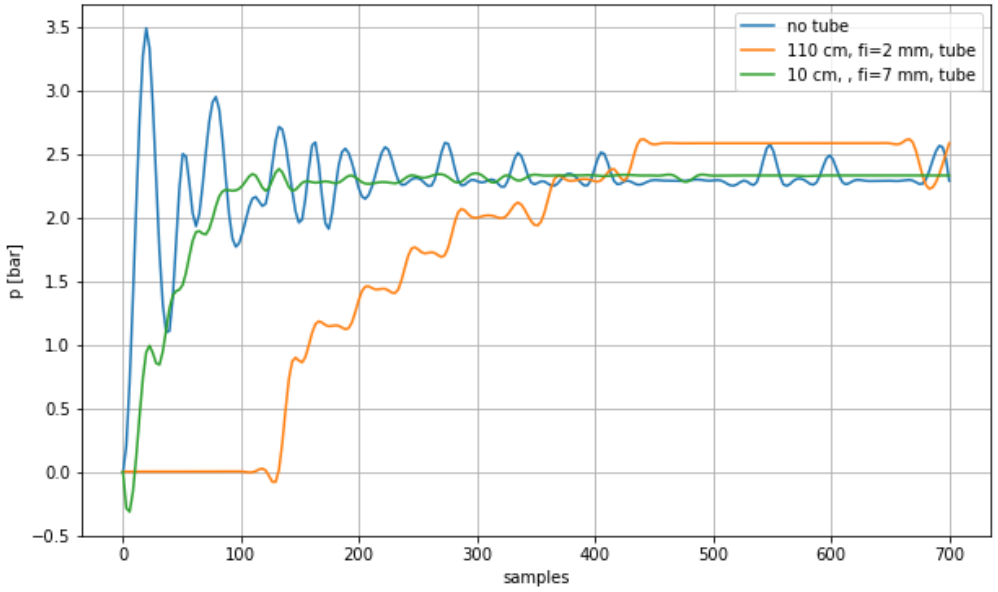


FIGURE 4.12: Influence of connecting tube length and diameter on the measured pressure build up time (100 samples correspond to 0.002 s).

The directly mounted transducer reproduced the fastest pressure rise and the pressure oscillations at the shock tube end wall. The transducer connected through the short and wide tube showed a delayed pressure increase and partial attenuation of the oscillatory component. The transducer connected through the long tube recorded a strongly damped pressure signal, with a delay of approximately 0.0032 s and a stabilization time of approximately 0.008 s.

The pressure tap channels used in the DTN test section were substantially shorter than the long tube configuration tested in the shock tube experiment. Their length was approximately 36 mm and their diameter was approximately 0.8 mm. Assuming acoustic propagation with the speed of sound in air, the estimated propagation delay is

$$\Delta t \approx \frac{L}{c}, \tag{4.1}$$

where L is the pressure tap length and c is the speed of sound in air. For the pressure taps used in the DTN model

$$\Delta t \approx \frac{0.036}{346} \approx 1.0 \times 10^{-4} \text{ s.} \tag{4.2}$$

This value is small compared with the characteristic response times investigated in the present study. Therefore, the pressure tap induced delay was considered acceptable for the analysis of the mean transient pressure response. Nevertheless, the possible attenuation and resonance of higher frequency pressure components were taken into account when interpreting the spectral content of the measured signals.

4.4.4 Quarter Wave Resonance of Pressure Tap Channels

A pressure tap channel perpendicular to the main flow may also behave as an acoustic resonator. In the present case, the channel can be approximated as a tube closed at the transducer end and open at the nozzle wall end. Therefore, the relevant acoustic mode is a quarter wave resonance rather than a Helmholtz resonance, because no distinct cavity volume is present at the closed end.

The resonance frequencies of a closed-open pressure tap channel can be estimated as

$$f_n = \frac{(2n-1)c}{4L_{\text{eff}}}, \tag{4.3}$$

where

$$L_{\text{eff}} = L + 0.3d, \tag{4.4}$$

and

$$c = \sqrt{\kappa RT}. \tag{4.5}$$

Here, L_{eff} is the effective acoustic length of the pressure tap, L is the physical channel length, d is the channel diameter, κ is the ratio of specific heats, R is the gas constant, and T is the absolute temperature.

For the pressure taps used in the DTN model, with $L = 36$ mm, $d = 0.8$ mm, and $c \approx 346$ m/s, the effective length is

$$L_{\text{eff}} = 0.036 + 0.3 \cdot 0.0008 = 0.03624 \text{ m.} \tag{4.6}$$

The first resonance frequency is therefore

$$f_1 = \frac{346}{4 \cdot 0.03624} \approx 2.39 \text{ kHz}. \tag{4.7}$$

The subsequent odd quarter wave modes are

$$f_3 = 3f_1 \approx 7.16 \text{ kHz}, \tag{4.8}$$

$$f_5 = 5f_1 \approx 11.93 \text{ kHz}, \tag{4.9}$$

and

$$f_7 = 7f_1 \approx 16.70 \text{ kHz}. \tag{4.10}$$

These frequencies define the ranges in which the pressure tap geometry may affect the measured spectral content. The first quarter wave mode, located at approximately 2.4 kHz, is the most relevant for the interpretation of the frequency domain results. Higher modes occur at odd multiples of the fundamental frequency and are expected to have lower amplitudes because of viscous, thermal, and geometric losses in the narrow channel.

As a simplified estimate, the relative amplitude of successive odd modes may be assumed to decrease according to

$$A_n \propto \frac{1}{(2n-1)^2}, \tag{4.11}$$

which gives

$$A_1 : A_3 : A_5 : A_7 \approx 1 : \frac{1}{9} : \frac{1}{25} : \frac{1}{49}. \tag{4.12}$$

Thus, spectral peaks near the calculated quarter wave frequencies should be interpreted with caution, because they may originate from the pressure tap dynamics rather than from the DTN flow field itself.

4.4.5 Signal Conditioning, Calibration and Pre-Processing

The pressure and temperature signals were acquired using a LabVIEW based measurement system and saved in TDMS (Technical Data Management Streaming) format.TDMS is a binary file format, natively developed by National Instruments, designed

to enable high-performance data writing and reading operations. Fast signal readout and processing required use functions of the FPGA (field-programmable gate array), which constitutes a class of configurable integrated circuits that can be reprogrammed multiple times following fabrication. FPGAs are a subset of digital logic components collectively known as programmable logic devices. The LabVIEW FPGA structure was responsible for deterministic acquisition of the analogue input channels, while the host VI provided run control, signal monitoring, and data saving. The LabVIEW FPGA block diagram, host block diagram, and host user interface are presented in Appendices D.

The recorded TDMS files were subsequently processed in Python. The first stage of the Python processing chain was the conversion of the LabVIEW TDMS files into raw numerical arrays. This operation was performed using the script listed in Appendix E. The script reads all TDMS files from the experiment directory, extracts the twelve recorded analogue channels, assigns them to the pressure-channel order used in the acquisition system, and saves each run as a NumPy array in the *raw_ inputs* directory. Because the individual TDMS channels may contain slightly different numbers of samples, all channels are limited to the shared minimum length prior to data storage. At this stage, the data remain in voltage units and no calibration, filtering, despiking, time shifting, or signal cutting is applied. The purpose of this step is therefore only to create a consistent raw data format for the subsequent processing stages.

The measurement chain depended on the type of pressure transducer. The MEAS XPM4 and XP5 transducers were connected directly to the *NI-9237* C-series module, which provides full bridge operation and a 10 V excitation voltage [68]. The Kulite XTL-190 transducers were connected to a custom five channel instrumentation amplifier, shown in Fig. 4.13, and then to the *NI 9234* C-series module [67]. The amplifier was based on INA-129 instrumentation amplifier chips, provided a 10 V excitation voltage, and used a gain of $G = 50$ set by precision resistors. The amplifier was battery powered in order to reduce power supply related noise in the amplified bridge signals. In the applied configuration, this measurement chain provided lower noise levels for the XTL-190 channels than the direct NI-9237 based acquisition used for the XP5 and XPM4 channels, which is visible later in the frequency domain analysis. This setup provided data acquisition at a rate of *51.2 kS/s/ch*, with 24-bit resolution.

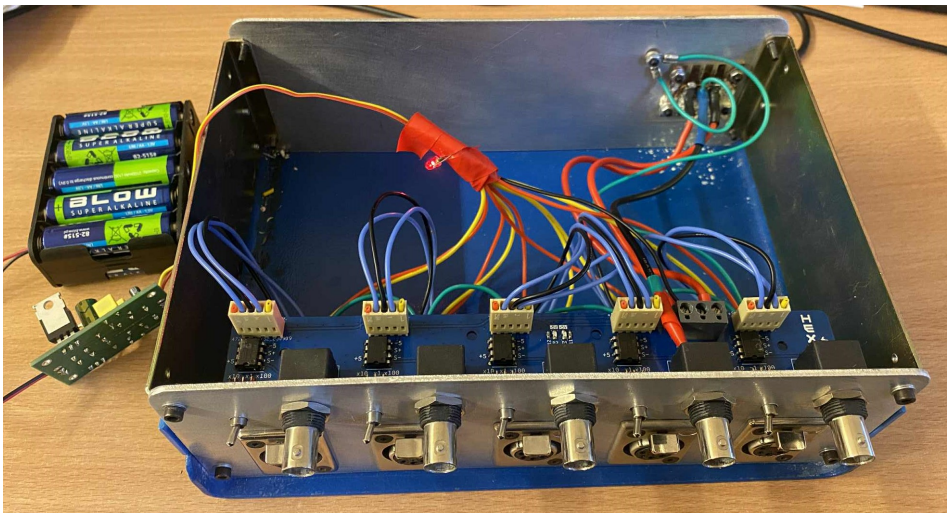


FIGURE 4.13: Custom five channel instrumentation amplifier used for the Kulite XTL-190 pressure transducers.

The purpose of data pre-processing was to convert the raw voltage signals from the pressure and temperature transducers into physical quantities required for further analysis. These quantities included the wall pressure distribution inside the DTN, the pressure upstream of the primary nozzle, the pressures required to determine the secondary mass flow rate, and the corresponding primary and secondary mass flow rates, $\dot{m}_p$ and $\dot{m}_s$.

Before the experimental campaign, calibration measurements were performed one again for the complete measurement chain. During calibration, the nozzle outlet was closed and the facility was filled to successive static pressure levels. Each calibration TDMS file contained voltage signals from all pressure channels at one known pressure level to verify that all channels display the same readings and characteristics An example of the calibration comparison is shown in Fig. 4.14.

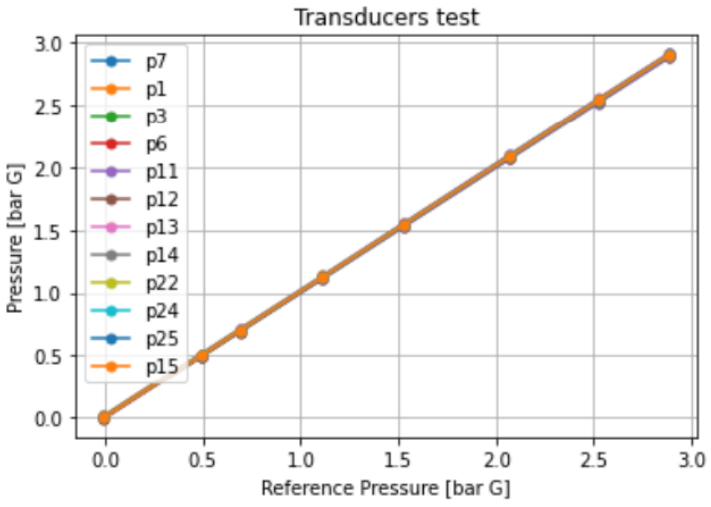


FIGURE 4.14: Calibration of the pressure measurement chain using a reference pressure transducer.

For each calibration file, the mean voltage was calculated over a steady part of the record. The zero offset was defined as the voltage measured at atmospheric pressure. This procedure allowed the same conversion approach to be used for both absolute and gauge pressure transducers. For the nozzle wall transducers, the zero offset could also be verified using the voltage level recorded immediately before diaphragm rupture, when the nozzle was still at atmospheric pressure.

The voltage to pressure conversion factor was determined from the voltage difference between successive calibration pressure levels. For each channel, the final divider was obtained as a weighted average and expressed in V/bar. The pressure transducer upstream of the primary nozzle, p_3, was used as the main reference channel during data normalization, because it provided the pressure used to determine the NPR and to present the wall pressures in dimensionless form. The calibration procedure and the corresponding processing code are included in Appendix F.

A comparison between the theoretical and experimentally determined voltage to pressure dividers is presented in Table 4.3. The theoretical values were calculated from the nominal transducer sensitivity, excitation voltage, and amplifier gain. The actual values were determined for the complete measurement chain, including the transducer, excitation source, amplifier, wiring, and data acquisition module. Therefore, the experimentally determined dividers were used in the final data reduction.

TABLE 4.3: Comparison of theoretical and experimentally determined voltage to pressure dividers for the pressure measurement channels.

Name	**Transducer**	**Theoretical Divider [V/bar]**	**Actual Divider [V/bar]**
p_7	22WP-116	0.4955	0.4959
p_1	22WP-115	0.83	0.83396
p_3	22WP-115	0.83	0.83000
p_6	22WP-116	0.4955	0.49807
p_{11}	XPM4	0.00085	0.00082
p_{12}	XP5	0.00429	0.00431
p_{13}	XP5	0.005115	0.00515
p_{14}	XP5	0.00545	0.0048
p_{22}	XTL-190	0.0143	0.01429
p_{24}	XTL-190	0.0143	0.01467
p_{25}	XTL-190	0.0143	0.01456
p_{15}	XTL-190	0.0142	0.01446

After calibration, each experimental run was processed individually. The relevant part of the signal was isolated from the full TDMS record. This interval included the establishment of the primary flow, the onset of secondary injection, and the transition

to the vectoring state. The beginning of the analyzed time window was selected so that the initial diaphragm rupture transient was excluded, while the end of the window was placed before the flow ceased to satisfy the quasi steady conditions of the Ludwieg tube run.

The output of this pre-processing stage was a synchronized data set containing calibrated pressure signals, calculated mass flow rates, and the time reference for the beginning of secondary injection. These data were used as the input for the subsequent time domain and frequency domain analyses.

4.5 Experimental Test Procedure

Each experimental run was performed according to a repeatable sequence controlled partly manually and partly by the LabVIEW acquisition system. The procedure included preparation of the initial pressure conditions, temperature stabilization of the compressed air in the driver tube, acquisition of all measurement channels, diaphragm rupture, delayed activation of the secondary injection, and termination of the run after the driver tube was discharged.

Before each run, the Ludwieg tube driver section was filled manually with compressed air. The filling process was controlled by an electromagnetic valve operated from a latch switch until the required initial pressure was reached. After filling, the system was left at rest until the air temperature measured near both ends of the driver tube became equal to the ambient temperature within the accepted measurement tolerance. This step was required to ensure repeatable initial stagnation conditions for consecutive runs.

After the pressure and temperature stabilization, data acquisition was started and all measurement channels were recorded to a TDMS file. The fast acting valve was then triggered from the LabVIEW host VI by NI-9482 relay module and electromagnetic valve.

The start of the primary flow was identified from the pressure signal of the first measurement point on the upper wall of the nozzle, by the function set in the LabVIEW FPGA structure. When the selected pressure signal reached the prescribed trigger level, corresponding to half of the target value, a programmed delay was initiated. After this delay, the NI-9482 relay module opened the solenoid valve in the secondary flow line, initiating secondary injection into the DTN. In this way, the experiment first established the axisymmetric primary flow condition and then introduced the secondary flow after a controlled time interval.

The full run duration was limited by the available quasi steady time of the Ludwieg tube and was contained within approximately 100 ms. The delay between start of the primary flow and the activation of secondary injection was set to 30 ms. This interval was

sufficient for the initial axisymmetric flow through the nozzle to stabilize before the onset of thrust vectoring conditions. After the driver tube was discharged, data acquisition was stopped and all electromagnetic valves were closed. The specific FPGA control sequence is shown in the Figure 4.15. Specific steps, programmed as LabVIEW sequence structure are as follows:

1. send signal to membrane burst actuator,
2. detect when p_3 signal reaches the value of 0.5 V,
3. wait 30 ms,
4. send signal to open secondary flow valve,
5. disable the actuators.

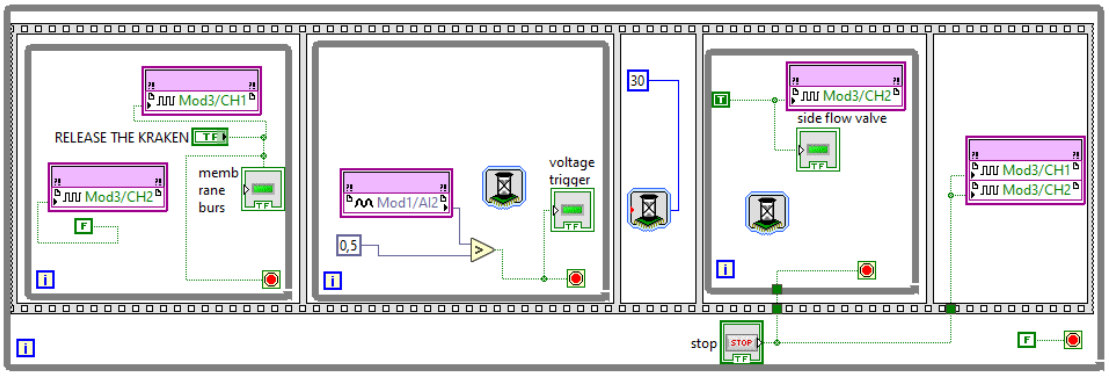


FIGURE 4.15: LabVIEW FPGA control sequence

The sequence of operations during a single run can be summarized as follows:

1. Fill the Ludwieg tube driver section to the required initial pressure.
2. Wait until the compressed air temperature in the driver tube equalizes with the ambient temperature.
3. Start acquisition of all measurement channels and save the data to a TDMS file.
4. Trigger diaphragm rupture using the pneumatic actuator controlled from the LabVIEW host VI.
5. Detect the start of the primary flow phase from the selected pressure signal.
6. After the prescribed delay, open the secondary flow solenoid valve using the NI 9482 relay module.
7. Continue acquisition until the driver tube is discharged, then stop recording and close the electromagnetic valves.

An example of the pressure signals recorded during a test is shown in Fig. 4.16. The pressure histories indicate successive phases of the run: initial filling state, diaphragm rupture, establishment of the primary flow, activation of the secondary injection, and decay of the flow after driver tube discharge. The time shift between the pressure increase in the driver tube signal and the pressure response upstream of the nozzle is consistent with the wave propagation mechanism of the Ludwieg tube. This confirms that the useful measurement interval corresponds to the quasi steady period between the starting transient and the return of the reflected wave to the nozzle.

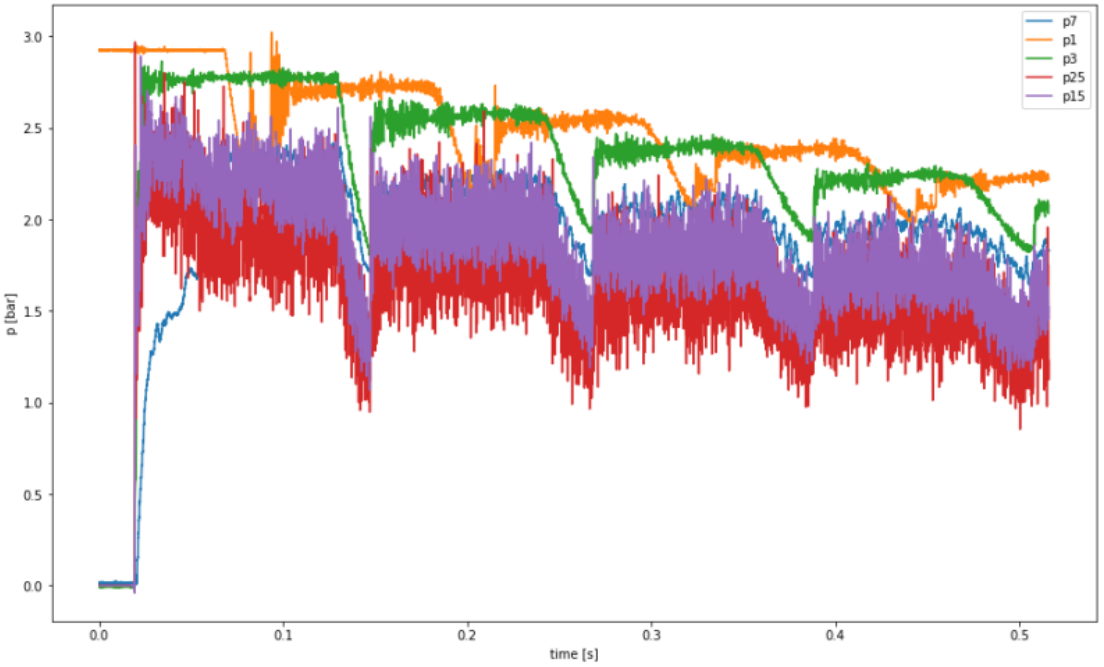


FIGURE 4.16: Example measured values

Chapter 5

Data analysis and the experimental results

5.1 Preliminary overview of the results

The experimental data obtained during the tests contain both the global response of the nozzle to the secondary injection and high-frequency pressure fluctuations associated with the inherently non stationary character of the flow. Therefore, before introducing the quantitative data analysis procedure, it is useful to first examine a representative raw time history of the measured quantities. This overview allows the main features of the response to be identified directly from the measurements and clarifies why additional signal processing and parameter estimation are required.

Figure 5.1 presents an example of the transient behavior of selected raw wall pressure signals together with the secondary mass flow rate. The presented case illustrates the sequence of phenomena observed after the secondary flow system is activated: the delayed increase of the injected mass flow, the gradual change in wall pressure distribution, and the superimposed pressure oscillations. Although the qualitative effect of the secondary injection is clearly visible, the raw signals do not allow for an unambiguous determination of characteristic response parameters, such as the delay time t_0 or the time constant τ.

Several important observations can be made from the presented signals. First, the onset of the secondary injection does not occur immediately after the control action, but appears after a measurable delay. Second, the wall pressure signals are affected by strong non-stationary fluctuations, which makes a direct interpretation of the dynamic response difficult. Third, the increase in secondary mass flow is accompanied by a change in wall pressure distribution, indicating the development of the vectoring flow field inside the nozzle. Finally, even in the initial non-vectoring state, the pressure values measured on opposite sides of the nozzle are not identical. This suggests that the primary jet is not

perfectly axisymmetric and may already be deflected by a small unintended angle, which is consistent with the asymmetry observed in the experimental pressure distribution.

For these reasons, analysis of the raw signals alone is insufficient for a reliable determination of the dynamic characteristics of the nozzle. Additional post-processing is required to separate the mean dynamic response from pressure pulsations and to determine the characteristic time parameters in a repeatable manner. In the following section, for step-like dynamic response, a first-order model with transport delay is therefore introduced and used to estimate the delay time t_0, the time constant τ, and the asymptotic value of the analyzed signal $y(t)$, representing either the wall pressure or the secondary mass flow rate.

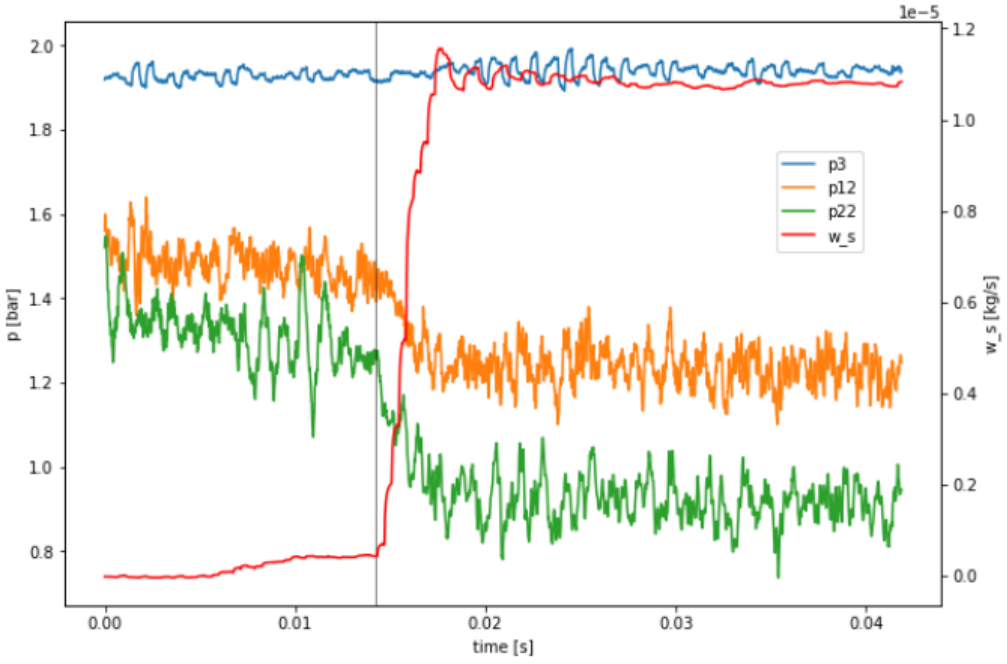


FIGURE 5.1: Pressures p_{12} and p_{22} and secondary mass flow $\dot{m}_s$. Vertical line indicates start of secondary flow.

5.2 Data analysis methods

All data-processing procedures described in this chapter were implemented using dedicated Python scripts. The processing chain was divided into separate stages corresponding to raw data segmentation, calibrated data reduction, time response identification, frequency domain analysis, and wall pressure profile generation. This structure allowed the original TDMS derived raw data to remain unchanged, while each subsequent processing step produced intermediate files that could be inspected independently.

5.2.1 Raw data segmentation and pre-processing

Before the quantitative analysis, the raw voltage arrays obtained from the TDMS conversion stage were reduced to time intervals containing the useful part of each Ludwieg tube run. This operation was performed using the script listed in Appendix G. The script reads the raw NumPy files, detects the beginning of the run from the selected reference pressure channel, and extracts a signal fragment containing 10000 samples before the detected start and the useful part of the transient flow record.

In the present implementation, channel p_{22} was empirically selected as the reference signal for start detection, because it provided a clear pressure rise after diaphragm rupture. Before thresholding, the signal was smoothed using a short moving average window of $0.8\,\mathrm{ms}$. The baseline level and noise amplitude were estimated from the initial part of the record, corresponding to the first 5% of the samples. The detection threshold was then set to the baseline value increased by six times noise amplitudes. This conservative threshold was selected to avoid triggering on electrical noise or isolated spikes. In addition, the signal had to remain above the threshold for 0.8ms and show a positive rise over the preceding 0.5ms. This ensured that the detected point corresponded to the beginning of a sustained pressure increase.

After the useful signal interval had been selected, isolated voltage spikes were removed using a Hampel type despiking procedure [43, 52]. The method compares each sample with the local median value calculated in a moving window. In the present implementation, a window of 11 samples was used and samples differing from the local median by more than six times local standard deviations were replaced by this median value. The relatively high threshold was used intentionally, so only physically meaningful pressure pulsations were preserved. The output of this stage consisted of a shortened signal file, saved as `_precut.npy`, and a corresponding zero offset file, saved as `_pre_zerooffset.npy`. These files formed the input for the subsequent calibrated pressure and mass flow analysis.

5.2.2 Mean values of the signal

For each analyzed signal, two quasi steady time intervals were extracted from the recorded time series. The first interval was located before the onset of secondary injection and is taken to represent the reference operating state of the nozzle. The second interval was selected after the completion of the dynamic response, once both the secondary mass flow and the wall pressure distribution had reached a new quasi steady condition. The windows were selected so that the initial and final levels were evaluated outside the transient part of the signal. Samples belonging to the intervening transition region were excluded from the computation of mean values.

The arithmetic mean of a discrete signal y_i over a selected interval containing N samples was computed as

$$\overline{y} = \frac{1}{N} \sum_{i=1}^{N} y_i. \tag{5.1}$$

In the present analysis, y_i can refer to either a pressure measurement at the wall or a quantity related to mass flow. Mean values determined from the interval preceding secondary injection are referred to as initial values, whereas those obtained from the post-transient interval are interpreted as final quasi-steady values. These quantities were used to describe the change in the pressure distribution caused by secondary injection and to establish reference levels for the analysis of the dynamic response.

The use of time-averaged quantities was necessary because the instantaneous pressure signals exhibited pronounced pulsations. Consequently, individual instantaneous samples extracted directly from the raw signals would not be representative of the underlying flow state. Averaging over carefully selected intervals attenuates the influence of local pressure pulsations and measurement noise, while retaining the mean effect of the secondary injection.

5.2.3 Smoothing the signals

The raw pressure signals contained both the mean dynamic response and superimposed pressure fluctuations. For this reason, direct fitting of the dynamic model to the raw data would make the estimated delay time and time constant dependent on the instantaneous phase of the fluctuations. Before the parameter identification step, the signals were therefore smoothed in order to extract the slowly varying component associated with the global response of the nozzle.

The Savitzky-Golay filter [78] was used for this purpose. This method performs a local least-squares fit of a polynomial to the data within a moving window. The value of the smoothed signal is obtained from the fitted polynomial at the centre of the window. In contrast to a simple moving average, the Savitzky-Golay filter preserves the local slope and the shape of the transition more effectively, which is important when the delay time and the time constant are determined from the transient part of the signal.

For a discrete signal y_i, the filtered value can be written in the general convolution form as:

$$y_{f,i} = \sum_{k=-m}^{m} c_k y_{i+k}, \tag{5.2}$$

where $y_{f,i}$ is the filtered value, c_k are the filter coefficients resulting from the local polynomial approximation, and $2m+1$ is the filter-window length. In the Savitzky–Golay method, the coefficients are obtained by fitting a polynomial of order p to the samples inside the moving window in the least-squares sense. For the central value of a quadratic Savitzky–Golay filter, which was used in this study, the coefficients can be written as:

$$c_k = \frac{3\left(3m^2 + 3m - 1 - 5k^2\right)}{(2m+1)\left(4m^2 + 4m - 3\right)}, \qquad k = -m, \ldots, m. \tag{5.3}$$

This formulation preserves the low-order polynomial trend of the signal while reducing high-frequency noise.

The implementation was carried out using the Python `scipy.signal` library, which provides the `savgol_filter` tool [93]. The window length was empirically selected to be long enough to reduce high frequency fluctuations but short enough not to shift or excessively flatten the dynamic response. During data processing, a sensitivity check with different lengths of the smoothing window was also performed. The resulting delay times and time constants showed only small changes within the tested range, and the fitted transient curves remained practically unchanged. This confirmed that the identified response parameters were not controlled by a single arbitrary filter setting. The same filtering procedure was applied consistently to all signals used in the time-response analysis.

The filtering was used only as a pre-processing step for the identification of the mean dynamic response. The raw signals were retained for the assessment of pressure fluctuations and for the frequency domain analysis. This separation was important because the oscillatory component is a physical feature of the non-stationary flow and should not be removed from the parts of the analysis where flow unsteadiness is discussed.

5.2.4 First-order model

The dynamic response of a DTN to secondary injection is the result of several coupled physical processes, including the opening dynamics of the secondary flow path, the development of the injected jet, the displacement of the separated flow region inside the cavity, and the redistribution of wall pressure. A complete description of these phenomena would require a fully unsteady flow field analysis. However, the purpose of the present experimental analysis is not to reconstruct the detailed instantaneous flow structure, but to determine repeatable and comparable time parameters describing the global response of the measured signals.

For this reason, a reduced-order model was introduced. The measured quantities, namely the secondary mass flow rate and the wall pressures, change from an initial quasi-steady level to a final quasi-steady level after the activation of the secondary injection. The transition between these two levels is not instantaneous. It contains an initial delay, associated with the actuation and transport of the secondary flow, followed by a gradual approach to the new mean state. Such behavior can be represented by a first-order, step-like, response with transport delay.

The first-order model was selected as the simplest model capable of describing the two key features required in this study: the delayed onset of the response and the finite rate of transition towards a new asymptotic value. Higher-order models could potentially reproduce local oscillations or more complex transient effects, but they would introduce additional parameters that are difficult to identify uniquely from signals affected by pressure pulsations. In contrast, the *first-order plus delay* model provides physically interpretable parameters: the delay time t_0, representing the beginning of the measurable response, and the time constant τ, representing the characteristic rate at which the signal approaches its final mean value.

It should therefore be emphasized that the model is not intended to describe the high-frequency non-stationary pressure fluctuations observed in the nozzle. These fluctuations are treated as a superimposed unsteady component of the signal. The model is used only to approximate the mean dynamic response between the pre-injection and post-injection states. This approach allows the response time to be determined in a robust and repeatable manner, while avoiding overfitting of local pressure oscillations.

The measured response to the opening of the secondary flow valve is asymptotic and approximately monotonic, which is typical of first-order systems with transport delay. Therefore, the following first-order plus delay model was assumed [81]

$$\hat{y}(t) = \begin{cases} y_0, & t < t_0, \\ y_\infty + (y_0 - y_\infty) \exp\left(-\dfrac{t - t_0}{\tau}\right), & t \geq t_0, \end{cases} \tag{5.4}$$

where:

- y_0 - quasi-steady value before the response, determined from the *before* window,
- y_∞ - quasi-steady value after the response, determined from the *after* window,
- t_0 - transport delay, corresponding to the beginning of the measurable response,
- τ - time constant of the mean dynamic response.

The time constant τ is defined as the time, measured from t_0, at which the response reaches the fraction $1 - e^{-1}$ of the total change between y_∞ and y_0. This can be written as

$$y(t_0 + \tau) = y_\infty + (y_0 - y_\infty)\, e^{-1} \tag{5.5}$$

A schematic interpretation of the first-order response model and the definitions of y_0, y_∞, t_0 and τ are shown in Figure 5.2.

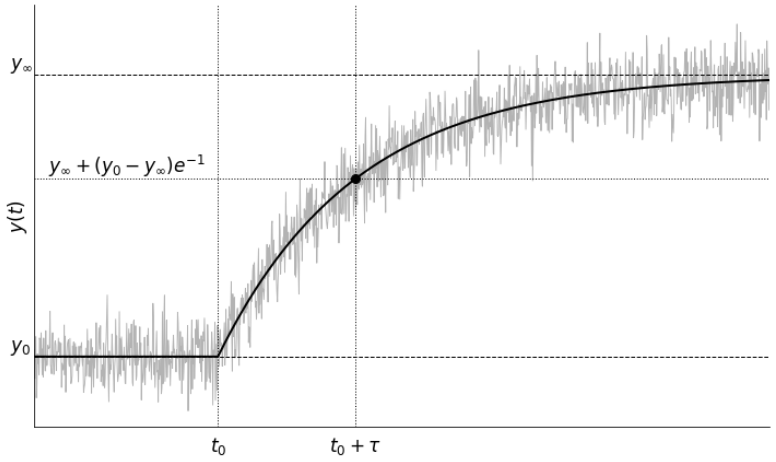


FIGURE 5.2: Parameters of the first-order system with transport delay

Because the steady-state values y_0 and y_∞ were obtained directly from the pre-transition (before) and post-transition (after) intervals, only two model parameters needed to be estimated from the smoothed signal. These parameters were grouped into the ordered pair:

$$\theta = (t_0, \tau). \tag{5.6}$$

5.2.5 Fitting the model function to the data

After determining the initial and final quasi-steady values and applying signal smoothing, a first-order model with transport delay was fitted to the selected time series. The parameter identification was performed only in the time interval containing the transition between the initial and final states. This interval was extended slightly before and after the detected transition to capture both the onset of the system response and the asymptotic approach to the new quasi-steady level. The amplitude of the response was defined as

$$K = y_\infty - y_0. \tag{5.7}$$

The initial estimates of y_0 and y_∞ were determined from the mean values computed over the "before" and "after" time intervals, respectively. During the parameter estimation

procedure, these quantities were treated as adjustable parameters, but constrained to vary within narrow bounds. This approach was adopted to accommodate small differences between the averaged quasi steady state levels and the local signal values within the transition interval used for fitting.

The optimal parameter set was obtained by minimizing the residuals between the smoothed experimental measurements y_i and the corresponding model predictions $\hat{y}_i$. The residual for the i-th sample was defined as

$$r_i = y_i - \hat{y}_i. \tag{5.8}$$

To fit the first-order model to the smoothed data, a nonlinear least-squares procedure based on the Trust Region Reflective algorithm was used. This method is grounded in the interior trust-region approach for bound-constrained nonlinear minimization introduced by Coleman and Li [13], together with its subspace implementation developed by Branch, Coleman, and Li [9]. The numerical implementation was performed in Python using the `least_squares` function from the `scipy.optimize` module [93]. This algorithm was chosen because it allows parameter bounds during the optimization process.

The following bounds were applied during the pressure signal fitting: the time constant was restricted to positive values within the analyzed interval, while y_0 and y_∞ were limited to a narrow neighbourhood around their respective reference mean values. Furthermore, the delay time associated with the wall pressure response was constrained by the dynamics of the secondary mass flow. Specifically, the secondary mass flow signal was fitted first, and the identified onset time was subsequently adopted as the lower bound for the delay time of the wall pressure response. This constraint prevents the fitted pressure response from starting before the actuation of the secondary injection.

The secondary mass flow signal was fitted using an analogous first-order delay function with zero initial value

$$\hat{w}_s(t) = \begin{cases} 0, & t < t_{0,\dot{m}_s}, \\ \dot{m}_{s,\infty}\left[1 - \exp\left(-\dfrac{t - t_{0,\dot{m}_s}}{\tau_{\dot{m}_s}}\right)\right], & t \geq t_{0,\dot{m}_s}, \end{cases} \tag{5.9}$$

where $\dot{m}_{s,\infty}$ is the final quasi-steady secondary mass flow, $t_{0,\dot{m}_s}$ is the onset time of the secondary flow, and $\tau_{\dot{m}_s}$ is its time constant. This fit was used to determine the physical reference time for the pressure-response fitting.

The time instant corresponding to one time constant after the beginning of the response was calculated as

$$t_{63} = t_0 + \tau. \tag{5.10}$$

For a first-order response, this instant corresponds to approximately 63.2% of the total change between y_0 and y_∞. Therefore, t_{63} was used as a convenient marker of the characteristic response time.

The quality of the fit was evaluated using the coefficient of determination R^2

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \overline{y})^2}, \tag{5.11}$$

where N denotes the number of samples within the fitted interval and $\overline{y}$ represents the mean value of the smoothed signal over this interval. The numerator corresponds to the residual sum of squares, whereas the denominator corresponds to the total sum of squares. An R^2 value approaching 1 indicates that the fitted model accounts for the majority of the variability in the smoothed signal. Conversely, lower values suggest that the first-order model provides a less accurate representation of the signal, for instance due to pressure oscillations, low response amplitude, or a non-monotonic transition.

The coefficient R^2 was employed solely as an auxiliary indicator of the goodness-of-fit. The physical interpretation relied primarily on the fitted delay time t_0, the time constant τ, the characteristic time t_{63}, and a non-dimensional time parameter used for inter-channel comparison.

The complete time-domain post-processing procedure was implemented in the Python script listed in Appendix H. The script loads the pre-cut voltage arrays and the corresponding zero-offset files, converts the voltage signals into absolute pressure using the experimentally determined calibration dividers, and calculates the secondary mass flow rate from the pressure measurements in the injection line. The same script performs transition-window detection, Savitzky-Golay smoothing, first-order delay fitting, calculation of t_0, τ, t_{63} and R^2, and generation of the summary plots and tables used in the time response analysis.

The post-processing was organized so that the secondary mass flow signal was fitted first. The identified onset time of $\dot{m}_s$ was then used as a lower bound for the onset of the wall pressure response. This constraint ensured that the fitted pressure response could not start before the physical activation of the secondary injection. The fitting results for all wall pressure channels were saved to a summary table and plotted together with the

raw signal, smoothed signal, before and after mean levels, and fitted first-order response, what will be show in the next chapter.

Chapter 6

Experimental results

6.1 Scope of the experimental results

This chapter presents the experimental results obtained for the investigated DTN during transient operation in the Ludwieg tube facility. The analysis is focused on the response of the nozzle to the activation of the secondary injection, with particular attention paid to the resulting changes in wall pressure, the dynamic character of the response, and the appearance of pressure fluctuations.

The chapter is organized as follows. First, the flow conditions of the test case are defined. Then, the time domain response of the wall pressure signals and secondary mass flow rate is analyzed in order to determine the delay time and response time of the nozzle. Next, the wall pressure distribution along the nozzle contour is examined. The frequency content of the pressure signals is then analyzed to assess the non-stationary character of the flow. Finally, the experimental pressure distribution is compared with the corresponding CFD results.

6.2 Flow conditions

The basic boundary conditions depended on the environmental conditions and were as follows:

- Atmospheric pressure: $p_0 = 101500\ Pa$
- Atmospheric temperature: $T_0 = 300\ K$
- Upstream temperature: $T_1 = 300\ K$

Flow conditions depended on the pressure values upstream of the nozzle p_3 as well as the pressure p_6 regulating secondary mass flow $\dot{m}_s$. The corresponding parameters are as follows:

- Nozzle Pressure Ratio: $NPR = 2.8$
- Primary mass flow: $\dot{m}_p = 0.445\,kg/s$
- Mass Flow Ratio: $MFR = 3\%$
- Secondary Mass Flow: $\dot{m}_s = 0.014\,kg/s$

The data analysis is described using this shot, which is closest to the flow conditions assumed by Flamm et al. [32] for the experimental geometry labeled "13-60", mentioned earlier in Section 3.

6.3 Time analysis results

6.3.1 Experiment results

Detailed results for the secondary mass flow and all wall pressure signals are listed in Table 6.1. The time analysis was performed for eight pressure measurement point on the wall. The identified time constants fell within the range from $\tau = 0.00079\ s$ to $\tau = 0.00244\ s$. The pressure signals measured at different wall taps have different t_0 values. This is expected because the local pressure response is affected by the propagation of the secondary injection effect through the separated flow region, pressure field pulsations, and the limited accuracy of the model fit. More importantly, the delay times for the pressures are nearly identical to the delay time for the secondary flow. In particular, point p_{24} has a delay time nearly identical to that of $\dot{m}_s$ ($t_0 = 0.01640\ s$ vs $t_0 = 0.01638\ s$), which clearly indicates that the dynamics of the actuator are of decisive importance. The influence of these factors is also reflected in the R^2 fit indices, which fell within the range from 0.87 to 0.98. The exception was the signal from point p_{25}, which, due to its high non-stationarity, could not be fitted using the methods described, and its fit was $R^2 = 0.44$.

It is difficult to estimate the magnitude of pressure changes during the transition directly from the raw signals due to the above mentioned unsteady flow and the non-zero thrust vector angle before intentional vectoring. It should be noted that, for some points, the pressure change was relatively small, on the order of $K = -0.08\ bar$, whereas for others it reached approximately $K = -0.24\ bar$ (Table 6.1). In all analyzed cases, the pressure decreased, which may be associated with an increase in local flow velocity and with flow separation in the cavity, as suggested by the CFD results (section 3.5), (Figure 3.9.

For comparison with DTNs of different dimensions and operating temperatures, and also for comparison with other results such as those reported by Ferlauto et al. [28], a non-dimensional time was added to the upper horizontal axis. It is defined as the ratio of time t to the characteristic flow time based on the nozzle length l and the reference isentropic velocity $\sqrt{RT}$

$$t^* = \frac{t\,\sqrt{RT}}{l}, \tag{6.1}$$

and consistently

$$\tau^* = \frac{\tau\,\sqrt{RT}}{l}. \tag{6.2}$$

TABLE 6.1: First-order delay fit parameters.

Type	Channel	K [bar]	t_0 [s]	τ [s]	t_{63} [s]	τ^* [-]	R^2 [-]
mass flow	$\dot{m}_s$	0.00967	0.01638	0.00176	0.01814	7.9	0.98
pressure	p_{11}	-0.093	0.01747	0.00079	0.01826	3.6	0.89
pressure	p_{12}	-0.096	0.01718	0.00147	0.01866	6.6	0.96
pressure	p_{13}	-0.078	0.01699	0.00181	0.01880	8.2	0.93
pressure	p_{14}	-0.079	0.01666	0.00244	0.01910	11.0	0.83
pressure	p_{22}	-0.243	0.01710	0.00127	0.01837	5.2	0.98
pressure	p_{24}	-0.197	0.01640	0.00193	0.01832	8.7	0.88
pressure	p_{25}	-0.141	0.01698	0.00072	0.01771	3.3	0.44
pressure	p_{15}	-0.090	0.01832	0.00163	0.01995	7.3	0.87

The non-dimensional time constant can also be referred to as the Strouhal number, in non-classical form as found in the literature [51], defined as :

$$\mathrm{S} = \frac{\tau\,u}{l}, \tag{6.3}$$

which, after substituting the isentropic velocity as velocity: $u = \sqrt{RT}$, leads to

$$\mathrm{S} = \frac{\tau\,\sqrt{RT}}{l}, \tag{6.4}$$

which, when compared to the equation 6.2 means that

$$\mathrm{S} = \tau^*, \tag{6.5}$$

This value is very useful from a design perspective. For identical thrust vector angle $\delta_p \approx 16°$, assuming a time constant equal to that of the longest response point $\tau^* = 0.00244\ s$, a dimensionless $\mathrm{S} = \tau^* = 11$, we can predict that, for example, at a reference

temperature $T = 1000\ K$ and a cavity length of DTN $l = 1\ m$, the response time constant will be $\tau \approx 0.02\ s$.

The image of the raw, smoothed and fitted values of wall pressure are shown in Figures from 6.1 to 6.8. The light blue lines represent the raw measurements, including the pressure pulsations and measurement noise discussed above. The orange lines represent the data smoothed by the Savitzky-Golay filter. Two horizontal dashed lines show the average pressure values in the "before" window (blue) and in the "after" window (orange). The optimized first-order response model is represented by a purple curve, with vertical markers indicating the time constant τ or its non-dimensional equivalent τ^*. For better comparison, each wall pressure graph is paired with a secondary flow graph. Appendix J contains complete graphs of the entire shot cycle of the Ludwieg tube. Time $t = 0$ is set at the moment when the signal is sent to the solenoid. To standardize the analysis and facilitate future comparisons, a time constant for valve dynamics was also determined.

As mentioned above, the solenoid valve and the secondary flow tap introduce their own dynamic response. In the secondary mass flow plot, the red curve represents the calculated secondary mass flow rate, while the pink curve represents the fitted first-order response. The final value reaches approximately $\dot{m}_s = 0.014\ kg/s$. The identified non-dimensional time constant of the secondary-flow rise was $\tau^* = 7.9$. It should be noted that the secondary mass flow does not start immediately after the reference trigger. The fitted delay time was $t_0 = 0.01638\ s$, which means that approximately 16 ms elapsed between the electrical activation signal and the beginning of the effective increase in secondary mass flow. A small disturbance visible shortly before the main rise of $\dot{m}_s$ is most probably related to the mechanical opening process of the valve and to the initial pressure redistribution in the secondary line. For very small values of $\dot{m}_s$ the calculated mass flow rate should not be taken into account, since the orifice plate correlation is strictly valid only under conditions of choked flow through the orifice.

The fitted response of the secondary mass flow was used as the reference for the interpretation of the wall pressure response. The identified delay time of the secondary mass flow was $t_{0,\dot{m}_s} = 0.01638\ s$, while the corresponding non-dimensional time constant was $\tau^*_{\dot{m}_s} = 7.9$. Therefore, the differences between the pressure response delay times and secondary flow delay, denoted as $\Delta t_0 = t_{0,p_i} - t_{0,\dot{m}_s}$, indicate how rapidly the internal pressure field reacted after the effective onset of secondary injection.

TABLE 6.2: Comparison of the wall pressure response parameters relative to the secondary mass flow response (from the shortest).

Channel	K [bar]	Δt_0 [s]	τ^* [-]	R^2 [-]
p_{24}	-0.197	0.00002	8.7	0.88
p_{14}	-0.079	0.00028	11.0	0.83
p_{13}	-0.078	0.00061	8.2	0.93
p_{25}	-0.141	0.00060	3.3	0.44
p_{22}	-0.243	0.00072	5.2	0.98
p_{12}	-0.096	0.00080	6.6	0.96
p_{11}	-0.093	0.00109	3.6	0.89
p_{15}	-0.090	0.00194	7.3	0.87

Table 6.2 compares the wall pressure response parameters with the fitted response of the secondary mass flow. The smallest delay difference was obtained for point p_{24}, for which $\Delta t_0 = 0.02\ ms$. This means that the pressure at this location started to change almost simultaneously with the effective increase in secondary mass flow. The next shortest delays were obtained for p_{14}, p_{13}, and p_{22}, all below 1 ms. The largest delay was observed at point p_{15}, where $\Delta t_0 = 1.94\ ms$, which suggests that the downstream pressure field reacted after the upstream part of the flow had already been modified by the secondary jet.

The comparison of the fitted pressure changes shows that the strongest response occurred on the upper wall. The largest pressure decrease was obtained at point p_{22}, with $K = -0.243\ bar$, followed by point p_{24}, with $K = -0.197\ bar$. In contrast, the lower wall pressure changes were smaller and remained close to $K = -0.08$ to $-0.10\ bar$. This confirms that the activation of secondary injection produced an asymmetric pressure redistribution inside the nozzle.

The non-dimensional time constants also show that the pressure field did not develop uniformly along the nozzle contour. The shortest reliable time constant was obtained for point p_{11}, while the longest value was obtained for p_{14}. Point p_{25} should be treated separately, because its low value of $R^2 = 0.44$ indicates that the fitted first-order response is not representative there. This signal is therefore more important as evidence of local non-stationary behavior than as a reliable source of the characteristic response time.

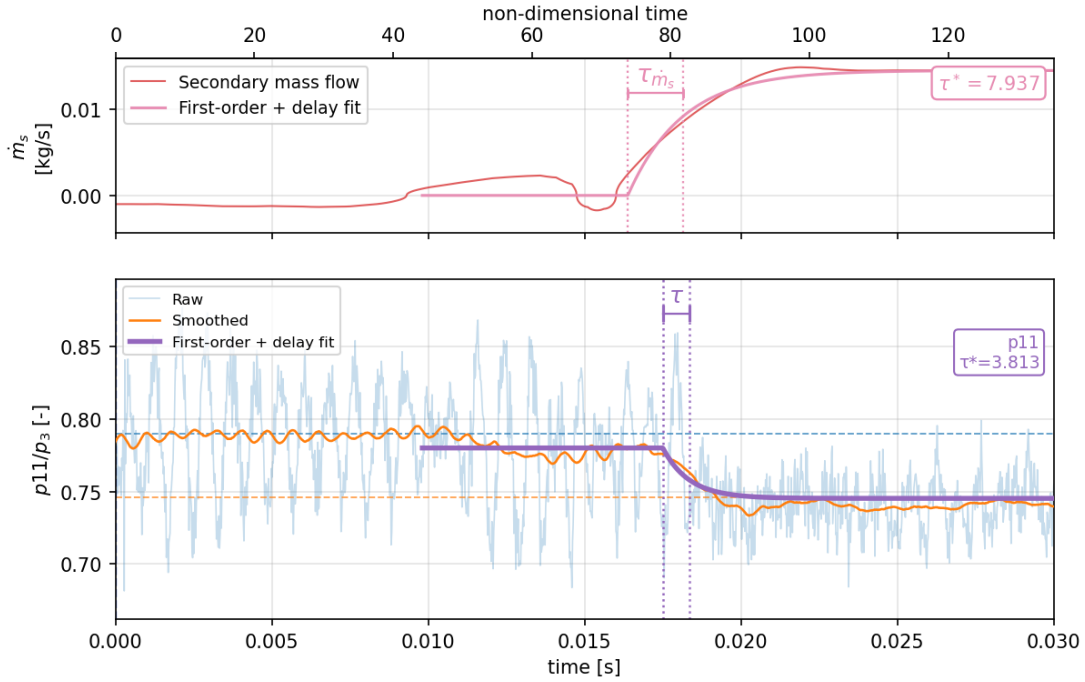


FIGURE 6.1: Raw, smoothed and fitted values of wall pressure at point p_{11} and secondary mass flow $\dot{m}_s$

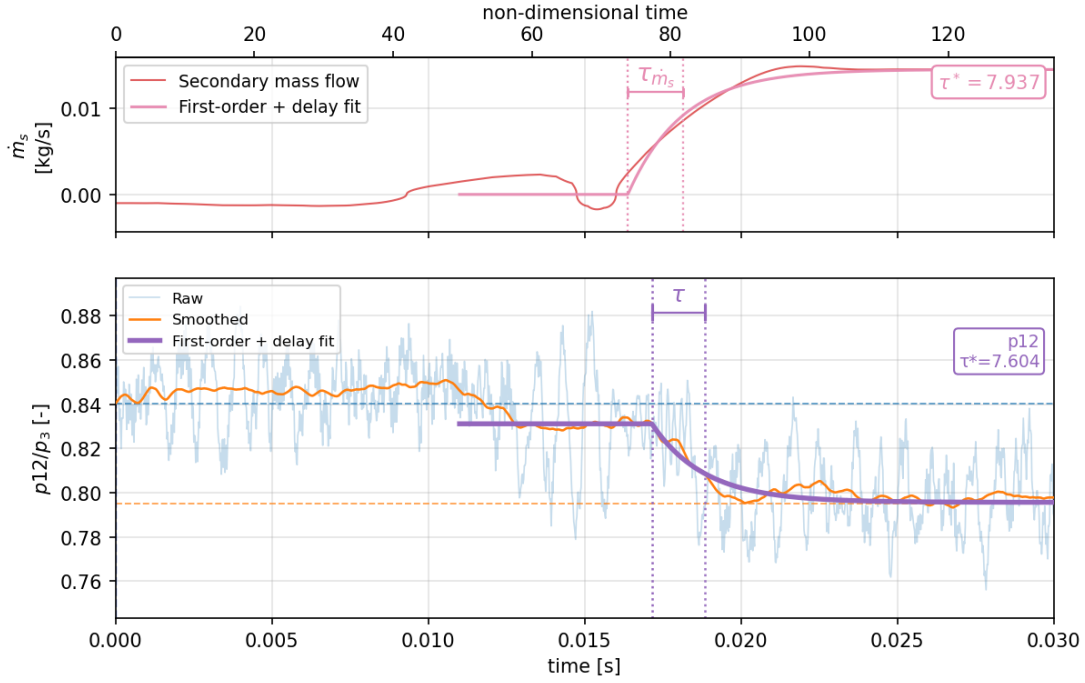


FIGURE 6.2: Raw, smoothed and fitted values of wall pressure at point p_{12} and secondary mass flow $\dot{m}_s$

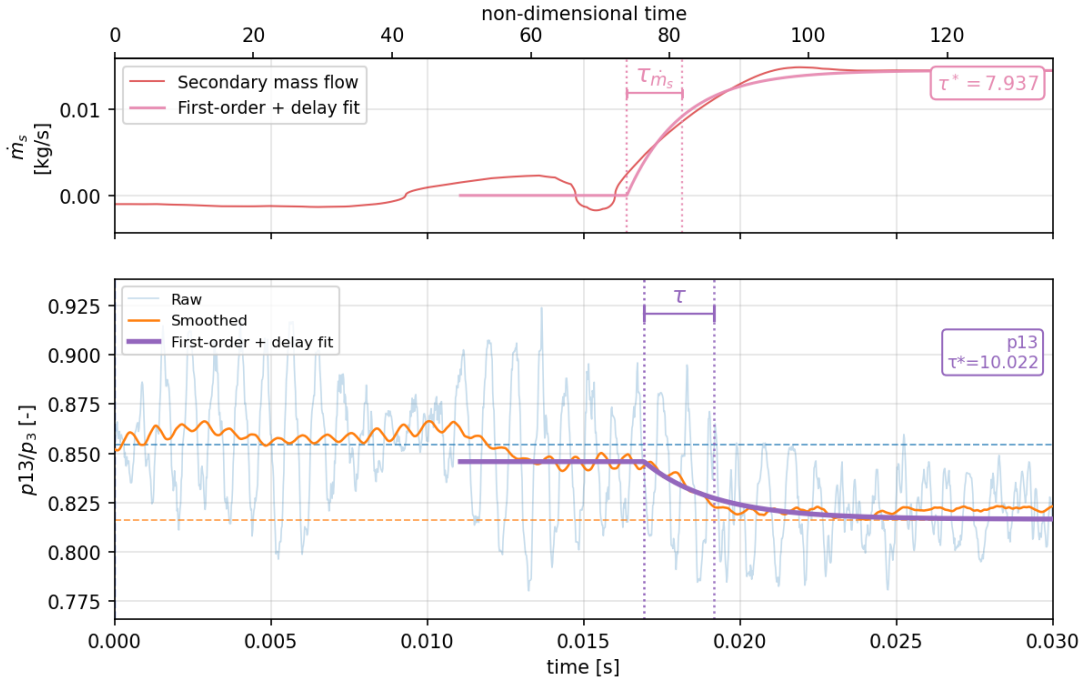


FIGURE 6.3: Raw, smoothed and fitted values of wall pressure at point p_{13} and secondary mass flow $\dot{m}_s$

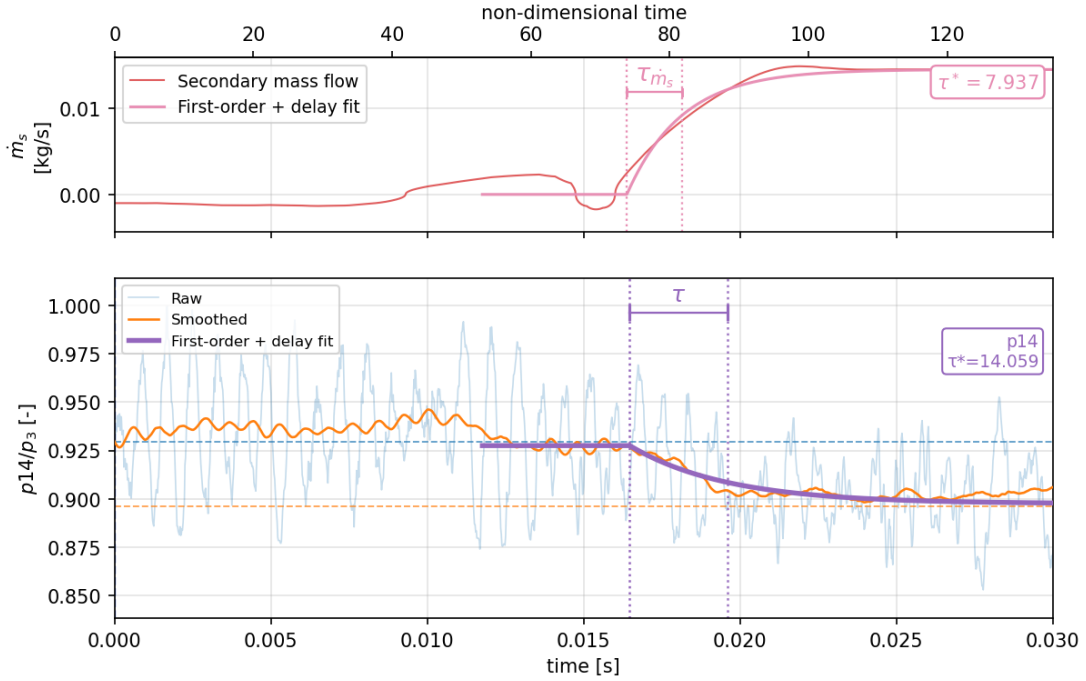


FIGURE 6.4: Raw, smoothed and fitted values of wall pressure at point p_{14} and secondary mass flow $\dot{m}_s$

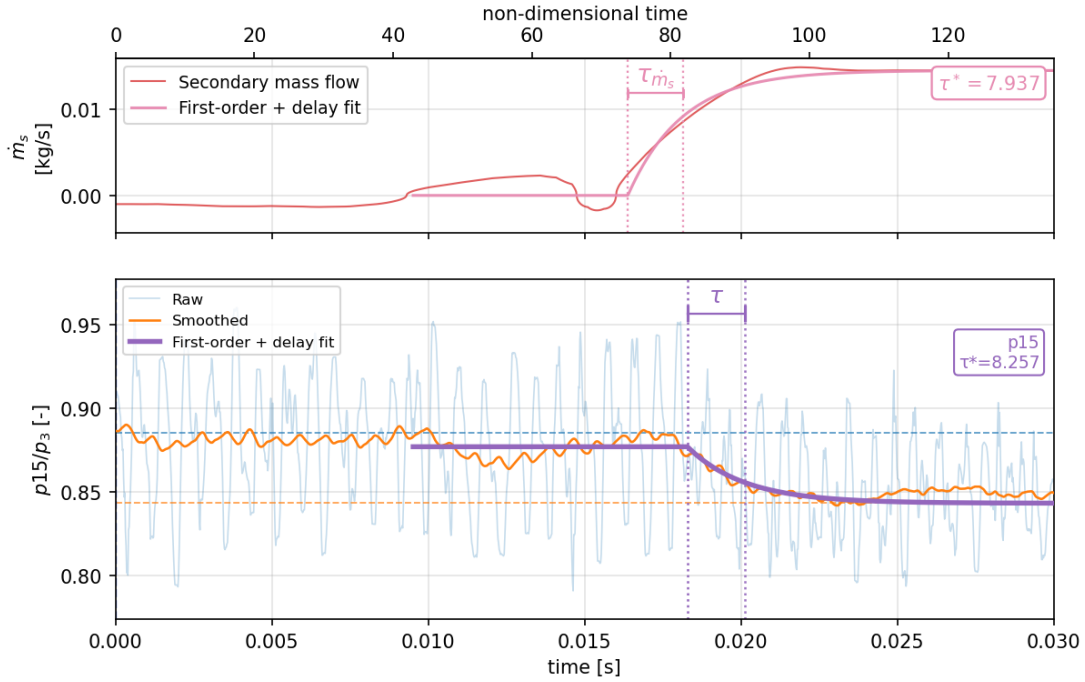


FIGURE 6.5: Raw, smoothed and fitted values of wall pressure at point p_{15} and secondary mass flow $\dot{m}_s$

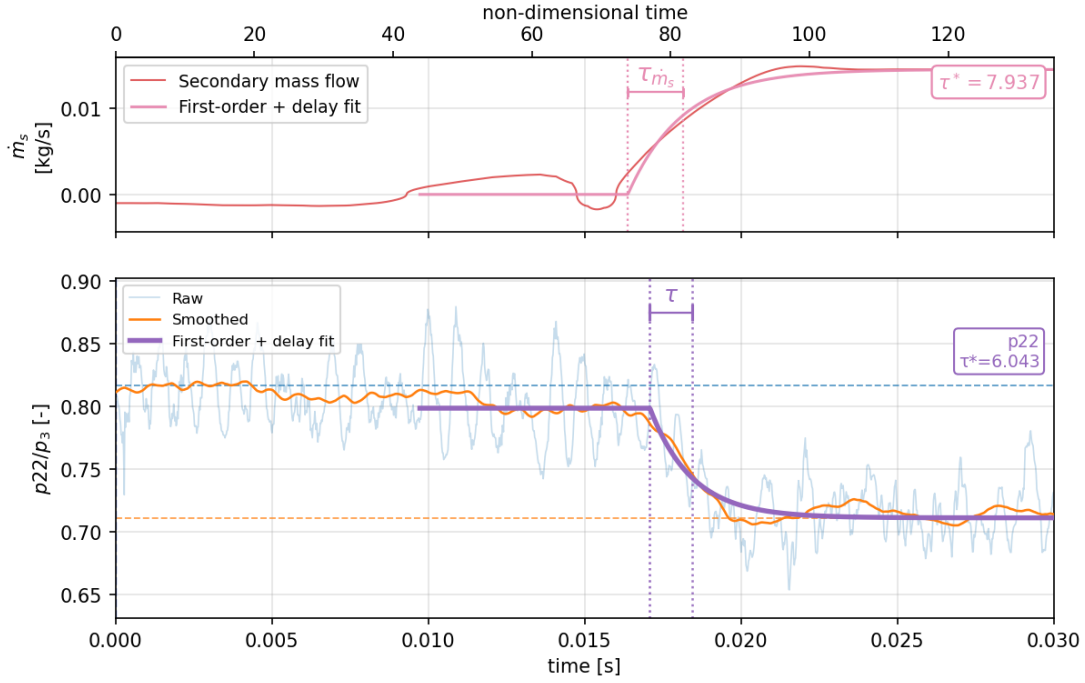


FIGURE 6.6: Raw, smoothed and fitted values of wall pressure at point p_{22} and secondary mass flow $\dot{m}_s$

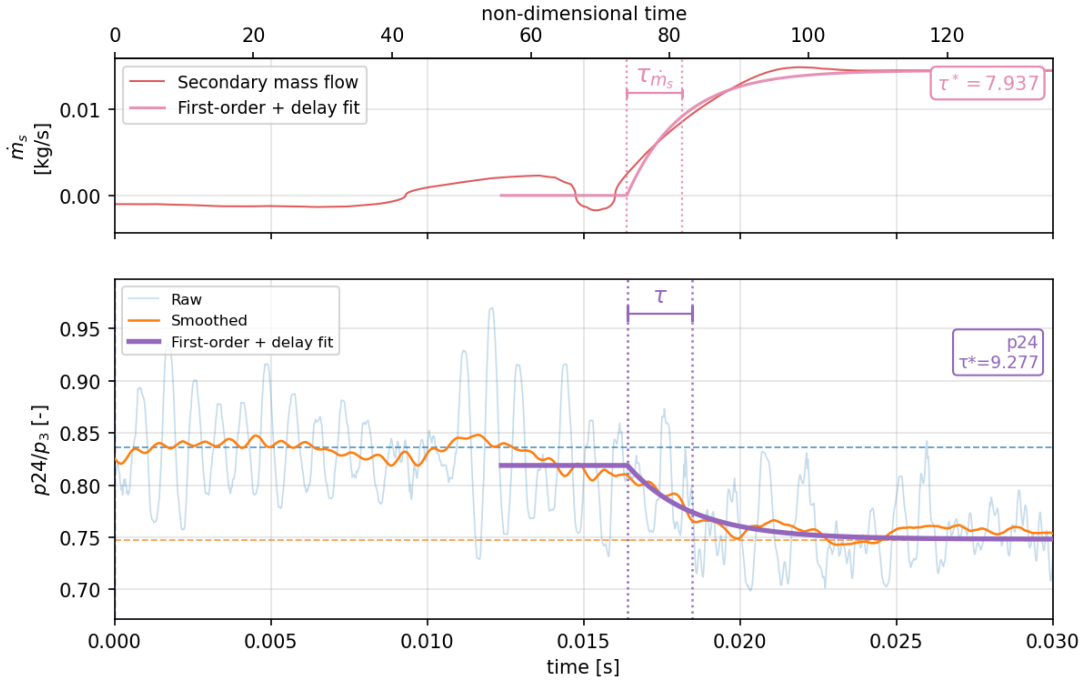


FIGURE 6.7: Raw, smoothed and fitted values of wall pressure at point p_{24} and secondary mass flow $\dot{m}_s$

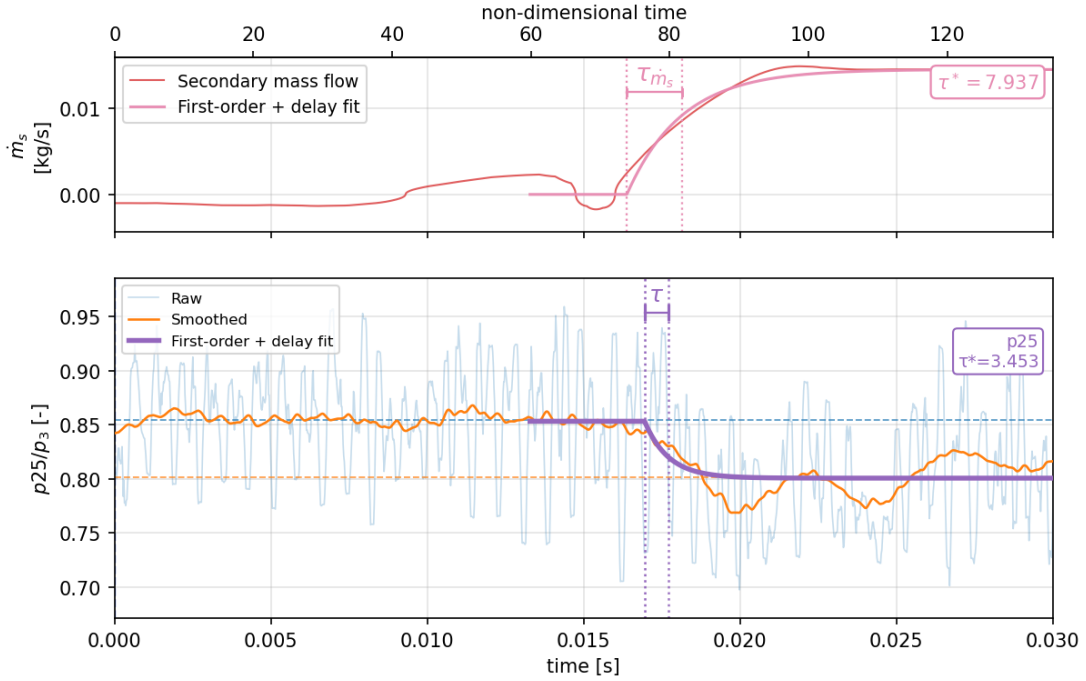


FIGURE 6.8: Raw, smoothed and fitted values of wall pressure at point p_{25} and secondary mass flow $\dot{m}_s$

6.3.2 Comparison with other studies

The most relevant reference for comparison is the numerical analysis conducted by Ferlauto and Marsilio [28, 29]. Their work was focused on the dynamic response of a DTN under fluidic actuation and also referred to the NASA LaRC family of DTN geometries. In contrast to the present experiment, their response quantity was not a local wall pressure but the thrust vector angle obtained from URANS calculations. Therefore, the comparison should be interpreted in terms of response time scale rather than direct equality of measured quantities.

In the present work, the representative experimental response time was taken as the largest reliable wall pressure time constant. Point p_{25} was not considered representative because of the poor fit quality, $R^2 = 0.44$. The largest reliable value was obtained at point p_{14} and was equal to $\tau^*_{p,\max} = 11.0$. The corresponding fitted time constant of the secondary mass flow response was $\tau^*_{\dot{m}_s} = 7.95$. Thus, the slowest reliable wall pressure response was approximately 1.38 times longer than the response time of the secondary injection system.

The values reported by Ferlauto and Marsilio show a similar coupling between actuator dynamics and nozzle response. In their case, an actuator dynamic of $\tau^*_{\text{act}} = 14.7$ produced a DTN response time constant of approximately $\tau^*_{\text{DTN}} \approx 14.2$. The ratio between the DTN response and the actuator response was therefore 0.97, whereas in the present experiment the corresponding ratio, based on the largest reliable wall pressure response, was 1.38.

A direct comparison of the absolute values shows that the DTN response time reported by Ferlauto and Marsilio was approximately 1.29 times larger than the largest reliable experimental wall pressure response obtained in the present work. A larger difference is observed for the actuation dynamics: their actuator time constant was approximately 1.85 times larger than the experimentally determined secondary mass flow time constant. This indicates that the difference between the two studies is partly related to the different imposed secondary-flow actuation dynamics.

The comparison is summarised in Table 6.3. The absolute values of τ^* differ, but both studies indicate that the DTN response time is of the same order as the secondary flow actuation time. This supports the conclusion that, for the investigated class of DTN configurations, the global response time is strongly coupled with the dynamics of the secondary injection system.

TABLE 6.3: Comparison of the present experimental response time with the numerical results of Ferlauto and Marsilio [28, 29].

Source	Actuation response τ^*	DTN response τ^*	Response ratio
Present experiment	7.95	11.0	1.38
Ferlauto and Marsilio	14.7	14.2	0.97

6.4 Wall pressure distribution & comparison with CFD results

The wall pressure distribution was obtained using the script listed in Appendix K. The script uses the pre-cut voltage data, the zero-offset file, and the time window indices previously determined by the time domain post-processor. The same two windows were used for all wall pressure taps: the *before* window, corresponding to the flow before the effective secondary injection, and the *after* window, corresponding to the developed vectorized state.

Additionally, for each wall pressure channel and for each time window, the corresponding standard deviation was computed. which is shown in Fig. 6.9 as an error bar. It represents the temporal variability of the normalized pressure inside the selected window, not the uncertainty of the pressure transducer.

The comparison between the *before* and *after* profiles shows that the secondary injection changed the pressure distribution on both nozzle walls. The pressure change was not uniform. The largest decrease occurred on the upper wall, especially at p_{22} and p_{24}. The lower-wall points showed smaller changes, remaining approximately within one group of similar pressure decreases. This confirms that the secondary injection produced an asymmetric redistribution of wall pressure inside the nozzle.

The strongest pressure response on the upper wall is consistent with the fitted dynamic response parameters discussed in Section 6.3. The same points (p_{22} and p_{24}) showed the largest values of the fitted pressure change K. Point p_{24} also had the smallest delay difference relative to the secondary mass flow response. Thus, both the time domain fit and the averaged wall pressure distribution identify the upper-wall divergent cavity region as the part of the nozzle most affected by the secondary injection.

The experimental pressure distribution was compared with the CFD wall pressure field discussed in Section 3.5. The comparison was made in terms of the sign and spatial distribution of the pressure change, rather than by direct point-to-point equality of pressure values. The CFD model represents an idealized geometry and boundary condition set, while the experiment includes the finite response of the pressure taps, short duration,

Ludwieg tube operation, and non-stationary pressure fluctuations. For this reason, exact agreement of p_i/p_3 at individual taps is not expected.

The agreement is qualitative but physically consistent. In both the CFD and the experiment, the secondary injection produces an asymmetric pressure field in the cavity region and downstream of the injection location. The experimental data confirm the same trend predicted numerically: the developed vectorized state is associated with a stronger modification of the pressure field on one side of the nozzle. The remaining differences are attributed mainly to the non-stationary character of the measured flow and to the simplified representation of the experimental boundary conditions in the CFD model.

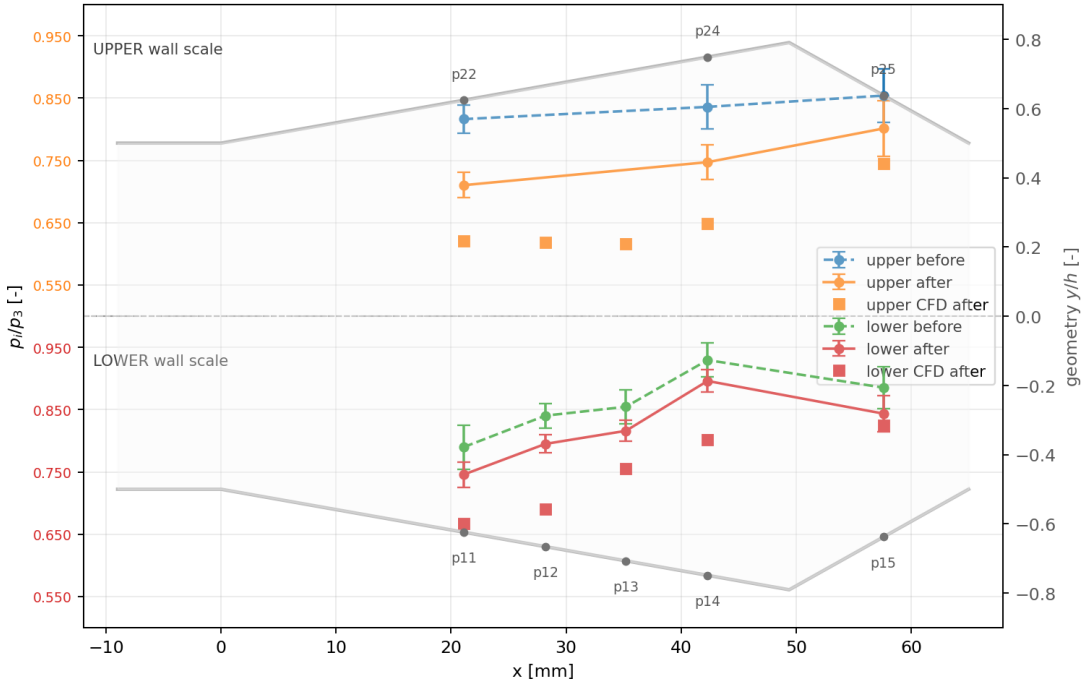


FIGURE 6.9: Wall pressure distribution before and after secondary injection, normalized by the primary stagnation pressure p_3. Error bars denote the standard deviation of p_i/p_3 in the corresponding time window.

6.5 Non-stationary flow

The time histories showed the appearance of strong pressure oscillations after the flow was established (blue line on the plots in Figures 6.1 to 6.8). Resonance in the pressure tap was considered as a possible source, but additional checks indicated that the amplitude and dominant frequency were not controlled by the shape of the pressure tap. Resonance in the pressure tap was discussed in Section 4.4.4, where first mode of resonance frequency was calculated as $f_1 = 2.39\ kHz$.

The frequency domain analysis was performed using a single sided amplitude spectrum obtained from the FFT. Before the transform was calculated, the mean value of

each analyzed signal fragment was subtracted. A Hann window was then applied to reduce spectral leakage caused by the finite length of the analyzed time interval. The analysis was performed separately for the before and after intervals (non-vectorized and vectorized flow states), which made it possible to compare their spectral content.

The frequency domain plots were generated using the script listed in the Appendix I. The script loads the same pre-cut data and zero-offset files as the time domain post-processor, converts all wall pressure channels into calibrated pressure values, and extracts the before and after windows defined during the time domain analysis. For each wall pressure channel, the script calculates the FFT amplitude spectrum and identifies the dominant peak frequency within the analyzed range. In addition to the graphical output, the script saved a table containing the dominant peak frequency and the corresponding amplitude for each channel analyzed and for both time intervals.

The FFT results for individual pressure taps are presented in Table 6.4 and the amplitude spectra are shown in Figure 6.11. In the first before (non-vectorized) interval, the dominant frequency at all points is similar and equals $f \approx 1200\ Hz$. The amplitude in this range varies from 0.026 to 0.055 bar. Analysis performed for interval after engaging secondary injection indicates a reduction in the amplitude of the dominant frequencies from 0.017 to 0.040 bar, as well as some blurring visible on the amplitude spectra in the form of multiple peaks around the dominant frequency. The frequencies with the highest amplitudes ranged from 800 to 1428.6 Hz.

Points p_{15} and p_{25}, located in the downstream part of the cavity close to the nozzle exit region, showed the largest dispersion of dominant frequencies and relatively high spectral amplitudes, which suggests the presence of unsteady separated or recirculating flow structures in the divergent part of the cavity. The similarity of the flow field on both sides of this nozzle section confirms the assumptions, based on numerical calculations, put forward in the literature by Wang et al. [94]. The presence of pressure fluctuations is qualitatively consistent with the numerical observations reported, where unsteady variations of DTN flow quantities were also identified. However, a direct quantitative comparison of dominant frequencies is not expected, because the numerical study was performed for different geometries and operating conditions. In particular, the period reported by Wang et al. was approximately 3.12 ms (Figure 6.10), which corresponds to 320.5 Hz, whereas the dominant frequencies measured in the present experiment were higher. The comparison should therefore be treated as qualitative support for the conclusion that DTN flow may remain unsteady even when the mean operating conditions are approximately constant.

The short measurement time results in low resolution of the analysis at low frequencies, and the use of zero-padding did not yield satisfactory results in terms of improving the accuracy of the results. The range of frequency values and a comparison with the

resonance frequency support the interpretation that the observed pressure pulsations are mainly caused by the non-stationary flow through the nozzle rather than by the measurement system itself.

TABLE 6.4: Dominant FFT peak frequency and amplitude before and after secondary injection.

Point	f_{before} [Hz]	A_{before} [bar]	f_{after} [Hz]	A_{after} [bar]
p_{11}	1200.0	0.043	1428.6	0.018
p_{12}	1085.7	0.032	857.1	0.017
p_{13}	1200.0	0.036	971.4	0.020
p_{14}	1200.0	0.033	914.3	0.019
p_{15}	1285.7	0.055	1428.6	0.037
p_{22}	1285.7	0.026	800.0	0.022
p_{24}	1257.1	0.041	971.4	0.030
p_{25}	1285.7	0.050	1428.6	0.040

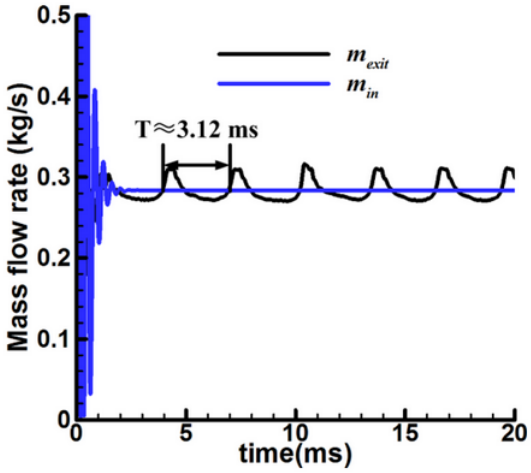


FIGURE 6.10: Calculated variation of mass flow rate [94]

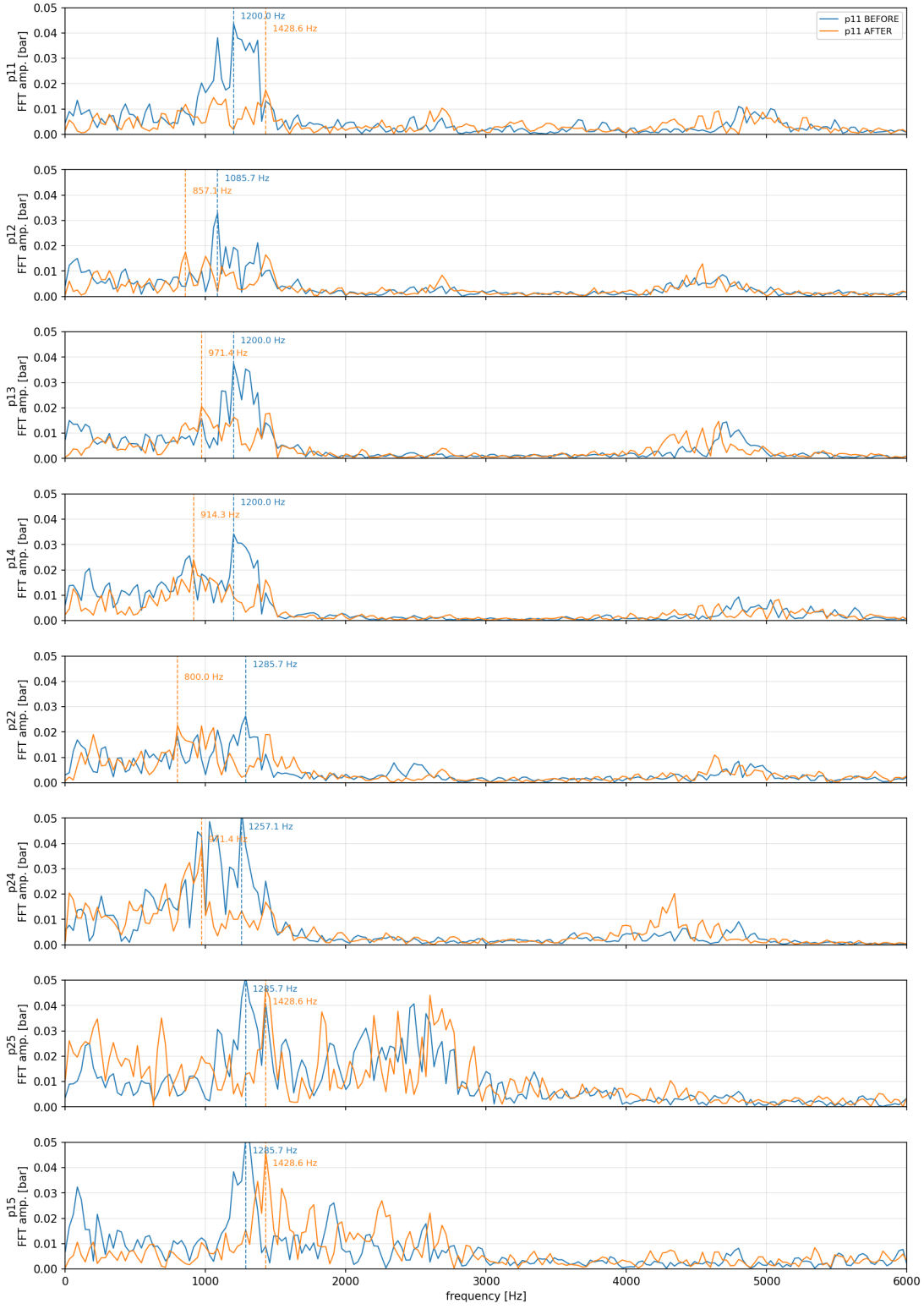


FIGURE 6.11: Frequency domain analysis of the wall pressure signals.

Chapter 7

Conclusion & future directions

The objective of this dissertation was to experimentally determine the dynamic response of an axisymmetric Dual-Throat Nozzle operating under transient secondary injection and to characterize the non-stationary pressure field inside the nozzle. The literature review demonstrated that previous studies concentrated mainly on steady-state performance, thrust vectoring efficiency and numerical simulations, while experimental data describing the transient response of DTN were not available. The presented work addresses this gap by providing the first experimentally determined dynamic response parameters for this type of nozzle.

To achieve this objective, a dedicated experimental facility based on a transonic Ludwieg tube was designed and commissioned. A modular DTN test section, a complete measurement system, and a dedicated data processing methodology were developed. Because the measured wall pressure signals contained significant oscillatory components, a processing procedure based on signal filtering and first-order model identification was implemented to separate the mean transient response from the superimposed pressure fluctuations. This enabled repeatable identification of the delay time and time constant for all analyzed measurement locations.

For the investigated operating conditions ($NPR = 2.8$, $MFR = 3\%$), and expected thrust vector angle $\delta_p \approx 16°$, the wall pressure response delay ranged from $t_0 = 0.01640$ to $0.01832\ s$, while the corresponding time constant ranged from $\tau = 0.00079$ to $0.00244\ s$. These results constitute, to the author's knowledge, the first experimentally determined dynamic response parameters of an axisymmetric DTN operating under transient secondary injection. The obtained values can be used directly in reduced-order dynamic models and preliminary control system design.

Comparison with the previously published numerical results of Ferlauto and Marsilio [28, 29] demonstrated agreement of the non-dimensional response time. Although the CFD model represented a different configuration and operating conditions, both

studies yielded response times of the same order of magnitude, confirming that the overall transient behavior of a DTN can be described by a characteristic non-dimensional response time.

The identified non-dimensional response time has practical engineering significance because it enables preliminary estimation of the response speed for geometrically similar nozzles operating under different conditions. Assuming the largest experimentally determined response time ($\tau = 0.00244\ s$), corresponding to a non-dimensional response time of approximately $\tau^* \approx 11$, the response of another DTN can be estimated directly. For example, for a nozzle with a cavity length of $l = 1\ m$ operating at a reference temperature of $T = 1000\ K$, the expected time constant is approximately $\tau \approx 0.02\ s$. Such scaling provides a simple engineering tool for preliminary DTN design and control system analysis.

The wall pressure measurements confirmed that secondary injection produced a clear redistribution of pressure over the nozzle walls, demonstrating successful establishment of the vectorized flow field. The experimentally measured pressure distributions exhibited agreement with the CFD predictions in terms of the overall pressure trends and location of the principal pressure changes, thereby indicating that the numerical model successfully reproduced the dominant flow mechanisms despite local discrepancies.

The frequency domain FFT analysis confirmed the presence of pressure pulsations in the DTN flow. In the before interval, the dominant frequencies were mainly in the range of about 1100-1300 Hz, with peak amplitudes of approximately 0.026-0.055 bar. After the activation of secondary flow, the dominant frequencies were less uniform and were found in the range of about 800–1400 Hz, while the corresponding amplitudes decreased to approximately 0.017-0.040 bar. Similar unsteady behavior of the DTN flow was also reported in numerical studies from the literature, where periodic variations of flow quantities were observed [94]. Therefore, the obtained results indicate that the DTN mass flow was not fully steady, but contained a measurable pulsating component both before and after secondary injection.

The principal original contributions of this dissertation are:

- development of a dedicated Ludwieg tube facility and experimental methodology for transient investigations of a Dual-Throat Nozzle,
- development of a repeatable data processing procedure enabling identification of DTN dynamic response parameters from short duration experiments,
- experimental determination of the delay time and time constant of an axisymmetric DTN operating under transient secondary injection,

- experimental confirmation that the transient wall pressure response can be represented by a first-order model with transport delay,
- experimental identification of non-stationary pressure oscillations occurring both before and after secondary injection,
- experimental verification of wall pressure redistribution associated with thrust vectoring together with comparison against CFD predictions.

The objectives formulated in the dissertation were achieved. The transient response of an axisymmetric Dual-Throat Nozzle was experimentally characterized by determining its delay time and time constant, while the frequency domain analysis demonstrated that the internal flow remains inherently unsteady during both non-vectorized and vectorized operation. The obtained results fill an identified gap in the experimental literature on DTN dynamics and provide quantitative data for reduced-order modeling, CFD validation and the design of future fluidic thrust vector control systems.

Future work should focus on extending the experimental database over a wider range of nozzle pressure ratios, secondary mass-flow ratios and nozzle geometries. Simultaneous measurements of thrust-vector angle, internal pressure distribution and dynamic response would enable direct correlation between transient wall-pressure evolution and vectoring performance. The proposed methodology may also be applied to other fluidic thrust-vectoring concepts, providing a consistent framework for experimental comparison of different nozzle configurations.

Appendix A

DTN test section geometry

The drawings of the DTN test section in this appendix include **only** the dimensions significant for the experimental studies.

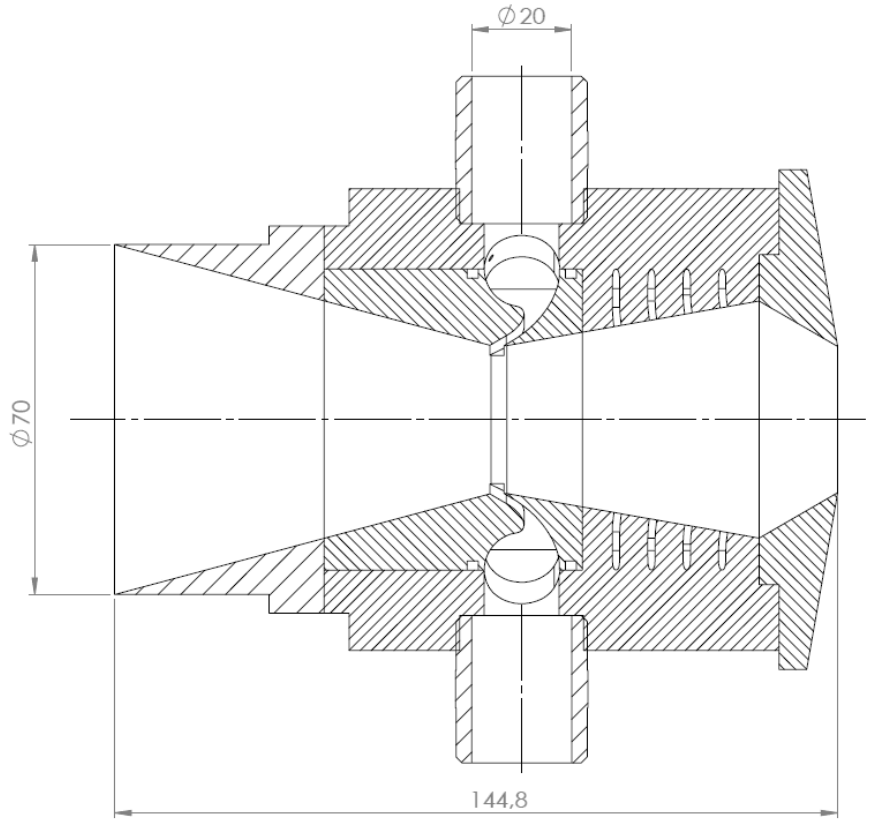


FIGURE A.1: DTN test section assembly

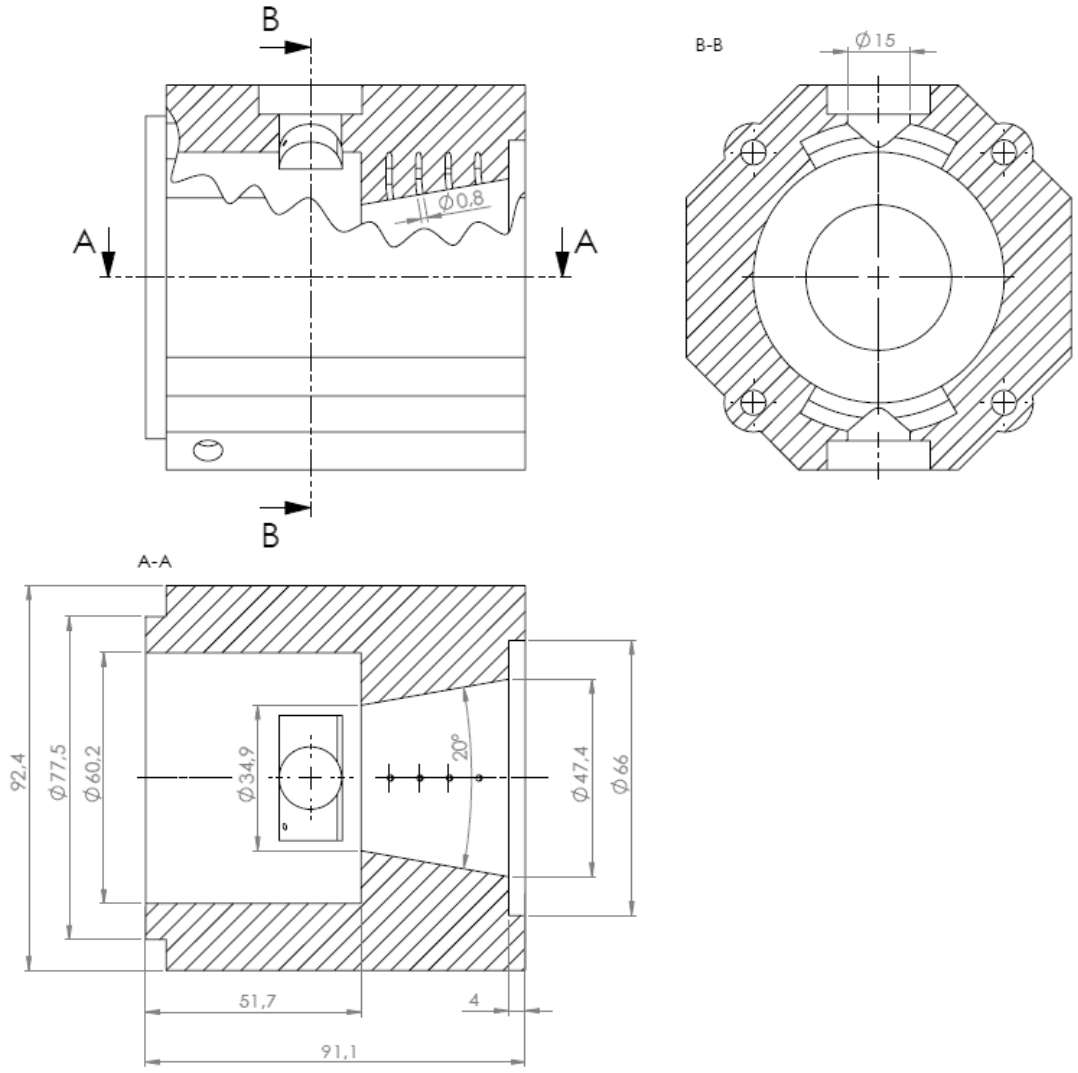


FIGURE A.2: DTN test section base and nozzle divergent part

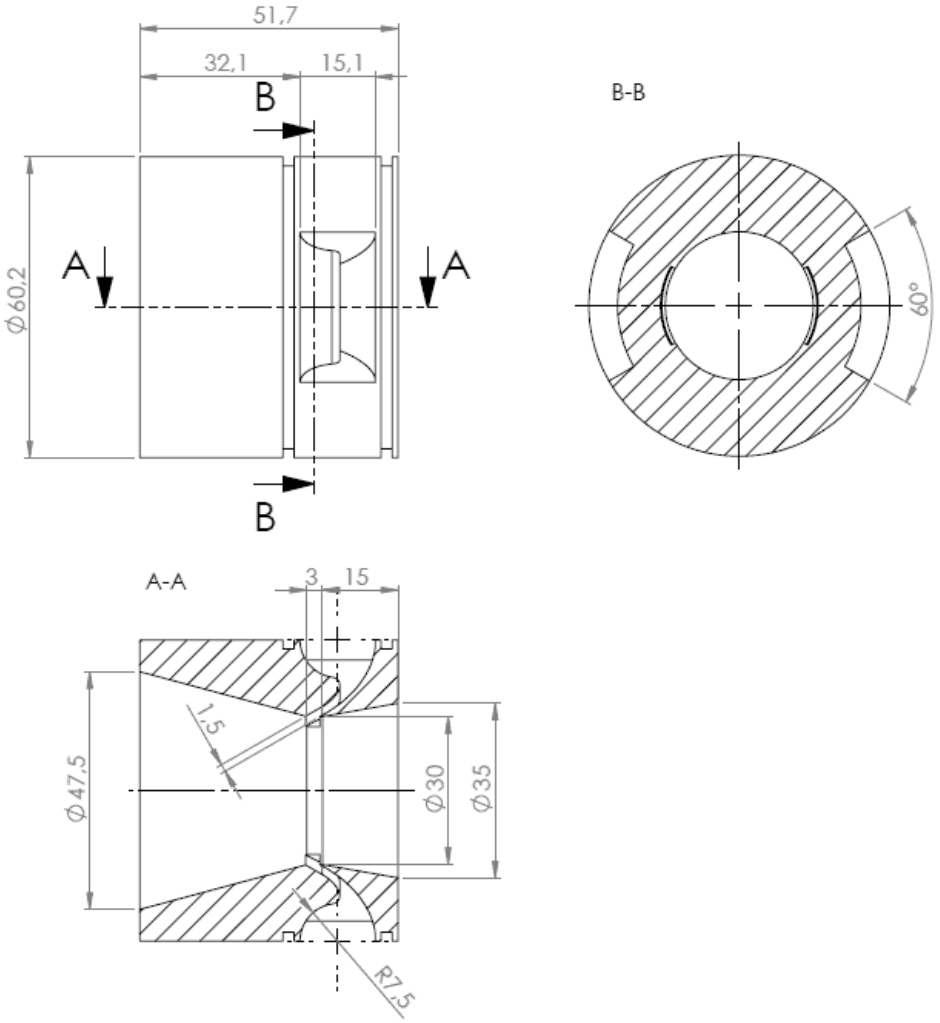


FIGURE A.3: Primary throat and secondary injection slot insert part

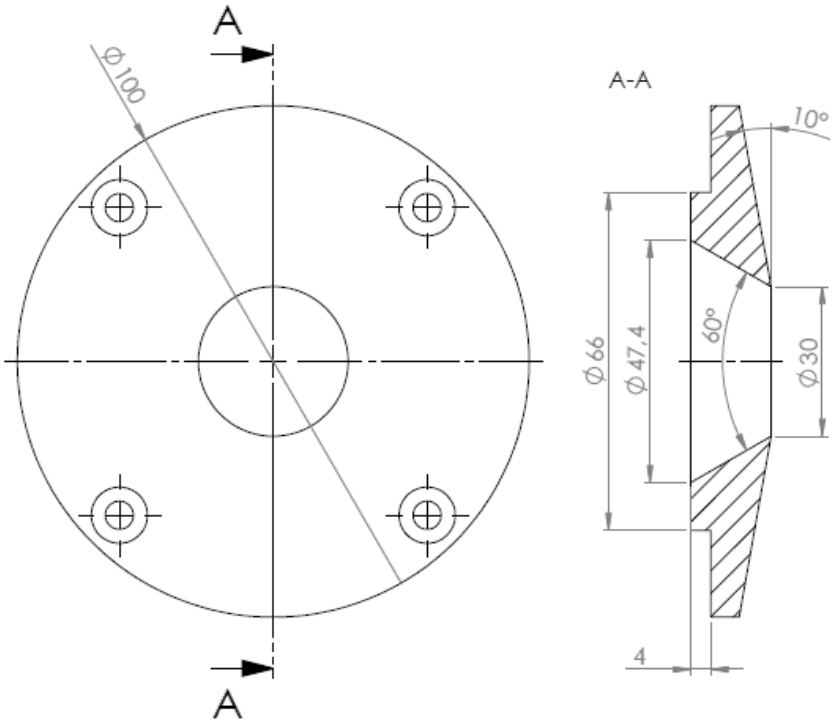


FIGURE A.4: DTN convergent and secondary (exit) throat part

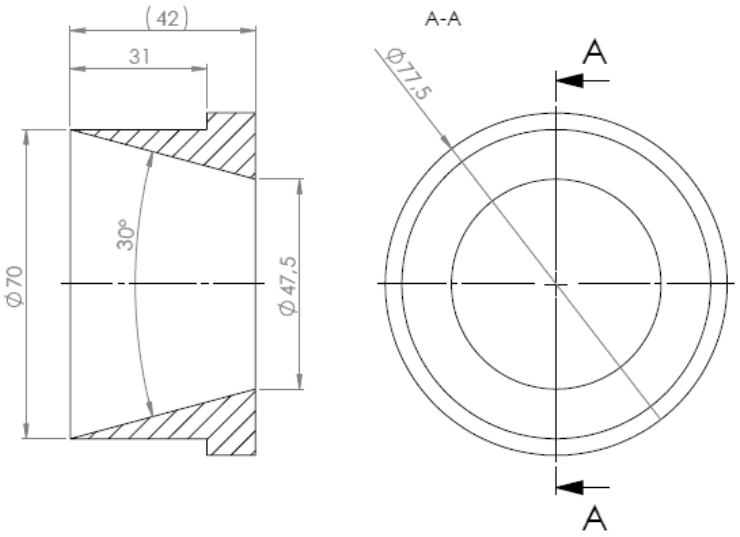


FIGURE A.5: DTN - Ludwieg tube adapter

Appendix B

Force Decoder

```
1
2   """
3   DTN_ForceDecoder_0.3.py
4   Wiktor Hoffmann
5
6   """
7   import numpy as np
8   import pandas as pd
9   from pathlib import Path
10
11  # ================== SETTINGS ==================
12
13  FILE = r"C:\Users\Victor\DTN\CFD_results\
14      DTN_connector_sec_order_transient_start_rep_fix-110000"
15
16
17  LAYER_THICKNESS = 0.58e-3   # m
18
19  USE_GAUGE_PRESSURE = True
20  P_REF = 101325.0            # Pa
21
22  DOMAIN_FACTOR = 2.0         # half-domain in XY symmetry plane
23
24  # normalna przekroju, dla outletu x = const
25  NX = 1.0
26  NY = 0.0
27  NZ = 0.0
28
29
30  # ================== READ FILE ==================
31
32  df = pd.read_csv(
33      FILE,
34      sep=r"\s*,\s*",
35      engine="python"
36  )
37
38  df.columns = [c.strip().lower() for c in df.columns]
```

```
39
40  for col in df.columns:
41      df[col] = pd.to_numeric(df[col])
42
43
44  # ================== DATA ==================
45
46  p = df["pressure"].to_numpy(dtype=float)
47  rho = df["density"].to_numpy(dtype=float)
48
49  vx = df["x-velocity"].to_numpy(dtype=float)
50  vy = df["y-velocity"].to_numpy(dtype=float)
51  vz = df["z-velocity"].to_numpy(dtype=float)
52
53  V_cell = df["cell-volume"].to_numpy(dtype=float)
54
55  if USE_GAUGE_PRESSURE:
56      p = p - P_REF
57
58
59  # ================== AREA FROM CELL VOLUME ==================
60
61  A = V_cell / LAYER_THICKNESS     # m^2
62
63  A_half = np.sum(A)
64  A_full = DOMAIN_FACTOR * A_half
65
66
67  # ================== NORMAL VELOCITY ==================
68
69  vn = vx * NX + vy * NY + vz * NZ
70
71
72  # ================== PRESSURE FORCE ==================
73
74  Fx_p = np.sum(p * A * NX)
75  Fy_p = np.sum(p * A * NY)
76  Fz_p = np.sum(p * A * NZ)
77
78
79  # ================== MOMENTUM FORCE ==================
80
81  Fx_m = np.sum(rho * vn * vx * A)
82  Fy_m = np.sum(rho * vn * vy * A)
83  Fz_m = np.sum(rho * vn * vz * A)
84
85
86  # ================== TOTAL FORCE - HALF DOMAIN ==================
87
88  Fx_half = Fx_p + Fx_m
89  Fy_half = Fy_p + Fy_m
90  Fz_half = Fz_p + Fz_m
91
92  F_half_3d = np.sqrt(Fx_half**2 + Fy_half**2 + Fz_half**2)
93
```

```
94
95  # ================== TOTAL FORCE - FULL DOMAIN ==================
96  # Symmetry plane: XY
97  # z-components cancel after mirroring.
98
99  Fx_full = 2.0 * Fx_half
100 Fy_full = 2.0 * Fy_half
101 Fz_full = 0.0
102
103 F_full = np.sqrt(Fx_full**2 + Fy_full**2)
104
105 alpha_xy = np.degrees(np.arctan2(Fy_full, Fx_full))
106
107
108 # ================== PRINT ==================
109
110 print("\n========== CFD FORCE FROM SECTION ==========")
111 print(f"Number of cells/nodes = {len(df)}")
112 print(f"Layer thickness       = {LAYER_THICKNESS:.6e} m")
113 print(f"Area half-domain      = {A_half:.6e} m^2")
114 print(f"Area full-domain      = {2.0 * A_half:.6e} m^2")
115
116 print("\n---------- HALF DOMAIN ----------")
117 print(f"Fx_pressure = {Fx_p:.6f} N")
118 print(f"Fy_pressure = {Fy_p:.6f} N")
119 print(f"Fz_pressure = {Fz_p:.6f} N")
120
121 print(f"Fx_momentum = {Fx_m:.6f} N")
122 print(f"Fy_momentum = {Fy_m:.6f} N")
123 print(f"Fz_momentum = {Fz_m:.6f} N")
124
125 print(f"Fx_total = {Fx_half:.6f} N")
126 print(f"Fy_total = {Fy_half:.6f} N")
127 print(f"Fz_total = {Fz_half:.6f} N")
128 print(f"F_total_3d_half = {F_half_3d:.6f} N")
129
130 print("\n---------- FULL DOMAIN, XY SYMMETRY ----------")
131 print(f"Fx = {Fx_full:.6f} N")
132 print(f"Fy = {Fy_full:.6f} N")
133 print(f"Fz = {Fz_full:.6f} N")
134 print(f"F  = {F_full:.6f} N")
135 print(f"alpha_xy = {alpha_xy:.6f} deg")
136 print("============================================\n")
```

Appendix C

CFD Pressure & FFT code

```
1
2   """
3   DTN_ForceDecoder_0.3.py
4   Wiktor Hoffmann
5
6   """
7   import re
8   import numpy as np
9   import pandas as pd
10  import matplotlib.pyplot as plt
11  from matplotlib.ticker import FormatStrFormatter
12
13
14  # ================== SETTINGS ==================
15
16  FILE = r"C:\Users\Victor\DTN\CFD_results\presure_all_1_1.out"
17
18  WINDOW = (0.003, 0.007)       # s, after optional time shift
19  SHIFT_TIME_TO_ZERO = True
20
21  P3_BAR = 2.85
22  P_SCALE = 1.0e-5 / P3_BAR     # Pa -> bar -> p/p3
23  P_LABEL = r"$p_i/p_3$ [-]"
24
25  REMOVE_MEAN_FOR_FFT = True
26  FFT_MAX_FREQ = 10000          # Hz; None = full range
27
28  SAVE_FIGS = False
29  OUT_PREFIX = "cfd_window_003_007"
30
31  # Display bands for pressure curves on the geometry plot.
32  # Coordinates are y/h, not pressure.
33  UPPER_PRESSURE_BAND = (0.52, 0.75)
34  LOWER_PRESSURE_BAND = (-0.75, -0.52)
35  PRESSURE_TICKS = 5
36  PRESSURE_PAD = 0.08           # relative padding of pressure axis for each wall
37
38
```

```
# ================== NOZZLE GEOMETRY ==================

GEO_UPPER_WALL = np.array([
    [-9.0,  15.00],
    [0.0,   15.00],
    [49.4,  23.71],
    [65.0,  15.00]
], dtype=float)

GEO_LOWER_WALL = np.array([
    [-9.0, -15.00],
    [0.0,  -15.00],
    [49.4, -23.71],
    [65.0, -15.00]
], dtype=float)

H_MM = 30.0

POS_X = {
    "p11": 21.1,
    "p12": 28.2,
    "p13": 35.2,
    "p14": 42.3,
    "p15": 57.6,
    "p21": 21.1,
    "p22": 28.2,
    "p23": 35.2,
    "p24": 42.3,
    "p25": 57.6,
}


# ================== FUNCTIONS ==================

def read_fluent_time_file(path):
    names = None
    rows = []

    with open(path, "r", encoding="utf-8", errors="ignore") as f:
        for line in f:
            line = line.strip()

            if not line:
                continue

            if line.startswith("(") and '"' in line:
                names = re.findall(r'"([^"]+)"', line)
                continue

            if line.startswith('"'):
                continue

            parts = line.split()

            if len(parts) > 1:
```

```
                try:
                    rows.append([float(x) for x in parts])
                except ValueError:
                    pass

    if names is None:
        raise ValueError("Nie znaleziono nazw kolumn.")

    df = pd.DataFrame(rows, columns=names)

    if "Time Step" in df.columns:
        df["Time Step"] = df["Time Step"].astype(int)

    return df


def channel_to_key(channel):
    return channel.replace("_", "").lower()


def wall_y_at_x(wall, x):
    return np.interp(x, wall[:, 0], wall[:, 1])


def calculate_fft(time, y, remove_mean=True):
    time = np.asarray(time, dtype=float)
    y = np.asarray(y, dtype=float)

    order = np.argsort(time)
    time = time[order]
    y = y[order]

    dt = np.mean(np.diff(time))
    fs = 1.0 / dt
    n = len(y)

    if remove_mean:
        y = y - np.mean(y)

    # No zero-padding.
    freq = np.fft.rfftfreq(n, d=dt)
    amp = 2.0 * np.abs(np.fft.rfft(y)) / n

    return freq, amp, fs, dt


def add_pressure_range(p):
    p = np.asarray(p, dtype=float)
    p_min = np.nanmin(p)
    p_max = np.nanmax(p)

    if np.isclose(p_min, p_max):
        d = max(abs(p_min) * 0.01, 1.0e-6)
    else:
        d = (p_max - p_min) * PRESSURE_PAD
```

```
149
150     return p_min - d, p_max + d
151
152
153 def pressure_to_band(p, p_lim, band):
154     p0, p1 = p_lim
155     b0, b1 = band
156     return b0 + (np.asarray(p) - p0) * (b1 - b0) / (p1 - p0)
157
158
159 def pressure_err_to_band(err, p_lim, band):
160     p0, p1 = p_lim
161     b0, b1 = band
162     return np.asarray(err) * abs(b1 - b0) / abs(p1 - p0)
163
164
165 def add_left_pressure_axis(fig, ax, p_lim, band, color, label=None):
166     # Transparent helper axis with the same physical y-range as the geometry axis,
167     # but with ticks labelled as pressure values only within the selected band.
168     aux = ax.twinx()
169     aux.patch.set_visible(False)
170     aux.set_ylim(ax.get_ylim())
171     aux.yaxis.set_ticks_position("left")
172     aux.yaxis.set_label_position("left")
173     aux.spines["right"].set_visible(False)
174     aux.spines["left"].set_position(("axes", 0.0))
175     aux.spines["left"].set_visible(False)
176
177     p_ticks = np.linspace(p_lim[0], p_lim[1], PRESSURE_TICKS)
178     y_ticks = pressure_to_band(p_ticks, p_lim, band)
179
180     aux.set_yticks(y_ticks)
181     aux.set_yticklabels([f"{v:.3f}" for v in p_ticks])
182     aux.tick_params(axis="y", colors=color, length=0, pad=32)
183
184     if label is not None:
185         aux.set_ylabel(label, color="black")
186         aux.yaxis.set_label_coords(-0.065, 0.5)
187
188     return aux
189
190 # ================= READ DATA =================
191
192 df = read_fluent_time_file(FILE)
193
194 time = df["flow-time"].to_numpy(dtype=float)
195
196 if SHIFT_TIME_TO_ZERO:
197     time = time - time[0]
198
199 pressure_channels = [
200     c for c in df.columns
201     if c not in ["Time Step", "flow-time"]
202 ]
203
```

```
w0, w1 = WINDOW
win_mask = (time >= w0) & (time <= w1)

df_win = df.loc[win_mask].copy()
time_win = time[win_mask]

if len(df_win) < 3:
    raise ValueError("Za mało próbek w wybranym oknie czasowym.")

# ================== WINDOW MEAN ==================

mean_pressure = df_win[pressure_channels].mean() * P_SCALE
std_pressure = df_win[pressure_channels].std() * P_SCALE

points = []

for ch in pressure_channels:
    key = channel_to_key(ch)

    if key not in POS_X:
        continue

    x = POS_X[key]

    if key.startswith("p1"):
        wall = "lower"
        y = wall_y_at_x(GEO_LOWER_WALL, x)
    elif key.startswith("p2"):
        wall = "upper"
        y = wall_y_at_x(GEO_UPPER_WALL, x)
    else:
        continue

    points.append({
        "channel": ch,
        "key": key,
        "wall": wall,
        "x_mm": x,
        "y_mm": y,
        "y_over_h": y / H_MM,
        "mean": mean_pressure[ch],
        "std": std_pressure[ch],
    })

points_df = pd.DataFrame(points)
points_df = points_df.sort_values(["wall", "x_mm"]).reset_index(drop=True)

print("\n========== WINDOW MEAN ==========")
print(f"window = {w0:.6f} - {w1:.6f} s")
print(f"samples = {len(df_win)}")
print(points_df.to_string(index=False))


# ================== PLOT: PRESSURE WITH GEOMETRY STYLE ==================

```

```
lower_df = points_df[points_df["wall"] == "lower"].copy()
upper_df = points_df[points_df["wall"] == "upper"].copy()

upper_p_lim = add_pressure_range(upper_df["mean"].to_numpy()) if not upper_df.empty
↪  else (0.0, 1.0)
lower_p_lim = add_pressure_range(lower_df["mean"].to_numpy()) if not lower_df.empty
↪  else (0.0, 1.0)

upper_df["plot_y"] = pressure_to_band(upper_df["mean"], upper_p_lim,
↪  UPPER_PRESSURE_BAND)
upper_df["plot_err"] = pressure_err_to_band(upper_df["std"], upper_p_lim,
↪  UPPER_PRESSURE_BAND)

lower_df["plot_y"] = pressure_to_band(lower_df["mean"], lower_p_lim,
↪  LOWER_PRESSURE_BAND)
lower_df["plot_err"] = pressure_err_to_band(lower_df["std"], lower_p_lim,
↪  LOWER_PRESSURE_BAND)

fig, ax = plt.subplots(figsize=(10, 6))

# Geometry in y/h.
upper_geo = GEO_UPPER_WALL.copy()
lower_geo = GEO_LOWER_WALL.copy()
upper_geo[:, 1] /= H_MM
lower_geo[:, 1] /= H_MM

# Filled nozzle area.
poly_x = np.r_[upper_geo[:, 0], lower_geo[::-1, 0]]
poly_y = np.r_[upper_geo[:, 1], lower_geo[::-1, 1]]
ax.fill(poly_x, poly_y, color="0.95", alpha=0.55, zorder=0)

ax.plot(upper_geo[:, 0], upper_geo[:, 1], color="0.75", lw=2.0, zorder=1)
ax.plot(lower_geo[:, 0], lower_geo[:, 1], color="0.75", lw=2.0, zorder=1)
ax.plot(upper_geo[:, 0], upper_geo[:, 1], color="0.55", lw=0.9, alpha=0.8, zorder=2)
ax.plot(lower_geo[:, 0], lower_geo[:, 1], color="0.55", lw=0.9, alpha=0.8, zorder=2)
ax.axhline(0.0, color="0.70", ls="--", lw=1.0, zorder=1)

# Geometry points and labels.
for _, row in points_df.iterrows():
    ax.plot(row["x_mm"], row["y_over_h"], "o", ms=4.5, color="0.45", zorder=3)
    va = "bottom" if row["wall"] == "upper" else "top"
    dy = 0.035 if row["wall"] == "upper" else -0.035
    ax.text(row["x_mm"], row["y_over_h"] + dy, row["key"],
            ha="center", va=va, fontsize=9, color="0.30", zorder=4)

# Pressure curves mapped into upper/lower bands.
if not upper_df.empty:
    ax.errorbar(upper_df["x_mm"], upper_df["plot_y"], yerr=upper_df["plot_err"],
                fmt="o-", capsize=4, lw=1.8, ms=5.5, label="upper wall", zorder=5)

if not lower_df.empty:
    ax.errorbar(lower_df["x_mm"], lower_df["plot_y"], yerr=lower_df["plot_err"],
                fmt="s-", capsize=4, lw=1.8, ms=5.5, label="lower wall", zorder=5)

ax.text(-11.0, 0.77, "UPPER wall scale", fontsize=11, color="0.25", va="center")
```

```
ax.text(-11.0, -0.14, "LOWER wall scale", fontsize=11, color="0.35", va="center")

ax.set_xlim(-12.0, 68.0)
ax.set_ylim(-0.9, 0.9)
ax.set_xlabel("x [mm]")
ax.yaxis.set_label_position("right")
ax.yaxis.tick_right()
ax.set_ylabel(r"geometry $y/h$ [-]")
ax.yaxis.set_major_formatter(FormatStrFormatter("%.1f"))
ax.tick_params(axis="y", colors="0.35")
ax.spines["left"].set_visible(False)
ax.grid(True, alpha=0.35)

# Left pressure scales.
add_left_pressure_axis(fig, ax, upper_p_lim, UPPER_PRESSURE_BAND,
                       color="C0", label=P_LABEL)
add_left_pressure_axis(fig, ax, lower_p_lim, LOWER_PRESSURE_BAND,
                       color="C1", label=None)

ax.legend(loc="center right")
fig.tight_layout()

if SAVE_FIGS:
    fig.savefig(f"{OUT_PREFIX}_wall_pressure_style.png", dpi=300)

plt.show()


# ================== PLOT: FFT FOR EACH CHANNEL ==================

plt.figure(figsize=(10, 6))

fft_info_printed = False

for ch in pressure_channels:
    y = df_win[ch].to_numpy(dtype=float) * P_SCALE

    valid = np.isfinite(time_win) & np.isfinite(y)
    freq, amp, fs, dt = calculate_fft(
        time_win[valid],
        y[valid],
        remove_mean=REMOVE_MEAN_FOR_FFT
    )

    if FFT_MAX_FREQ is not None:
        mask = freq <= FFT_MAX_FREQ
    else:
        mask = np.ones_like(freq, dtype=bool)

    plt.plot(freq[mask], amp[mask], label=ch)

    if not fft_info_printed:
        print("\n========== FFT INFO ==========")
        print(f"dt = {dt:.6e} s")
        print(f"fs = {fs:.3f} Hz")
```

```
        print(f"df = {freq[1] - freq[0]:.3f} Hz")
        fft_info_printed = True

plt.xlabel("frequency [Hz]")
plt.ylabel("FFT amplitude")
plt.title(f"FFT, t = {w0:.3f}-{w1:.3f} s")
plt.grid(True)
plt.legend(ncol=2)
plt.tight_layout()

if SAVE_FIGS:
    plt.savefig(f"{OUT_PREFIX}_fft.png", dpi=300)

plt.show()
```

Appendix D

LabVIEW structures

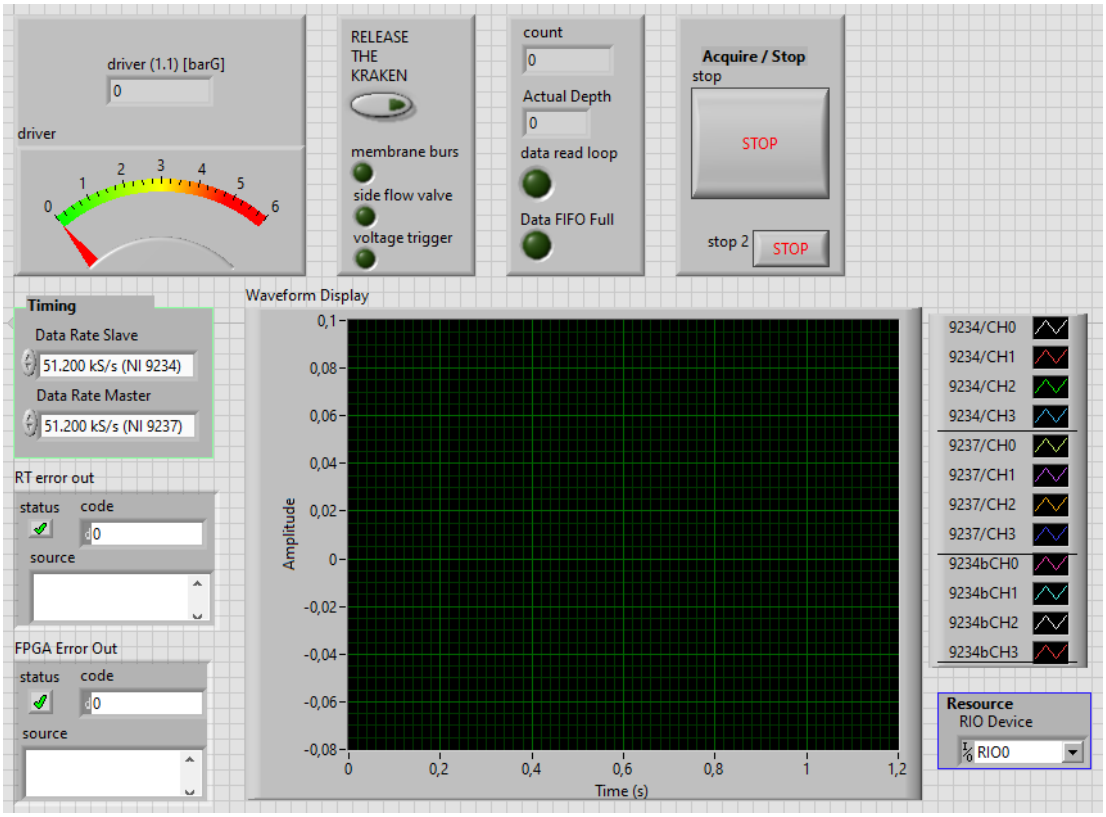


FIGURE D.1: LabVIEW host user interface

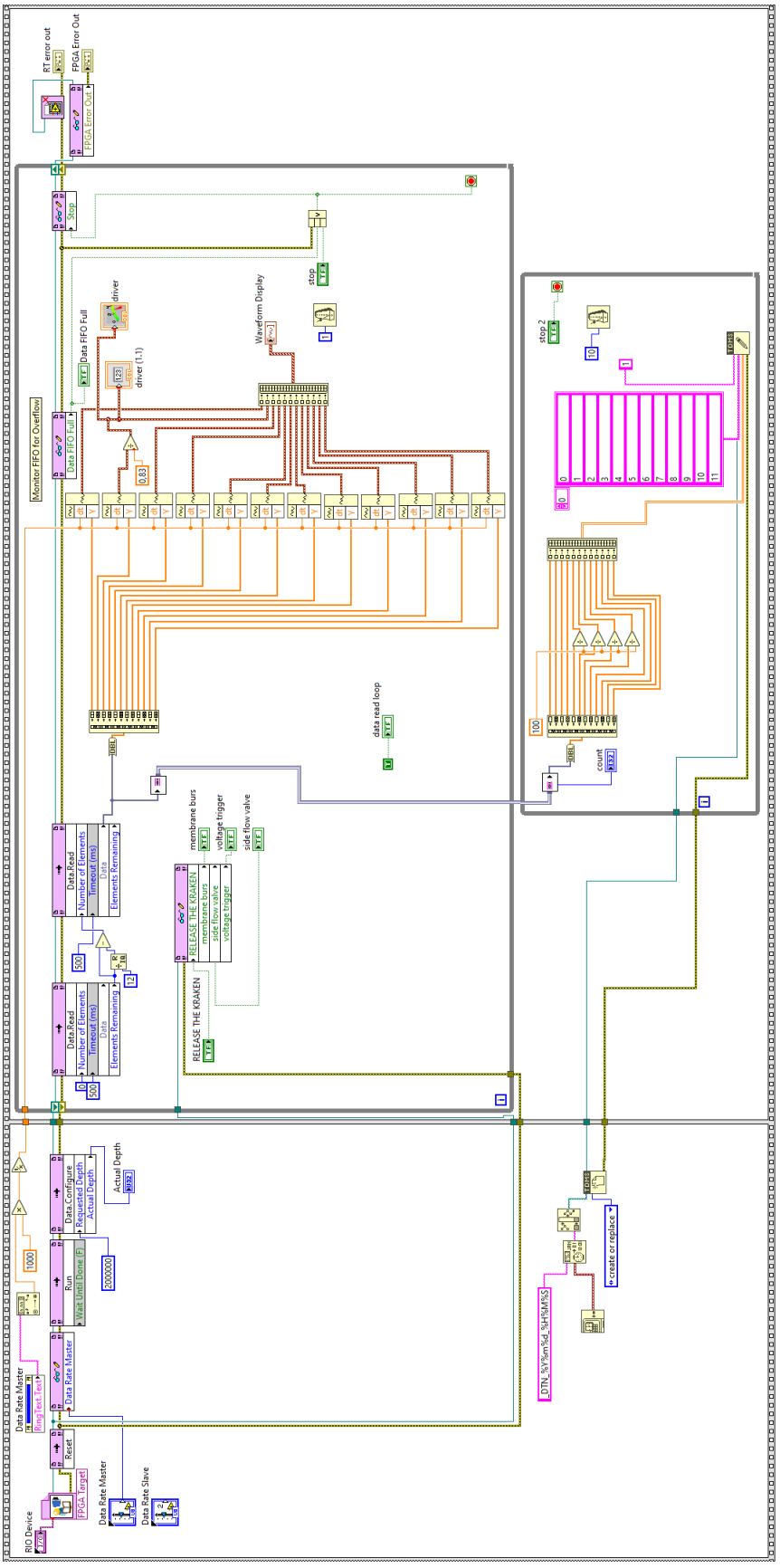


FIGURE D.2: LabVIEW host structure VI

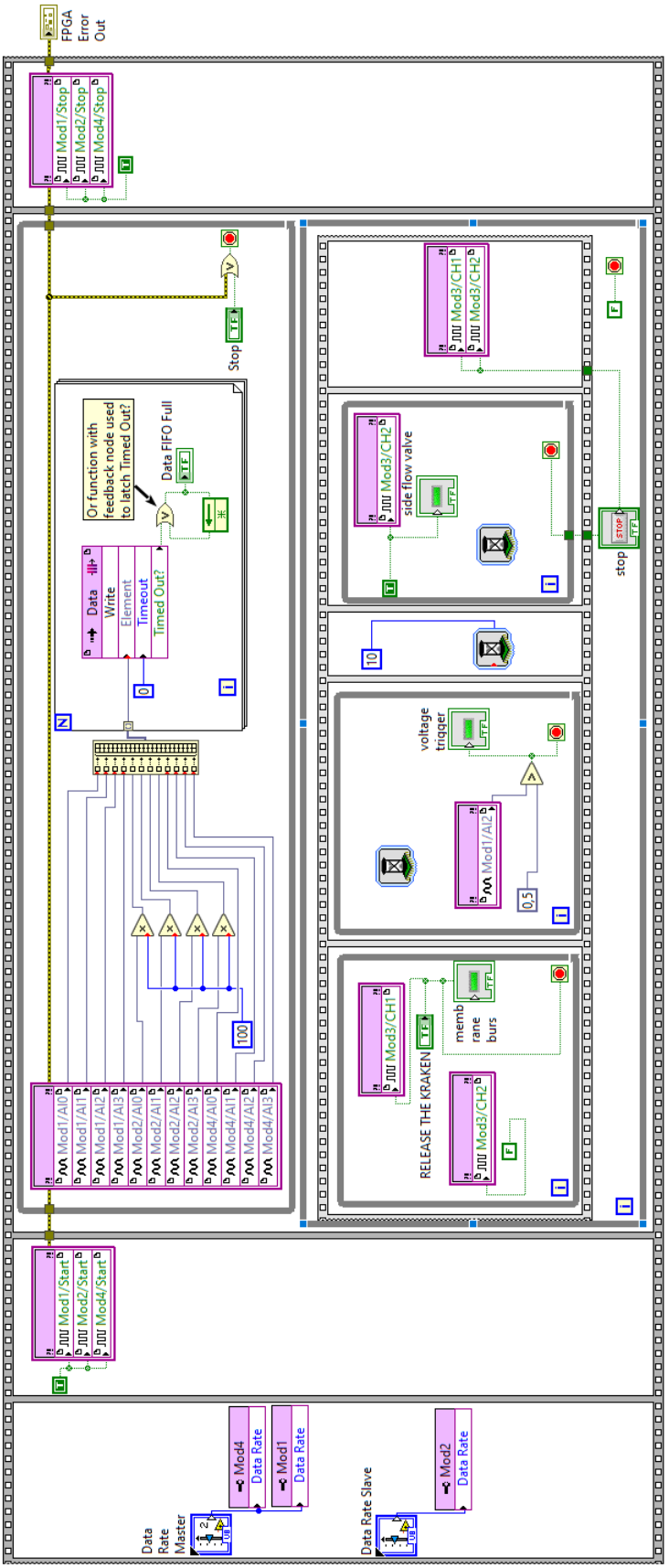


FIGURE D.3: LabVIEW FPGA structure VI

Appendix E

TDMS Reader

```
"""
TDMS_Reader_1.1.py
Wiktor Hoffmann
"""

import pathlib
import numpy as np
from nptdms import TdmsFile

BASE_DIR = pathlib.Path(r"C:/Users/hoffm/DTN/_out_20241030_1")
TDMS_DIR = BASE_DIR
RAW_DIR  = BASE_DIR / "raw_inputs"
RAW_DIR.mkdir(parents=True, exist_ok=True)
labels = ['p7','p1','p3','p6','p11','p12','p13','p14','p22','p24','p25','p15']
channels = [str(i) for i in range(len(labels))]
GROUP = '1'

def read_tdms(path):
    g = TdmsFile.read(path)[GROUP]
    n = min(len(g[ch]) for ch in channels)
    return np.asarray([g[ch][:n] for ch in channels], dtype=float).T

def main():
    files = sorted(TDMS_DIR.glob("*.tdms"))
    if not files:
        print(f"[WARN] brak TDMS w {TDMS_DIR}")
        return
    for p in files:
        try:
            U = read_tdms(p)
            out = RAW_DIR / (p.stem + "_raw.npy")
            np.save(out, U)
            print(f"OK: {p.name} -> {out.name} {U.shape}")
        except Exception as e:
            print(f"[ERROR] {p.name}: {e}")

if __name__ == "__main__":
    main()
```

Appendix F

Voltage to pressure conversion rate calculator

```
"""
Created on Mon Jul 10 13:32:40 2023
#### dividers V/bar ####
@author: Wiktor Hoffmann
"""
import time
start_time = time.time()
import numpy as np
from nptdms import TdmsFile
from pathlib import Path

labels = np.array(['p7','p1','p3','p6',
                   'p11','p12','p13','p14',
                   'p22', 'p24', 'p25', 'p15'])

filenames = ["_DTN_20240716_163119.tdms", #~0 bar G
             "_DTN_20240716_162526.tdms", #~1,15 bar G
             "_DTN_20240716_162205.tdms"] #~2 bar G]
directory = Path("C:/pomiary")
number_of_channels = len(labels)
channels_numeric = np.arange(0, number_of_channels)
channels = channels_numeric.astype(str)
gains = [99.82, 99.57, 99.70, 99.79]
means_per_file = []
ref_pressures = []
for filename in filenames:
    TDMSfilepath = directory / filename
    tdms_file = TdmsFile.read(TDMSfilepath)
    group = tdms_file['1']
    n = len(group[channels[0]])
    raw_voltages = np.array([channel[:] for channel in [group[channels[i]] for i in
    ↪  range(number_of_channels)]]).T
    raw_voltages[:, 8:12] = raw_voltages[:, 8:12] / gains
    means = np.mean(raw_voltages, axis=0)
    means_per_file.append(means)
```

```
    ref_pressure = np.mean(raw_voltages[:, 2])/0.83
    ref_pressures.append(ref_pressure)

differences = []
ref_pressure_differences = []
for i in range(1, len(means_per_file)):
    differences_col = means_per_file[i] - means_per_file[i-1]
    differences.append(differences_col)
    ref_pressure_differences.append(ref_pressures[i] - ref_pressures[i-1])

normalized_differences = []
for i in range(len(differences)):
    normalized_differences_col = differences[i] / ref_pressure_differences[i]
    normalized_differences.append(normalized_differences_col)

differences = np.array(differences)
ref_pressure_differences = np.array(ref_pressure_differences)
normalized_differences = differences / ref_pressure_differences[:, np.newaxis]
stds = np.std(normalized_differences, axis=0)
print("Weighted Means:", normalized_differences)
print("Standard Deviations:", stds)
np.save('dividers.npy', normalized_differences)
print("Execution time: {:.2f} seconds".format(time.time() - start_time))
```

Appendix G

Raw-to-Precut Processing code

```
1
2   """
3   Raw_to_Precut_0.2.py
4   Wiktor Hoffmann
5
6   """
7   import pathlib
8   import numpy as np
9   import matplotlib.pyplot as plt
10  from scipy.ndimage import uniform_filter1d
11  from scipy.signal import medfilt
12
13  BASE_DIR = pathlib.Path(r"C:/Users/hoffm/DTN/_out_20241030_1")
14  INPUT_DIR = BASE_DIR / "raw_inputs"
15  GLOB = "*_raw.npy"
16  FS = 51200.0
17  DT = 1.0 / FS
18  START_CH = 8
19  PRE_SAMPLES = 10000
20  START_GAP = 100
21  SMOOTH_WIN_S = 0.0008
22  HOLD_ABOVE_S = 0.0008
23  PRE_RISE_S = 0.0005
24  K_SIGMA = 6.0
25  HAMPEL_WIN = 11
26  HAMPEL_KSIG = 6.0
27  SAVE_PLOTS = True
28
29
30  def robust_baseline(x, frac=0.05):
31      x0 = x[:max(32, int(np.ceil(len(x)*frac)))]
32      med = np.median(x0)
33      mad = np.median(np.abs(x0-med))
34      sig = 1.4826*mad if mad > 0 else np.std(x0)
35      return med, sig
36
37
38  def detect_start(u):
```

```
    win = max(1, int(round(SMOOTH_WIN_S*FS)))
    hold = max(1, int(round(HOLD_ABOVE_S*FS)))
    pre = max(1, int(round(PRE_RISE_S*FS)))
    us = uniform_filter1d(u, size=win, mode='nearest')
    mu, sig = robust_baseline(us)
    thr = mu + K_SIGMA*sig
    above = (us > thr).astype(np.int8)
    cs = np.cumsum(above)
    n = len(us)
    for i in range(max(1, n-hold)):
        s = cs[min(n, i+hold)-1] - (cs[i-1] if i > 0 else 0)
        if s == hold and us[i] - us[max(0, i-pre)] > 0:
            return i
    k = int(np.argmax(us > thr))
    return k if us[k] > thr else 0


def hampel(x, win=HAMPEL_WIN, ksig=HAMPEL_KSIG):
    win = max(3, int(win) | 1)
    med = medfilt(x, kernel_size=win)
    dev = np.abs(x-med)
    sig = 1.4826*medfilt(dev, kernel_size=win)
    y = x.copy()
    y[dev > ksig*np.maximum(sig, 1e-12)] = med[dev > ksig*np.maximum(sig, 1e-12)]
    return y

def despike(A):
    B = A.copy()
    for c in range(B.shape[1]):
        B[:, c] = hampel(B[:, c])
    return B

def process(path):
    U = np.load(path)
    start = detect_start(U[:, START_CH])
    pre_end = max(0, start-START_GAP)
    seg0 = max(0, pre_end-PRE_SAMPLES)
    stop = min(U.shape[0], start + int(round(0.1/DT*1.3)))
    S = despike(U[seg0:stop, :])
    pre_a = 0
    pre_b = pre_end - seg0
    z = np.mean(S[pre_a:pre_b, :], axis=0) if pre_b-pre_a >= 8 else
    ↪  np.zeros(S.shape[1])
    z[:4] = 0.0
    out_s = path.with_name(path.stem + "_precut.npy")
    out_z = path.with_name(path.stem + "_pre_zerooffset.npy")
    np.save(out_s, S)
    np.save(out_z, z)
    if SAVE_PLOTS:
        t = (seg0 + np.arange(S.shape[0])) * DT
        fig, ax = plt.subplots(figsize=(12, 5))
        ax.plot(t, S[:, START_CH], label=f"p22 ch{START_CH}")
        ax.axvline(pre_end*DT, ls='--', lw=1, label="start-100")
        ax.axvline(start*DT, ls=':', lw=1, label="start")
        ax.axvline(stop*DT, ls=':', lw=1, label="stop")
```

```
93          if pre_b > 1:
94              ax.axvspan(t[pre_a], t[pre_b-1], alpha=0.18)
95          ax.set_xlabel("time [s]")
96          ax.set_ylabel("U [V]")
97          ax.grid(True)
98          ax.legend()
99          fig.tight_layout()
100         fig.savefig(path.with_name(path.stem + "_precut_preview.png"), dpi=150)
101         plt.close(fig)
102     print(f"OK: {path.name} -> {out_s.name}, {out_z.name} | start={start},
        ↪  seg=[{seg0}:{stop}]")
103
104 def main():
105     files = sorted(INPUT_DIR.glob(GLOB))
106     if not files:
107         print(f"[WARN] brak {GLOB} w {INPUT_DIR}")
108         return
109     for p in files:
110         if "_precut" in p.name or "_pre_zerooffset" in p.name:
111             continue
112         try:
113             process(p)
114         except Exception as e:
115             print(f"[ERROR] {p.name}: {e}")
116
117 if __name__ == "__main__":
118     main()
```

Appendix H

Data post-processor

```
1
2   """
3   TVC_TimeMaker_1.27.py
4   W.Hoffmann
5   """
6
7   import pathlib
8   import numpy as np
9   import matplotlib.pyplot as plt
10  from scipy.signal import savgol_filter
11  from scipy.optimize import curve_fit, least_squares
12  import csv
13
14  # ================== PATHS ==================
15  BASE_DIR    = pathlib.Path(r"C:/Users/hoffm/DTN/_out_20241030_1")
16  INPUT_DIR   = BASE_DIR
17  SUMMARY_DIR = BASE_DIR / "_analysis_from_precut" / "_npy"
18  OUT_DIR     = BASE_DIR / "_analysis_from_precut" / "_winplots"
19  OUT_DIR.mkdir(parents=True, exist_ok=True)
20
21  # ================== CONSTANTS ==================
22  FS = 51200.0
23  DT = 1.0 / FS
24
25  R  = 287.0
26  Tgas = 290.0
27  L  = 0.064 #30
28  TAU_SCALE = L / np.sqrt(R * Tgas)
29
30  PRE_SAMPLES = 10_000
31  START_GAP   = 100
32  PLOT_SKIP   = PRE_SAMPLES + START_GAP
33
34  # ================== CHANNELS ==================
35  labels = ['p7','p1','p3','p6','p11','p12','p13','p14','p22','p24','p25','p15']
36  idx = {lab: i for i, lab in enumerate(labels)}
37
38  # kanały do wykresu
```

```
WALL_CHANNELS = ['p11', 'p12', 'p13', 'p14', 'p22', 'p24', 'p25', 'p15']

# ================== CALIBRATION ==================
DIVIDERS_ACTUAL = np.array([
    0.49590, 0.83396, 0.83000, 0.49807,   # p7, p1, p3, p6
    0.00082, 0.00431, 0.00515, 0.00480,   # p11..p14
    0.01429, 0.01467, 0.01456, 0.01446    # p22,p24,p25,p15
], dtype=float)
pOt = 1.01000  # [bar] ABS

def compute_pressures_from_precut(U, zero_offsets):
    """U, zero_offsets: (N,12), (12,) w Voltach. Zwraca P [bar] ABS. Uwzględnia gain=50 dla Kulite (8..11)."""
    U = U.copy(); zero_offsets = zero_offsets.copy()
    U[:, 8:12]         = U[:, 8:12] / 50.0
    zero_offsets[8:12] = zero_offsets[8:12] / 50.0
    return (U - zero_offsets) / DIVIDERS_ACTUAL + pOt

# ================== SECONDARY MASS FLOW (orifice 4 mm) ==================
k_gas         = 1.4
T0_secondary  = 27.0 + 273  # K
D_SECONDARY   = 0.004          # m (orifice)
A_SECONDARY   = np.pi * (D_SECONDARY**2) / 4.0
Cd_secondary  = 0.925

def _mass_flux_isentropic(p0_Pa, pe_Pa, T0):
    p0_Pa = np.asarray(p0_Pa, dtype=float)
    pe_Pa = np.asarray(pe_Pa, dtype=float)
    p0_Pa, pe_Pa = np.broadcast_arrays(p0_Pa, pe_Pa)

    BetaCrit = (2/(k_gas+1))**(k_gas/(k_gas-1))
    Beta = pe_Pa / p0_Pa
    G = np.zeros_like(Beta, dtype=float)

    sub  = (Beta <= 1.0) & (Beta > BetaCrit)
    crit = (Beta <= BetaCrit)
    sup  = (Beta > 1.0)

    if np.any(sub):
        Msq = (Beta[sub]**(-(k_gas-1)/k_gas) - 1.0) * 2.0 / (k_gas - 1.0)
        G[sub] = p0_Pa[sub]*np.sqrt(k_gas/(R*T0)) * np.sqrt(Msq) * \
                 (1+(k_gas-1)*0.5*Msq)**(-(k_gas+1)/(2*(k_gas-1)))
    if np.any(crit):
        G[crit] = p0_Pa[crit]*np.sqrt(k_gas/(R*T0)) *
        ↪  ((k_gas+1)/2.0)**(-(k_gas+1)/(2.0*(k_gas-1)))
    if np.any(sup):
        Beta2 = 1.0 / Beta[sup]
        Msq = (Beta2**(-(k_gas-1)/k_gas) - 1.0) * 2.0 / (k_gas - 1.0)
        G[sup] = -pe_Pa[sup]*np.sqrt(k_gas/(R*T0)) * np.sqrt(Msq) * \
                 (1+(k_gas-1)*0.5*Msq)**(-(k_gas+1)/(2*(k_gas-1)))
    return G

def secondary_mass_flow_series(P):
    p0_bar = P[:, idx['p6']]   # ABS [bar]
    pe_bar = P[:, idx['p7']]   # ABS [bar]
```

```
    p0 = p0_bar * 1e5
    pe = pe_bar * 1e5
    G  = _mass_flux_isentropic(p0, pe, T0_secondary)   # kg/(s·m^2)
    return Cd_secondary * A_SECONDARY * G              # kg/s

# ================== ADDONS ==================
def moving_avg(x, n_win):
    n = max(1, int(round(n_win)))
    if n <= 1: return x.copy()
    kf  = np.ones(n)/n
    pad = n-1
    xp  = np.pad(x, (pad, pad), mode='edge')
    return np.convolve(xp, kf, mode='same')[pad:-pad]

def block_median_repeat(x, block):
    x = np.asarray(x, dtype=float)
    N = x.size
    if block <= 1 or N < 2:
        return x.copy()

    nblk = int(np.ceil(N / block))
    pad = nblk * block - N
    if pad > 0:
        xpad = np.pad(x, (0, pad), mode="edge")
    else:
        xpad = x

    X = xpad.reshape(nblk, block)
    m = np.median(X, axis=1)
    y = np.repeat(m, block)[:N]
    return y

def smooth_integrated_transducer(x, fs, block_ms=2.0, sg_ms=8.0, sg_poly=2):
    x = np.asarray(x, dtype=float)
    block = max(1, int(round(block_ms * 1e-3 * fs)))
    y = block_median_repeat(x, block)

    # opcjonalny Sav-Gol po de-step
    win = max(5, int(round(sg_ms * 1e-3 * fs)) | 1)
    if win >= 5 and win < len(y):
        try:
            y = savgol_filter(y, window_length=win, polyorder=sg_poly, mode="interp")
        except Exception:
            pass
    return y

def add_dimensionless_time_axis(ax):
    def to_star(x):    return x / TAU_SCALE
    def from_star(xs): return xs * TAU_SCALE
    secax = ax.secondary_xaxis('top', functions=(to_star, from_star))
    secax.set_xlabel(r"non-dimensional time")
    return secax

def find_valve_time_from_p3_voltage(U, thr_volt=0.5, delay_s=0.035):
    v = np.asarray(U[:, idx['p3']], float)
```

```
    above = v >= thr_volt
    cross = np.flatnonzero(above & np.r_[False, ~above[:-1]])
    if cross.size == 0:
        return np.nan, np.nan
    i_trig = int(cross[0])
    t_trig = i_trig * DT
    return t_trig, t_trig + delay_s

# ================== TRANSITION DETECTION ==================
TRANS_SMOOTH_S   = 0.005   # ~5 ms smoothing for detection
START_FRAC       = 0.05
RISE_HOLD_S      = 0.001
DERIV_KSIG       = 3.0
END_FRAC         = 0.02
END_HOLD_S       = 0.002
SLOPE_EPS        = 5e-4
SEARCH_GUARD_S   = 0.003
MAX_TRANS_S      = 0.050

def _mad_sigma(x):
    if x.size == 0: return 0.0
    med = np.median(x); mad = np.median(np.abs(x - med))
    return 1.4826*mad if mad>0 else float(np.std(x))

def detect_transition_window_for_channel(y_bar, w_before, w_after):
    """Zwraca (start_idx, end_idx, y_smooth) -- do detekcji okna przejścia."""
    N = len(y_bar)
    b0, b1 = w_before
    a0, a1 = w_after
    guard = int(round(SEARCH_GUARD_S * FS))

    yb = float(np.mean(y_bar[b0:b1+1]))
    ya = float(np.mean(y_bar[a0:a1+1]))
    A  = yb - ya
    if not np.isfinite(A) or abs(A) < 1e-9:
        A = max(abs(yb)*0.02, 1e-6)

    sL = min(max(b1 + 1 + guard, 0), N-1)
    sR = max(min(a0 - 1 - guard, N-1), sL)

    win = max(5, int(round(TRANS_SMOOTH_S * FS)) | 1)
    try:
        y_s = savgol_filter(y_bar, window_length=win, polyorder=2, mode='interp')
    except Exception:
        y_s = moving_avg(y_bar, win)

    dy = np.gradient(y_s, DT)
    base = dy[max(0, sL - int(0.02*FS)) : sL]
    if base.size < 16:
        base = dy[:max(16, int(0.02*FS))]
    mu  = float(np.median(base))
    sig = _mad_sigma(base)
    if sig <= 0:
        sig = max(float(np.std(base)), 1e-9)
    thr_deriv = mu - DERIV_KSIG * sig
```

```
202
203     # START
204     hold_rise = max(1, int(round(RISE_HOLD_S*FS)))
205     start = None; cnt = 0
206     for i in range(sL, sR+1):
207         cond_mag = (yb - y_s[i]) >= (START_FRAC * abs(A))
208         cond_der = dy[i] < thr_deriv
209         if cond_mag and cond_der:
210             cnt += 1
211             if cnt >= hold_rise:
212                 start = i - hold_rise + 1; break
213         else:
214             cnt = 0
215     if start is None:
216         for i in range(sL, sR+1):
217             if (yb - y_s[i]) >= (START_FRAC * abs(A)):
218                 start = i; break
219         if start is None:
220             start = sL
221
222     # KONIEC
223     hold_end = max(1, int(round(END_HOLD_S*FS)))
224     end = None; cnt = 0
225     for j in range(start, min(a0, N)):
226         cond_close = abs(y_s[j] - ya) <= (END_FRAC * abs(A))
227         cond_slope = abs(dy[j]) <= SLOPE_EPS
228         if cond_close and cond_slope:
229             cnt += 1
230             if cnt >= hold_end:
231                 end = j - hold_end + 1; break
232         else:
233             cnt = 0
234     if end is None:
235         end = min(N-1, start + int(round(MAX_TRANS_S * FS)))
236
237     end = int(min(end, start + int(round(MAX_TRANS_S*FS))))
238     return int(start), int(end), y_s
239
240 # ================== FIT: FO /w delay (curve_fit) ==================
241 FIT_SMOOTH_S = 0.006
242
243 # ===== parameteres control =====
244 PRESS_FIT_PAD_LEFT_S   = 0.001
245 PRESS_FIT_PAD_RIGHT_S  = 0.010
246 PRESS_TAU_MIN_S        = 1e-4
247 PRESS_TAU_MAX_S        = 0.10
248 PRESS_Y_FRAC_BOUND     = 0.20   # y0,yinf mogą się ruszyć o +/-20% wielkości skoku
    ↪  lub minimalny margines
249 PRESS_Y_ABS_MIN_BOUND  = 0.01   # [bar]
250 PRESS_TRANSITION_BOOST = 3.0    # wagi residuali w pobliżu przejścia
251 PRESS_TRANSITION_HALF_WINDOW_S = 0.004
252 SOFTPLUS_EPS = 1e-12
253
254 def _softplus(x):
255     x = np.asarray(x, dtype=float)
```

```
256     return np.log1p(np.exp(-np.abs(x))) + np.maximum(x, 0.0)
257
258 def _inv_softplus(y):
259     y = np.asarray(y, dtype=float)
260     return np.where(y > 30.0, y, np.log(np.expm1(np.maximum(y, SOFTPLUS_EPS))))
261
262 def fit_first_order_with_delay(t, y_s_det, w_before, w_after, start_idx, end_idx,
    ↪  t0_lower_bound=None):
263     b0, b1 = w_before
264     a0, a1 = w_after
265     y0_ref   = float(np.mean(y_s_det[b0:b1+1]))
266     yinf_ref = float(np.mean(y_s_det[a0:a1+1]))
267     K_ref    = yinf_ref - y0_ref
268
269     if not (np.isfinite(y0_ref) and np.isfinite(yinf_ref)):
270         return {"ok": False}
271     if abs(K_ref) < 1e-6:
272         return {"ok": False}
273
274     jL = int(max(start_idx - int(PRESS_FIT_PAD_LEFT_S * FS), 0))
275     jR = int(min(end_idx   + int(PRESS_FIT_PAD_RIGHT_S * FS), len(t)-1))
276
277     if t0_lower_bound is not None and np.isfinite(t0_lower_bound):
278         i_lb = int(np.searchsorted(t, float(t0_lower_bound), side='left'))
279         jR = max(jR, min(len(t)-1, i_lb + int(PRESS_FIT_PAD_RIGHT_S * FS)))
280
281     if jR - jL < 20:
282         return {"ok": False}
283
284     t_seg = t[jL:jR+1]
285     win_fit = max(5, int(round(FIT_SMOOTH_S * FS)) | 1)
286     y_raw_seg = y_s_det[jL:jR+1]
287     try:
288         y_seg = savgol_filter(y_raw_seg, window_length=win_fit, polyorder=2,
        ↪  mode='interp')
289     except Exception:
290         y_seg = moving_avg(y_raw_seg, win_fit)
291
292     # model
293     def fo_delay_model(ts, y0, yinf, t0, tau_resp):
294         tau_eff = np.maximum(float(tau_resp), PRESS_TAU_MIN_S)
295         ts = np.asarray(ts, dtype=float)
296         return np.where(ts < t0, y0, yinf + (y0 - yinf) * np.exp(-(ts - t0) /
        ↪  tau_eff))
297
298     t0_guess = float(t[start_idx])
299     if t0_lower_bound is not None and np.isfinite(t0_lower_bound):
300         t0_guess = max(t0_guess, float(t0_lower_bound) + 2.0 * DT)
301     tau_guess = max(PRESS_TAU_MIN_S, 0.25 * max(t[end_idx] - t[start_idx], 5.0 * DT))
302
303     amp_ref = max(abs(K_ref), PRESS_Y_ABS_MIN_BOUND)
304     y_margin = max(PRESS_Y_FRAC_BOUND * amp_ref, PRESS_Y_ABS_MIN_BOUND)
305
306     y0_min = y0_ref - y_margin
307     y0_max = y0_ref + y_margin
```

```
    yinf_min = yinf_ref - y_margin
    yinf_max = yinf_ref + y_margin

    t_min = float(t_seg[0])
    t_max = float(t_seg[-1])

    w = np.ones_like(t_seg, dtype=float)
    center = t0_guess
    halfwin = PRESS_TRANSITION_HALF_WINDOW_S
    trans_mask = np.abs(t_seg - center) <= halfwin
    w[trans_mask] = PRESS_TRANSITION_BOOST

    if t0_lower_bound is not None and np.isfinite(t0_lower_bound):
        lb = float(t0_lower_bound)

        alpha_guess = float(_inv_softplus(max(t0_guess - lb, 2.0 * DT)))

        def unpack(p):
            y0, yinf, alpha, log_tau = p
            t0 = lb + float(_softplus(alpha))
            tau_resp = float(np.exp(log_tau))
            return y0, yinf, t0, tau_resp

        def residuals(p):
            y0, yinf, t0, tau_resp = unpack(p)
            y_hat = fo_delay_model(t_seg, y0, yinf, t0, tau_resp)
            return np.sqrt(w) * (y_hat - y_seg)

        x0 = np.array([y0_ref, yinf_ref, alpha_guess, np.log(tau_guess)],
        ↪  dtype=float)

        t0_upper_eff = max(lb + 2.0 * DT, t_max - 2.0 * DT)
        alpha_min = float(_inv_softplus(1.0 * DT))
        alpha_max = float(_inv_softplus(max(t0_upper_eff - lb, 2.0 * DT)))

        lower = np.array([y0_min, yinf_min, alpha_min, np.log(PRESS_TAU_MIN_S)],
        ↪  dtype=float)
        upper = np.array([y0_max, yinf_max, alpha_max, np.log(PRESS_TAU_MAX_S)],
        ↪  dtype=float)

        try:
            res = least_squares(
                residuals, x0, bounds=(lower, upper),
                method='trf', loss='soft_l1', f_scale=max(0.01 * amp_ref, 1e-4),
                max_nfev=30000
            )
        except Exception:
            return {"ok": False}

        if not res.success:
            return {"ok": False}

        y0_fit, yinf_fit, t0_fit, tau_fit = unpack(res.x)

    else:
```

```
360         def residuals(p):
361             y0, yinf, t0, log_tau = p
362             tau_resp = float(np.exp(log_tau))
363             y_hat = fo_delay_model(t_seg, y0, yinf, t0, tau_resp)
364             return np.sqrt(w) * (y_hat - y_seg)
365 
366         x0 = np.array([y0_ref, yinf_ref, t0_guess, np.log(tau_guess)], dtype=float)
367         lower = np.array([y0_min, yinf_min, t_min, np.log(PRESS_TAU_MIN_S)],
            ↪ dtype=float)
368         upper = np.array([y0_max, yinf_max, t_max, np.log(PRESS_TAU_MAX_S)],
            ↪ dtype=float)
369 
370         try:
371             res = least_squares(
372                 residuals, x0, bounds=(lower, upper),
373                 method='trf', loss='soft_l1', f_scale=max(0.01 * amp_ref, 1e-4),
374                 max_nfev=30000
375             )
376         except Exception:
377             return {"ok": False}
378 
379         if not res.success:
380             return {"ok": False}
381 
382         y0_fit, yinf_fit, t0_fit, log_tau_fit = res.x
383         tau_fit = float(np.exp(log_tau_fit))
384 
385     if not (np.isfinite(y0_fit) and np.isfinite(yinf_fit) and np.isfinite(t0_fit) and
        ↪ np.isfinite(tau_fit)):
386         return {"ok": False}
387     if tau_fit <= 0:
388         return {"ok": False}
389 
390     y_hat = fo_delay_model(t_seg, y0_fit, yinf_fit, t0_fit, tau_fit)
391 
392     ss_res = float(np.sum((y_seg - y_hat)**2))
393     ss_tot = float(np.sum((y_seg - np.mean(y_seg))**2))
394     r2 = 1.0 - ss_res / ss_tot if ss_tot > 0 else np.nan
395     t63 = t0_fit + tau_fit
396     K_fit = yinf_fit - y0_fit
397 
398     return {
399         "ok": True,
400         "y0": float(y0_fit), "yinf": float(yinf_fit), "K": float(K_fit),
401         "t0": float(t0_fit), "tau_resp": float(tau_fit), "t63": float(t63),
402         "r2": float(r2),
403         "fit_t": t_seg, "fit_y": y_hat
404     }
405 
406 def fit_secondary_mass_flow_with_delay(t, smf_series, w_after, start_idx, end_idx):
407     a0, a1 = w_after
408     ws_inf = float(np.mean(smf_series[a0:a1+1]))
409 
410     if not np.isfinite(ws_inf) or ws_inf <= 1e-9:
411         return {"ok": False}
```

```
412
413     jL = int(max(start_idx - int(0.001 * FS), 0))
414     jR = int(min(end_idx   + int(0.006 * FS), len(t) - 1))
415     if jR - jL < 20:
416         return {"ok": False}
417
418     t_seg = t[jL:jR+1]
419     y_seg_raw = smf_series[jL:jR+1]
420
421     win_fit = max(5, int(round(FIT_SMOOTH_S * FS)) | 1)
422     try:
423         y_seg = savgol_filter(y_seg_raw, window_length=win_fit, polyorder=2,
        ↪  mode='interp')
424     except Exception:
425         y_seg = moving_avg(y_seg_raw, win_fit)
426
427     def fo_ws_model(ts, t0, T):
428         T_eff = np.maximum(T, 1e-6)
429         ts = np.asarray(ts)
430         return np.where(ts < t0, 0.0, ws_inf * (1.0 - np.exp(-(ts - t0) / T_eff)))
431
432     t0_guess = float(t[start_idx])
433     T_guess  = max(1e-4, 0.25 * (t[end_idx] - t[start_idx]))
434
435     t0_min = float(t[jL])
436     t0_max = float(t[jR])
437     T_min  = 1e-4
438     T_max  = 0.1
439
440     try:
441         popt, _ = curve_fit(
442             fo_ws_model,
443             t_seg,
444             y_seg,
445             p0=[t0_guess, T_guess],
446             bounds=([t0_min, T_min], [t0_max, T_max]),
447             maxfev=20000
448         )
449     except Exception:
450         return {"ok": False}
451
452     t0_fit, T_fit = float(popt[0]), float(popt[1])
453     if not np.isfinite(t0_fit) or not np.isfinite(T_fit) or T_fit <= 0:
454         return {"ok": False}
455
456     y_hat = fo_ws_model(t_seg, t0_fit, T_fit)
457
458     ss_res = float(np.sum((y_seg - y_hat) ** 2))
459     ss_tot = float(np.sum((y_seg - np.mean(y_seg)) ** 2))
460     r2 = 1.0 - ss_res / ss_tot if ss_tot > 0 else np.nan
461
462     t63 = t0_fit + T_fit
463
464     return {
465         "ok": True,
```

```
        "ws_inf": ws_inf,
        "t0": t0_fit,
        "tau_resp": T_fit,
        "t63": t63,
        "r2": float(r2),
        "fit_t": t_seg,
        "fit_y": y_hat
    }

# ================== PLOT: FO-fit ==================
def plot_windows_grid(t, P, w_before, w_after, t_valve, summary,
                      smf_series, stem, out_png, zoom=False):
    n_ch = len(WALL_CHANNELS)
    fig_h = max(3.0 * n_ch + 4.5, 20)
    fig, axes = plt.subplots(
        n_ch + 1, 1, figsize=(16, fig_h), sharex=True,
        gridspec_kw={'height_ratios': [1.25] + [1.0] * n_ch}
    )
    fig.subplots_adjust(left=0.08, right=0.94, top=0.98, bottom=0.06, hspace=0.24)

    ax_ws = axes[0]
    pressure_axes = axes[1:]

    i0 = min(PLOT_SKIP, len(t)-1)
    t_plot = t[i0:]
    smf_plot = smf_series[i0:]

    def clamp_time(x):
        if not np.isfinite(x):
            return t_plot[0]
        return max(t_plot[0], min(t_plot[-1], float(x)))

    ref_lab = 'p15' if 'p15' in WALL_CHANNELS else WALL_CHANNELS[-1]
    y_ref = P[:, idx[ref_lab]].astype(float)
    start_ref, end_ref, _ = detect_transition_window_for_channel(y_ref, w_before,
    ↪  w_after)
    fit_ws = fit_secondary_mass_flow_with_delay(t, smf_series, w_after, start_ref,
    ↪  end_ref)
    t0_lower_bound_press = fit_ws["t0"] if fit_ws.get("ok", False) else None

    # ---------- top panel: ws only ----------
    ax_ws.plot(t_plot, smf_plot, lw=1.0, color='tab:red', alpha=0.75,
    ↪  label='Secondary mass flow')

    if fit_ws.get("ok", False):
        tt_ws = fit_ws["fit_t"]
        yy_ws = fit_ws["fit_y"]
        mask_ws = (tt_ws >= t_plot[0]) & (tt_ws <= t_plot[-1])
        ax_ws.plot(tt_ws[mask_ws], yy_ws[mask_ws], color='#E78BB2', lw=1.5,
        ↪  alpha=0.95,
                   label='ws FO-delay fit')

        t0_ws  = clamp_time(fit_ws["t0"])
        t63_ws = clamp_time(fit_ws["t63"])
        ax_ws.axvline(t0_ws, color='#E78BB2', linestyle=':', lw=1.1)
```

```
517         ax_ws.axvline(t63_ws, color='#E78BB2', linestyle=':', lw=1.1, alpha=0.9)
518
519         ws_low, ws_high = ax_ws.get_ylim()
520         yb_ws = ws_high - 0.10 * (ws_high - ws_low)
521         hb_ws = 0.02 * (ws_high - ws_low)
522         ax_ws.plot([t0_ws, t63_ws], [yb_ws, yb_ws], color='#E78BB2', lw=1.1,
            ↪  zorder=6, clip_on=True)
523         ax_ws.plot([t0_ws, t0_ws], [yb_ws - hb_ws, yb_ws + hb_ws], color='#E78BB2',
            ↪  lw=1.4, zorder=6, clip_on=True)
524         ax_ws.plot([t63_ws, t63_ws], [yb_ws - hb_ws, yb_ws + hb_ws], color='#E78BB2',
            ↪  lw=1.4, zorder=6, clip_on=True)
525         ax_ws.text(0.5 * (t0_ws + t63_ws), yb_ws + 1.2 * hb_ws, r"$\tau_{ws}$",
526                    color='#E78BB2', fontsize=13, ha='center', va='bottom', zorder=7,
                       ↪  clip_on=True)
527
528         dt_tau_ws = (fit_ws["t63"] - fit_ws["t0"]) / TAU_SCALE
529         box_txt_ws = (
530             fr"$\tau^* = {dt_tau_ws:.3f}$"
531         )
532         ax_ws.text(0.985, 0.92, box_txt_ws, transform=ax_ws.transAxes,
533                    color='#E78BB2', fontsize=10, va='top', ha='right',
534                    bbox=dict(boxstyle="round,pad=0.30", fc='#ffffff', ec='#E78BB2',
                       ↪  lw=1.1, alpha=1.0))
535
536     if zoom:
537         ax_ws.set_xlim(0.0, 0.03)
538     else:
539         t_star_max = 1000.0
540         t_max = t_star_max * TAU_SCALE
541         ax_ws.set_xlim(t_plot[0], min(t_plot[-1], t_max))
542
543     ax_ws.grid(True, alpha=0.35)
544     ax_ws.set_ylabel("ws\n[kg/s]")
545     ax_ws.legend(loc="upper left", fontsize=9)
546
547     # ---------- pressure panels ----------
548     table_rows = []
549     for iax, (ax, lab) in enumerate(zip(pressure_axes, WALL_CHANNELS)):
550         y = P[:, idx[lab]].astype(float)
551         start, end, y_s_det = detect_transition_window_for_channel(y, w_before,
            ↪  w_after)
552
553         y_plot  = y[i0:]
554         ys_plot = y_s_det[i0:]
555
556         b0, b1 = w_before
557         a0, a1 = w_after
558         yb = float(np.mean(y[b0:b1+1]))
559         ya = float(np.mean(y[a0:a1+1]))
560
561         ax.plot(t_plot, y_plot, lw=0.8, alpha=0.25, zorder=2, label=f"{lab} raw")
562         ax.plot(t_plot, ys_plot, lw=1.25, zorder=3, label=f"{lab} smoothed")
563         ax.axhline(yb, color='tab:blue', linestyle='--', lw=0.9, alpha=0.7,
            ↪  label='_nolegend_')
```

```
564         ax.axhline(ya, color='tab:orange', linestyle='--', lw=0.9, alpha=0.7,
            ↪  label='_nolegend_')
565
566         if np.isfinite(t_valve):
567             ax.axvline(clamp_time(t_valve), color='royalblue', linestyle='--',
                ↪  lw=1.1)
568
569         fit = fit_first_order_with_delay(
570             t, y_s_det, w_before, w_after, start, end,
571             t0_lower_bound=t0_lower_bound_press
572         )
573         if fit.get("ok", False):
574             tt = fit["fit_t"]
575             yy = fit["fit_y"]
576             mask = (tt >= t_plot[0]) & (tt <= t_plot[-1])
577             ax.plot(tt[mask], yy[mask], color='tab:purple', lw=2.6, alpha=1.0,
                    ↪  zorder=8, label='Pressure FO-delay fit')
578
579             t0c  = clamp_time(fit["t0"])
580             t63c = clamp_time(fit["t63"])
581             ax.axvline(t0c,  color='tab:purple', linestyle=':', lw=1.2)
582             ax.axvline(t63c, color='tab:purple', linestyle=':', lw=1.2, alpha=0.9)
583
584             dt_tau = (fit["t63"] - fit["t0"]) / TAU_SCALE
585
586             if zoom:
587                 x0_tmp, x1_tmp = 0.0, 0.03
588                 m_zoom_tmp = (t_plot >= x0_tmp) & (t_plot <= x1_tmp)
589                 y_for_bracket = y_plot[m_zoom_tmp] if np.any(m_zoom_tmp) else y_plot
590                 y_low = float(np.nanmin(y_for_bracket))
591                 y_high = float(np.nanmax(y_for_bracket))
592                 y_span_tmp = y_high - y_low
593                 if not np.isfinite(y_span_tmp) or y_span_tmp <= 0:
594                     y_span_tmp = max(abs(y_low), 1.0) * 0.02
595                 y_low = y_low - 0.10 * y_span_tmp
596                 y_high = y_high + 0.10 * y_span_tmp
597             else:
598                 y_low, y_high = ax.get_ylim()
599             y_bracket = y_high - 0.10 * (y_high - y_low)
600             h_bracket = 0.02 * (y_high - y_low)
601
602             ax.plot([t0c, t63c], [y_bracket, y_bracket],
603                     color='tab:purple', lw=1.1, zorder=6, clip_on=True)
604             ax.plot([t0c, t0c], [y_bracket - h_bracket, y_bracket + h_bracket],
605                     color='tab:purple', lw=1.5, zorder=6, clip_on=True)
606             ax.plot([t63c, t63c], [y_bracket - h_bracket, y_bracket + h_bracket],
607                     color='tab:purple', lw=1.5, zorder=6, clip_on=True)
608             ax.text(0.5 * (t0c + t63c), y_bracket + 1.0 * h_bracket, r"$\tau$",
609                     color='tab:purple', fontsize=12, ha='center', va='bottom',
                        ↪  zorder=7, clip_on=True)
610
611             box_txt = (
612                 f"{lab}\n"
613                 f"\tau*={dt_tau:.3f}"
614             )
```

```
            ax.text(0.985, 0.84, box_txt, transform=ax.transAxes,
                    color='tab:purple', fontsize=9, va='top', ha='right',
                    bbox=dict(boxstyle="round,pad=0.25", fc='#ffffff',
                    ↪  ec='tab:purple', lw=1.0, alpha=1.0))

            table_rows.append({
                "channel": lab,
                "pinf_minus_p0_bar": fit["K"],
                "t0_s": fit["t0"],
                "tau_s": fit["tau_resp"],
                "t63_s": fit["t63"],
                "tau_star": dt_tau,
                "r2": fit["r2"],
            })
        else:
            table_rows.append({
                "channel": lab,
                "pinf_minus_p0_bar": np.nan,
                "t0_s": np.nan,
                "tau_s": np.nan,
                "t63_s": np.nan,
                "tau_star": np.nan,
                "r2": np.nan,
            })

        if zoom:
            ax.set_xlim(0.0, 0.03)

            x0, x1 = ax.get_xlim()
            m_zoom = (t_plot >= x0) & (t_plot <= x1)
            y_for_ylim = y_plot[m_zoom] if np.any(m_zoom) else y_plot

            y_min = float(np.nanmin(y_for_ylim))
            y_max = float(np.nanmax(y_for_ylim))
            y_span = y_max - y_min

            if not np.isfinite(y_span) or y_span <= 0:
                y_span = max(abs(y_min), 1.0) * 0.02

            pad = 0.10 * y_span
            ax.set_ylim(y_min - pad, y_max + pad)
        else:
            t_star_max = 1000.0
            t_max = t_star_max * TAU_SCALE
            ax.set_xlim(t_plot[0], min(t_plot[-1], t_max))

        ax.grid(True, alpha=0.35)
        ax.set_ylabel(f"{lab}\n[bar]")
        if iax == 0:
            ax.legend(loc="upper left", fontsize=8)

    pressure_axes[-1].set_xlabel("time [s]")
    add_dimensionless_time_axis(ax_ws)

    table_path = pathlib.Path(out_png).with_suffix('.csv')
```

```
669     with open(table_path, "w", newline="") as f:
670         w = csv.writer(f)
671         w.writerow(["section", "channel", "pinf_minus_p0_bar", "t0_s", "tau_s",
            ↪  "t63_s", "tau_star", "r2", "ws_inf_kg_s"])
672         if fit_ws.get("ok", False):
673             dt_tau_ws = (fit_ws["t63"] - fit_ws["t0"]) / TAU_SCALE
674             w.writerow(["ws", "ws", "", fit_ws["t0"], fit_ws["tau_resp"],
                ↪  fit_ws["t63"], dt_tau_ws, fit_ws["r2"], fit_ws["ws_inf"]])
675         else:
676             w.writerow(["ws", "ws", "", np.nan, np.nan, np.nan, np.nan, np.nan,
                ↪  np.nan])
677 
678         for row in table_rows:
679             w.writerow([
680                 "pressure", row["channel"], row["pinf_minus_p0_bar"], row["t0_s"],
681                 row["tau_s"], row["t63_s"], row["tau_star"], row["r2"], ""
682             ])
683 
684     fig.savefig(out_png, dpi=150, bbox_inches="tight")
685     plt.close(fig)
686 
687 # ================== PLOT: RAW ==================
688 SUMMARY_CSV = OUT_DIR / "_summary_table.csv"
689 
690 if not SUMMARY_CSV.exists():
691     with open(SUMMARY_CSV, "w", newline="") as f:
692         w = csv.writer(f)
693         w.writerow([
694             "file",
695             "primary_before_kg_s", "primary_after_kg_s",
696             "secondary_before_kg_s", "secondary_after_kg_s",
697             "secondary_percentage",
698             "NPR_before", "NPR_after"
699         ])
700 
701 def plot_raw_signals(t, P, smf_series, stem, out_png):
702     """Wykresy surowych ciśnień p25/p15 i secondary mass flow."""
703     fig, axes = plt.subplots(2, 1, figsize=(10, 7), sharex=True)
704     fig.subplots_adjust(left=0.08, right=0.92, top=0.90, bottom=0.12, hspace=0.35)
705 
706     i0 = min(PLOT_SKIP, len(t)-1)
707     t_plot   = t[i0:]
708     smf_plot = smf_series[i0:]
709 
710     for ax, lab in zip(axes.flat, WALL_CHANNELS):
711         y = P[:, idx[lab]]
712         y_plot = y[i0:]
713 
714         ax.plot(t_plot, y_plot, lw=1.0, color='tab:blue', alpha=0.8, label=f"{lab}
            ↪  raw")
715         ax.set_ylabel("pressure [bar abs]")
716         ax.grid(True, alpha=0.35)
717 
718         ax2 = ax.twinx()
719         ax2.patch.set_visible(False)
```

```
        ax2.plot(
            t_plot, smf_plot,
            lw=1.4,
            color='tab:red',
            alpha=0.9,
            label='Secondary mass flow'
        )
        ax2.set_ylabel("secondary mass flow [kg/s]", color='tab:red')
        ax2.tick_params(axis='y', labelcolor="tab:red")

        h1, l1 = ax.get_legend_handles_labels()
        h2, l2 = ax2.get_legend_handles_labels()
        ax.legend(h1+h2, l1+l2, fontsize=9, loc="upper left")

    ax.set_xlabel("time [s]")
    fig.savefig(out_png, dpi=150, bbox_inches="tight")
    plt.close(fig)

def plot_secondary_mass_flow_fit(t, smf_series, w_after, stem, out_png, zoom=False):
    fig, ax = plt.subplots(1, 1, figsize=(10, 5))
    fig.subplots_adjust(left=0.10, right=0.95, top=0.88, bottom=0.16)

    i0 = min(PLOT_SKIP, len(t) - 1)
    t_plot = t[i0:]
    ws_plot = smf_series[i0:]

    try:
        ws_smooth = savgol_filter(
            smf_series,
            window_length=max(5, int(round(FIT_SMOOTH_S * FS)) | 1),
            polyorder=2,
            mode='interp'
        )
    except Exception:
        ws_smooth = moving_avg(smf_series, max(5, int(round(FIT_SMOOTH_S * FS))))

    i_start_search = np.searchsorted(t, 0.0)
    i_end_search   = min(len(t) - 1, i_start_search + int(0.05 * FS))

    # próg 5% wartości końcowej
    a0, a1 = w_after
    ws_inf_est = float(np.mean(smf_series[a0:a1+1]))
    if np.isfinite(ws_inf_est) and ws_inf_est > 1e-9:
        thr = 0.05 * ws_inf_est
        above = np.where(ws_smooth[i_start_search:i_end_search] > thr)[0]
        if above.size:
            start_idx = i_start_search + int(above[0])
        else:
            start_idx = i_start_search
    else:
        start_idx = i_start_search

    end_idx = min(len(t) - 1, start_idx + int(0.04 * FS))

```

```
    fit = fit_secondary_mass_flow_with_delay(t, smf_series, w_after, start_idx,
    ↪  end_idx)

    ax.plot(t_plot, ws_plot, lw=1.0, alpha=0.35, color='tab:red', label='ws raw')
    ax.plot(t_plot, ws_smooth[i0:], lw=1.5, color='tab:orange', label='ws smoothed')

    if fit.get("ok", False):
        tt = fit["fit_t"]
        yy = fit["fit_y"]
        mask = (tt >= t_plot[0]) & (tt <= t_plot[-1])
        ax.plot(tt[mask], yy[mask], lw=2.0, color='tab:purple', label='ws FO-delay
        ↪  fit')

        t0c  = fit["t0"]
        t63c = fit["t63"]
        ax.axvline(t0c,  color='tab:purple', linestyle=':', lw=1.3)
        ax.axvline(t63c, color='tab:purple', linestyle=':', lw=1.1, alpha=0.8)

        dt_tau = (fit["t63"] - fit["t0"]) / TAU_SCALE

        # bracket showing tau between t0 and t63 for ws fit
        ymin, ymax = ax.get_ylim()
        y_bracket = ymax - 0.08 * (ymax - ymin)
        h_bracket = 0.015 * (ymax - ymin)

        ax.plot([t0c, t63c], [y_bracket, y_bracket],
                color='tab:purple', lw=1.2, zorder=6)
        ax.plot([t0c, t0c], [y_bracket - h_bracket, y_bracket + h_bracket],
                color='tab:purple', lw=1.2, zorder=6)
        ax.plot([t63c, t63c], [y_bracket - h_bracket, y_bracket + h_bracket],
                color='tab:purple', lw=1.2, zorder=6)
        ax.text(0.5 * (t0c + t63c), y_bracket + 1.5 * h_bracket, r"$\tau$",
                color='tab:purple', fontsize=12, ha='center', va='bottom', zorder=7)

        txt = (
            f"ws_inf={fit['ws_inf']:.5f} kg/s\n"
            f"t0={fit['t0']:.5f}s\n"
            f"\u03c4={fit['tau_resp']:.5f}s\n"
            f"t63={fit['t63']:.5f}s\n"
            f"\Delta\tau={dt_tau:.4f}\n"
        #    f"R²={fit['r2']:.2f}"
        )
        ax.text(
            0.98, 0.98, txt,
            transform=ax.transAxes,
            color='tab:purple',
            fontsize=9,
            va='top', ha='right',
            bbox=dict(boxstyle="round,pad=0.35",
                      fc='#ffffff', ec='tab:purple', lw=1.2, alpha=1.0),
            zorder=1000,
            clip_on=False
        )

    if zoom:
```

```
827         ax.set_xlim(0.04, 0.06)
828     else:
829         t_star_max = 1000.0
830         t_max = t_star_max * TAU_SCALE
831         ax.set_xlim(t_plot[0], min(t_plot[-1], t_max))
832 
833     ax.grid(True, alpha=0.35)
834     ax.set_xlabel("time [s]")
835     ax.set_ylabel("secondary mass flow [kg/s]")
836     ax.set_title(f"{stem} -- ws FO-delay fit" + (" (zoom 0.04-0.06 s)" if zoom else
        ↪  ""))
837 
838     add_dimensionless_time_axis(ax)
839     ax.legend(loc="upper left", fontsize=8)
840 
841     fig.savefig(out_png, dpi=150, bbox_inches="tight")
842     plt.close(fig)
843 
844 # ================== FFT: p15 before/after ==================
845 def compute_single_sided_fft(y, fs):
846     y = np.asarray(y, dtype=float)
847     if y.size < 2:
848         return np.array([]), np.array([])
849     y = y - np.mean(y)
850     win = np.hanning(y.size)
851     yw = y * win
852     spec = np.fft.rfft(yw)
853     freq = np.fft.rfftfreq(y.size, d=1.0/fs)
854     amp = (2.0 / np.sum(win)) * np.abs(spec)
855     if amp.size:
856         amp[0] *= 0.5
857         if y.size % 2 == 0 and amp.size > 1:
858             amp[-1] *= 0.5
859     return freq, amp
860 
861 def plot_fft_before_after_p15(t, P, w_before, w_after, stem, out_png):
862     lab = 'p15'
863     y = P[:, idx[lab]].astype(float)
864     b0, b1 = map(int, w_before)
865     a0, a1 = map(int, w_after)
866 
867     y_before = y[b0:b1+1]
868     y_after  = y[a0:a1+1]
869     f_b, A_b = compute_single_sided_fft(y_before, FS)
870     f_a, A_a = compute_single_sided_fft(y_after, FS)
871 
872     fig, ax = plt.subplots(1, 1, figsize=(10, 5))
873     fig.subplots_adjust(left=0.10, right=0.96, top=0.90, bottom=0.14)
874 
875     line_b, = ax.plot(f_b, A_b, lw=1.2, label='p15 BEFORE')
876     line_a, = ax.plot(f_a, A_a, lw=1.2, label='p15 AFTER')
877 
878     # peak markers only within 0-1000 Hz
879     m_b = f_b <= 10000.0
880     m_a = f_a <= 10000.0
```

```
881
882     if np.any(m_b):
883         f_b_lim = f_b[m_b]
884         A_b_lim = A_b[m_b]
885         i_b = int(np.argmax(A_b_lim))
886         fpk_b = float(f_b_lim[i_b])
887         Apk_b = float(A_b_lim[i_b])
888         ax.axvline(fpk_b, color=line_b.get_color(), lw=0.8, alpha=0.9)
889         ax.text(
890             fpk_b + 8.0, Apk_b, f'{fpk_b:.1f} Hz',
891             color=line_b.get_color(), fontsize=9,
892             ha='left', va='bottom'
893         )
894
895     if np.any(m_a):
896         f_a_lim = f_a[m_a]
897         A_a_lim = A_a[m_a]
898         i_a = int(np.argmax(A_a_lim))
899         fpk_a = float(f_a_lim[i_a])
900         Apk_a = float(A_a_lim[i_a])
901         ax.axvline(fpk_a, color=line_a.get_color(), lw=0.8, alpha=0.9)
902         ax.text(
903             fpk_a + 8.0, Apk_a, f'{fpk_a:.1f} Hz',
904             color=line_a.get_color(), fontsize=9,
905             ha='left', va='bottom'
906         )
907
908     ax.set_xlim(0, 10000)
909     ax.set_xlabel('frequency [Hz]')
910     ax.set_ylabel('FFT amplitude [bar]')
911     #ax.set_title(f'{stem} -- FFT before/after for p15')
912     ax.grid(True, alpha=0.35)
913     ax.legend(loc='upper right')
914     fig.savefig(out_png, dpi=150, bbox_inches='tight')
915     plt.close(fig)
916
917 # ================== MAIN LOOP ==================
918 def process_pair(precut_path: pathlib.Path):
919     stem = precut_path.stem
920     off_path     = precut_path.with_name(stem.replace("_precut", "_pre_zerooffset") +
    ↪  ".npy")
921     summary_path = SUMMARY_DIR / f"{stem}_summary.npy"
922
923     if not summary_path.exists():
924         print(f"[WARN] Brak summary dla {stem}, pomijam."); return
925     if not off_path.exists():
926         print(f"[WARN] Brak zerooffset dla {stem}, pomijam."); return
927
928     U    = np.load(precut_path)
929     zero = np.load(off_path).astype(float)
930     if U.ndim != 2 or U.shape[1] != 12:
931         print(f"[ERROR] {precut_path.name}: oczekuję (N,12), mam {U.shape}"); return
932     if zero.shape[0] != 12:
933         print(f"[ERROR] {off_path.name}: oczekuję (12,), mam {zero.shape}"); return
934
```

```
    summary   = np.load(summary_path, allow_pickle=True).item()
    w_before  = tuple(summary["windows"]["before_idx"])
    w_after   = tuple(summary["windows"]["after_idx"])
    mf        = summary.get("mass_flows", {})

    # ciśnienia i czas (z t=0 @ p3==0.5V)
    P = compute_pressures_from_precut(U, zero)
    P_for_ws = P.copy()
    P_for_ws[:, idx['p6']] = smooth_integrated_transducer(P[:, idx['p6']], FS,
    ↪  block_ms=0.5, sg_ms=8.0)
    P_for_ws[:, idx['p7']] = smooth_integrated_transducer(P[:, idx['p7']], FS,
    ↪  block_ms=0.5, sg_ms=8.0)

    t = np.arange(P.shape[0]) * DT
    t_trig, t_valve_abs = find_valve_time_from_p3_voltage(U)
    SHIFT_T = 0.035  # s

    if np.isfinite(t_trig):
        t = t - t_trig - SHIFT_T
        t_valve = t_valve_abs - t_trig - SHIFT_T
    else:
        t = t - SHIFT_T
        t_valve = np.nan


    smf_series = secondary_mass_flow_series(P_for_ws)

    out_png_zoom = OUT_DIR / f"{stem}_p11_to_p15_FOdelay_smf_cf_zoom.png"
    plot_windows_grid(t, P, w_before, w_after, t_valve,
                      summary, smf_series, stem, out_png_zoom, zoom=True)
    print(f"[OK] {out_png_zoom.name}")

    # === summary CSV ===
    pb = mf.get("primary_before_kg_s", np.nan)
    pa = mf.get("primary_after_kg_s", np.nan)
    sb = mf.get("secondary_before_kg_s", np.nan)
    sa = mf.get("secondary_after_kg_s", np.nan)
    percent = sa / (pa + sa) * 100 if np.isfinite(pa) and np.isfinite(sa) and (pa+sa)
    ↪  != 0 else np.nan

    p3_info = summary.get("p3_bar", {})
    p3_before = float(p3_info.get("mean_before", np.nan))
    p3_after  = float(p3_info.get("mean_after",  np.nan))
    NPR_before = p3_before / pOt if np.isfinite(p3_before) else np.nan
    NPR_after  = p3_after  / pOt if np.isfinite(p3_after)  else np.nan

    with open(SUMMARY_CSV, "a", newline="") as f:
        w = csv.writer(f)
        w.writerow([
            stem,
            pb, pa,
            sb, sa,
            percent,
            NPR_before, NPR_after
        ])
```

```
987
988 def main():
989     #files = sorted(INPUT_DIR.glob("*_precut.npy"))
990     files = sorted(INPUT_DIR.glob("_DTN_20241030_145052*_precut.npy"))
991     if not files:
992         print(f"[WARN] Brak plików *_precut.npy w {INPUT_DIR}")
993         return
994     for f in files:
995         try:
996             process_pair(f)
997         except Exception as e:
998             print(f"[ERROR] {f.name}: {e}")
999
1000 if __name__ == "__main__":
1001     main()
```

Appendix I

FFT Analysis

```
"""
TVC_FFT_AllChannels_0.2.py

"""

import pathlib
import numpy as np
import matplotlib.pyplot as plt

# ================== PATHS ==================
BASE_DIR    = pathlib.Path(r"C:/Users/hoffm/DTN/_out_20241030_1")
INPUT_DIR   = BASE_DIR
SUMMARY_DIR = BASE_DIR / "_analysis_from_precut" / "_npy"
OUT_DIR     = BASE_DIR / "_analysis_from_precut" / "_winplots"
OUT_DIR.mkdir(parents=True, exist_ok=True)

# ================== CONSTANTS ==================
FS = 51200.0
DT = 1.0 / FS

# ================== CHANNELS==================
labels = ['p7','p1','p3','p6','p11','p12','p13','p14','p22','p24','p25','p15']
idx = {lab: i for i, lab in enumerate(labels)}

WALL_CHANNELS = ['p11', 'p12', 'p13', 'p14', 'p22', 'p24', 'p25', 'p15']

# ================== CALIBRATION ==================
DIVIDERS_ACTUAL = np.array([
    0.49590, 0.83396, 0.83000, 0.49807,   # p7, p1, p3, p6
    0.00082, 0.00431, 0.00515, 0.00480,   # p11..p14
    0.01429, 0.01467, 0.01456, 0.01446    # p22,p24,p25,p15
], dtype=float)

pOt = 1.01000  # [bar] ABS

def compute_pressures_from_precut(U, zero_offsets):

    U = U.copy()
```

```
    zero_offsets = zero_offsets.copy()

    U[:, 8:12] = U[:, 8:12] / 50.0
    zero_offsets[8:12] = zero_offsets[8:12] / 50.0

    return (U - zero_offsets) / DIVIDERS_ACTUAL + pOt

def compute_single_sided_fft(y, fs):
    y = np.asarray(y, dtype=float)

    if y.size < 2:
        return np.array([]), np.array([])

    y = y - np.mean(y)

    win = np.hanning(y.size)
    yw = y * win

    spec = np.fft.rfft(yw)
    freq = np.fft.rfftfreq(y.size, d=1.0/fs)

    amp = (2.0 / np.sum(win)) * np.abs(spec)

    if amp.size:
        amp[0] *= 0.5
        if y.size % 2 == 0 and amp.size > 1:
            amp[-1] *= 0.5

    return freq, amp

def _find_peak_in_range(f, A, fmin=0.0, fmax=6000.0):
    f = np.asarray(f, dtype=float)
    A = np.asarray(A, dtype=float)

    mask = (f >= fmin) & (f <= fmax)

    if not np.any(mask):
        return np.nan, np.nan

    ff = f[mask]
    AA = A[mask]

    if AA.size == 0:
        return np.nan, np.nan

    i = int(np.nanargmax(AA))
    return float(ff[i]), float(AA[i])

def plot_fft_all_channels_before_after(t, P, w_before, w_after, stem, out_png):
    n_ch = len(WALL_CHANNELS)

    fig_h = max(2.2 * n_ch, 14)
    fig, axes = plt.subplots(
        n_ch, 1,
        figsize=(12, fig_h),
```

```
        sharex=True
    )

    if n_ch == 1:
        axes = [axes]

    fig.subplots_adjust(left=0.08, right=0.97, top=0.97, bottom=0.06, hspace=0.28)

    b0, b1 = map(int, w_before)
    a0, a1 = map(int, w_after)

    peak_rows = []
    # ===== global y-scale =====
    all_amps = []
    for lab in WALL_CHANNELS:
        y = P[:, idx[lab]].astype(float)
        y_before = y[b0:b1+1]
        y_after  = y[a0:a1+1]
        _, A_b = compute_single_sided_fft(y_before, FS)
        _, A_a = compute_single_sided_fft(y_after, FS)
        if A_b.size: all_amps.append(np.nanmax(A_b))
        if A_a.size: all_amps.append(np.nanmax(A_a))
    if all_amps:
        global_ymax = max(all_amps) * 1.10
    else:
        global_ymax = 1.0

    for iax, (ax, lab) in enumerate(zip(axes, WALL_CHANNELS)):
        y = P[:, idx[lab]].astype(float)

        y_before = y[b0:b1+1]
        y_after  = y[a0:a1+1]

        f_b, A_b = compute_single_sided_fft(y_before, FS)
        f_a, A_a = compute_single_sided_fft(y_after, FS)

        line_b, = ax.plot(f_b, A_b, lw=1.0, label=f'{lab} BEFORE')
        line_a, = ax.plot(f_a, A_a, lw=1.0, label=f'{lab} AFTER')

        fpk_b, Apk_b = _find_peak_in_range(f_b, A_b, 0.0, 6000.0)
        fpk_a, Apk_a = _find_peak_in_range(f_a, A_a, 0.0, 6000.0)

        if np.isfinite(fpk_b):
            ax.axvline(fpk_b, color=line_b.get_color(), lw=0.9, linestyle='--',
            ↪  alpha=0.9)
            ax.text(
                fpk_b + 50.0,
                0.045,
                f'{fpk_b:.1f} Hz',
                color=line_b.get_color(),
                fontsize=8,
                ha='left',
                va='bottom'
            )

```

```
148         if np.isfinite(fpk_a):
149             ax.axvline(fpk_a, color=line_a.get_color(), lw=0.9, linestyle='--',
                ↪  alpha=0.9)
150             ax.text(
151                 fpk_a + 50.0,
152                 0.04,
153                 f'{fpk_a:.1f} Hz',
154                 color=line_a.get_color(),
155                 fontsize=8,
156                 ha='left',
157                 va='bottom'
158             )
159 
160         peak_rows.append((lab, fpk_b, Apk_b, fpk_a, Apk_a))
161 
162         ax.set_xlim(0, 6000)
163         ax.set_ylim(0, global_ymax)
164         ax.set_ylim(0, 0.05)
165         ax.set_ylabel(f'{lab}\nFFT amp. [bar]')
166         ax.grid(True, alpha=0.30)
167 
168         if iax == 0:
169             ax.legend(loc='upper right', fontsize=8)
170 
171     axes[-1].set_xlabel('frequency [Hz]')
172 
173     fig.savefig(out_png, dpi=150, bbox_inches='tight')
174     plt.close(fig)
175 
176     # tabela peaków obok grafiki
177     out_csv = pathlib.Path(out_png).with_suffix('.csv')
178     with open(out_csv, 'w', encoding='utf-8') as f:
179         f.write('channel,f_peak_before_Hz,A_peak_before_bar,f_peak_after_Hz,
180         A_peak_after_bar\n')
181         for lab, fpk_b, Apk_b, fpk_a, Apk_a in peak_rows:
182             f.write(f'{lab},{fpk_b},{Apk_b},{fpk_a},{Apk_a}\n')
183 
184     return out_csv
185 
186 def process_pair(precut_path: pathlib.Path):
187     stem = precut_path.stem
188 
189     off_path = precut_path.with_name(stem.replace("_precut", "_pre_zerooffset") +
    ↪  ".npy")
190     summary_path = SUMMARY_DIR / f"{stem}_summary.npy"
191 
192     if not summary_path.exists():
193         print(f"[WARN] Brak summary dla {stem}, pomijam.")
194         return
195 
196     if not off_path.exists():
197         print(f"[WARN] Brak zerooffset dla {stem}, pomijam.")
198         return
199 
200     U = np.load(precut_path)
```

```
    zero = np.load(off_path).astype(float)

    if U.ndim != 2 or U.shape[1] != 12:
        print(f"[ERROR] {precut_path.name}: oczekuję (N,12), mam {U.shape}")
        return

    if zero.shape[0] != 12:
        print(f"[ERROR] {off_path.name}: oczekuję (12,), mam {zero.shape}")
        return

    summary = np.load(summary_path, allow_pickle=True).item()
    w_before = tuple(summary["windows"]["before_idx"])
    w_after  = tuple(summary["windows"]["after_idx"])

    P = compute_pressures_from_precut(U, zero)
    t = np.arange(P.shape[0]) * DT

    out_png = OUT_DIR / f"{stem}_FFT_all_wall_channels_before_after.png"
    out_csv = plot_fft_all_channels_before_after(t, P, w_before, w_after, stem,
    ↪  out_png)

    print(f"[OK] {out_png.name}")
    print(f"[OK] {out_csv.name}")

def main():
    files = sorted(INPUT_DIR.glob("_DTN_20241030_145052*_precut.npy"))

    if not files:
        print(f"[WARN] Brak plików *_precut.npy w {INPUT_DIR}")
        return

    for f in files:
        try:
            process_pair(f)
        except Exception as e:
            print(f"[ERROR] {f.name}: {e}")


if __name__ == "__main__":
    main()
```

Appendix J

Raw and optimized values of wall pressure and secondary mass flow

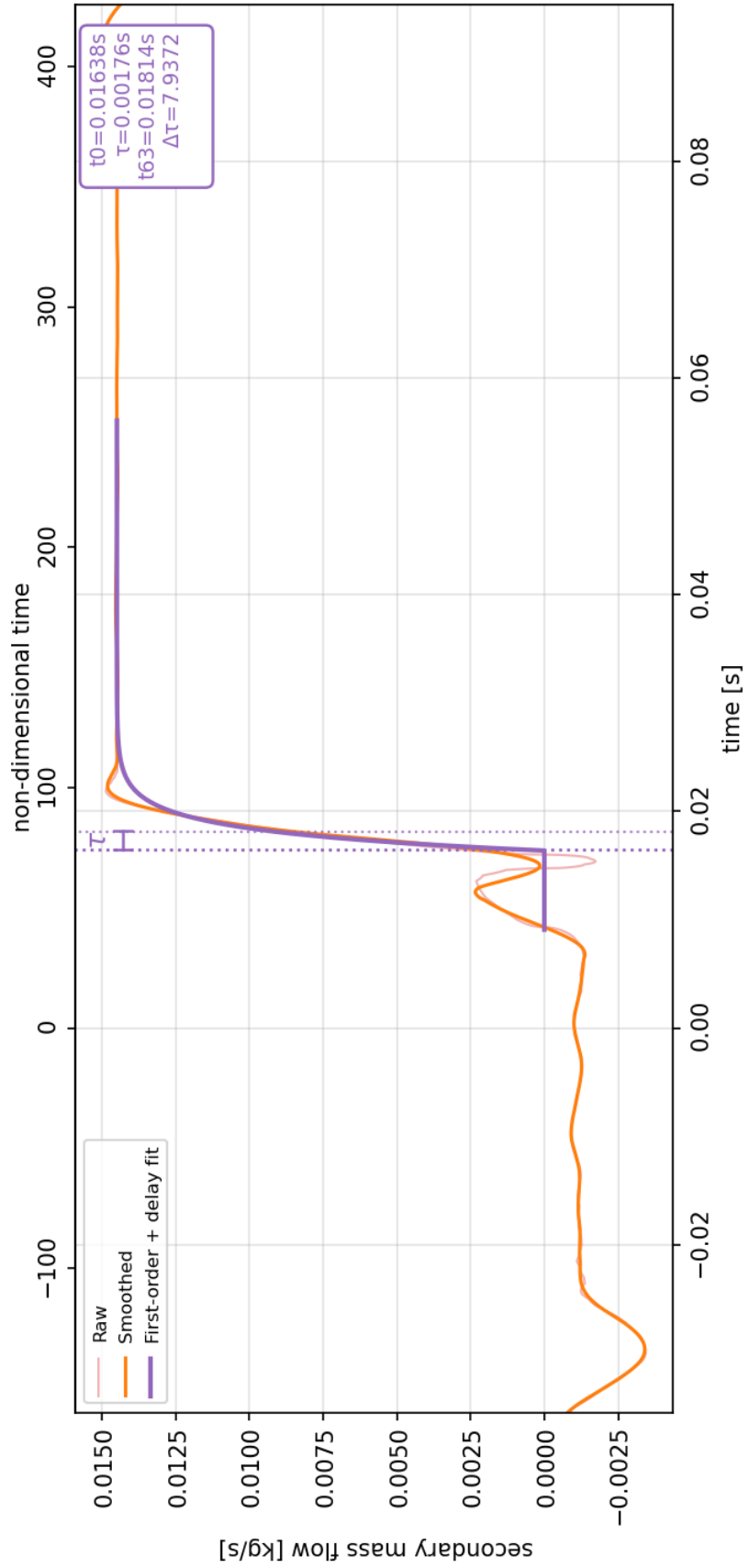


FIGURE J.1: Raw and optimized values of secondary mass flow w_s over time and dimensionless time

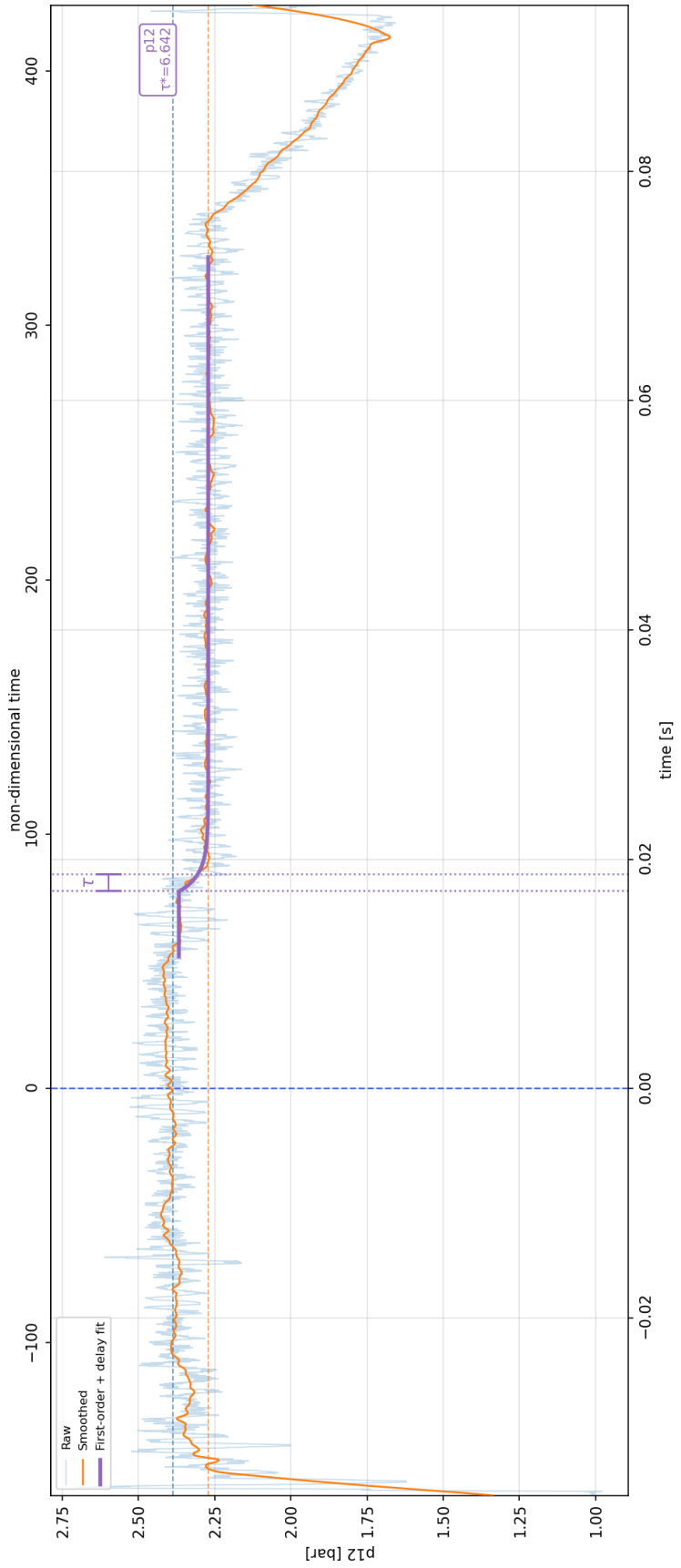


FIGURE J.2: Raw and optimized values of wall pressure at point p_{11} over time and dimensionless time

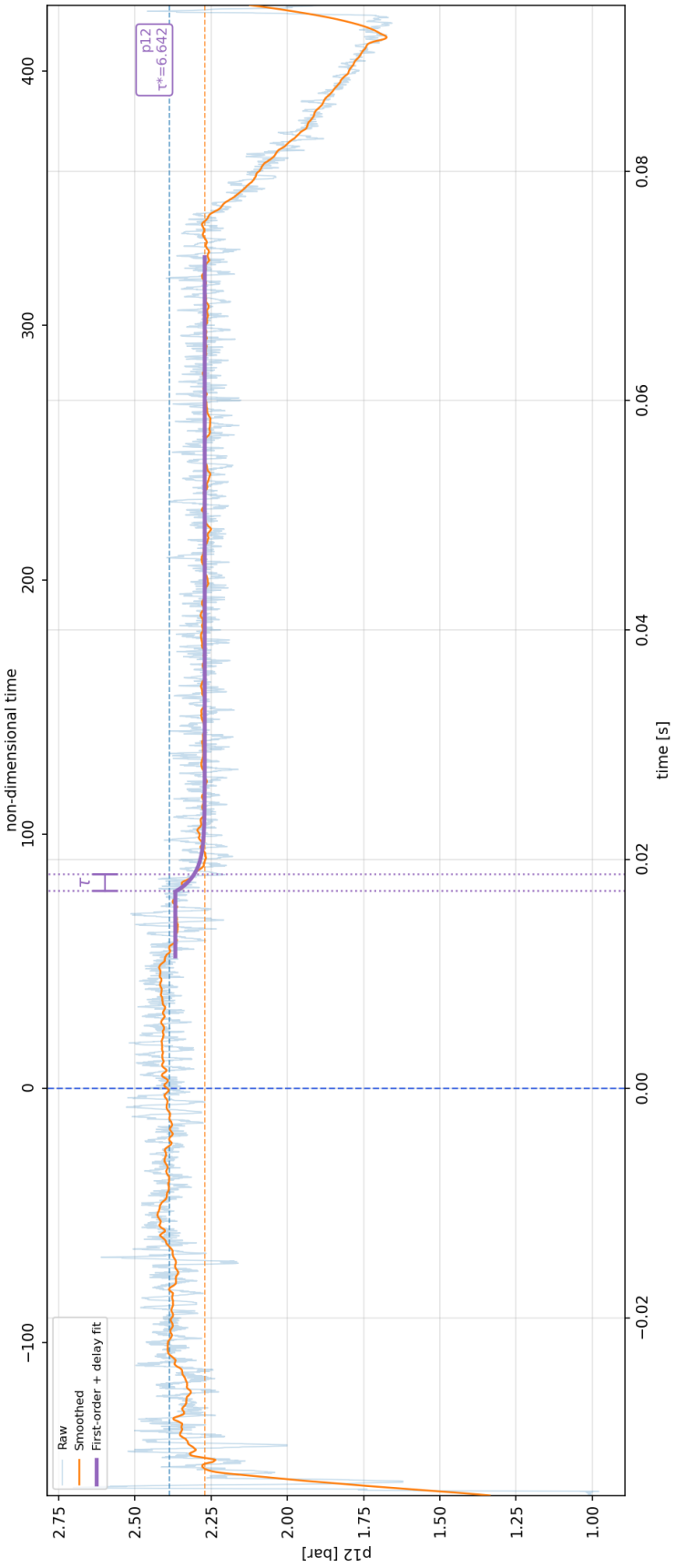


FIGURE J.3: Raw and optimized values of wall pressure at point p_{12} over time and dimensionless time

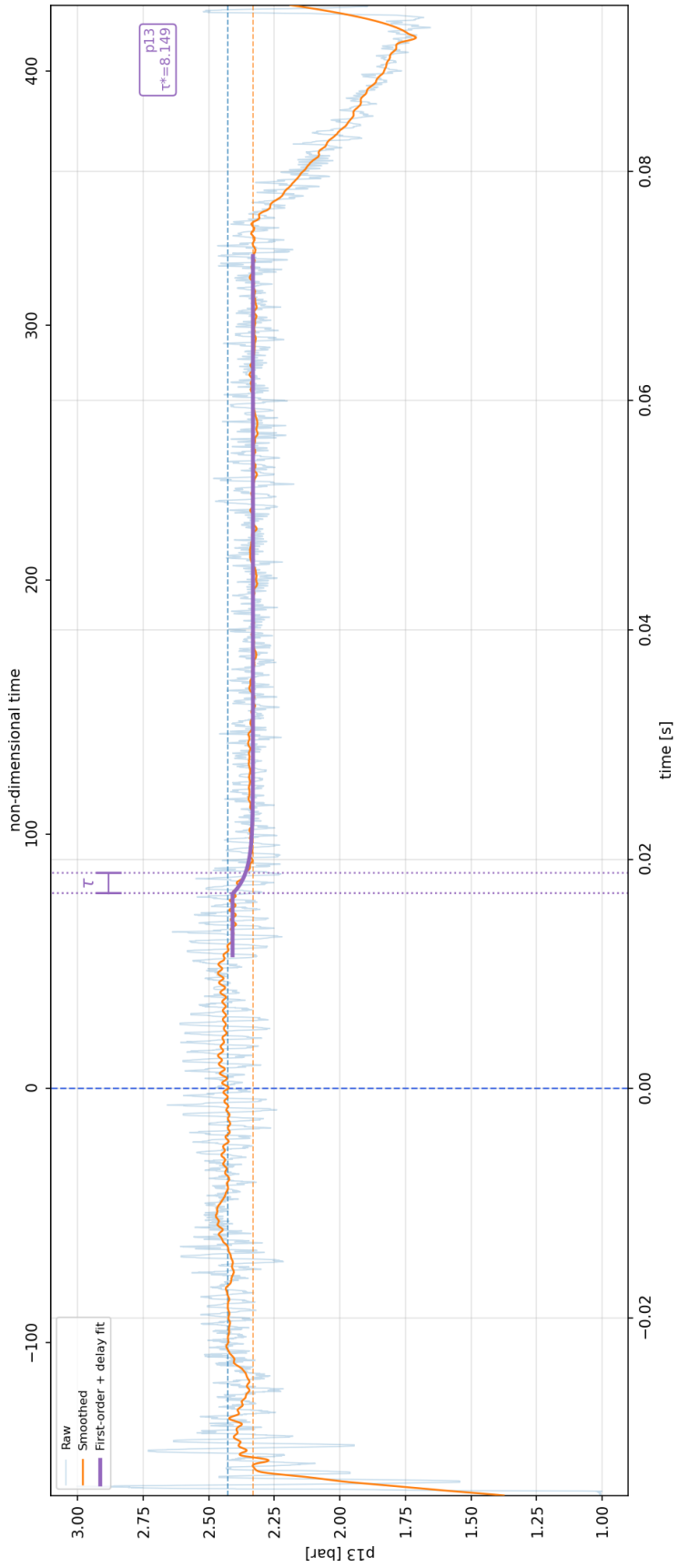


FIGURE J.4: Raw and optimized values of wall pressure at point p_{13} over time and dimensionless time

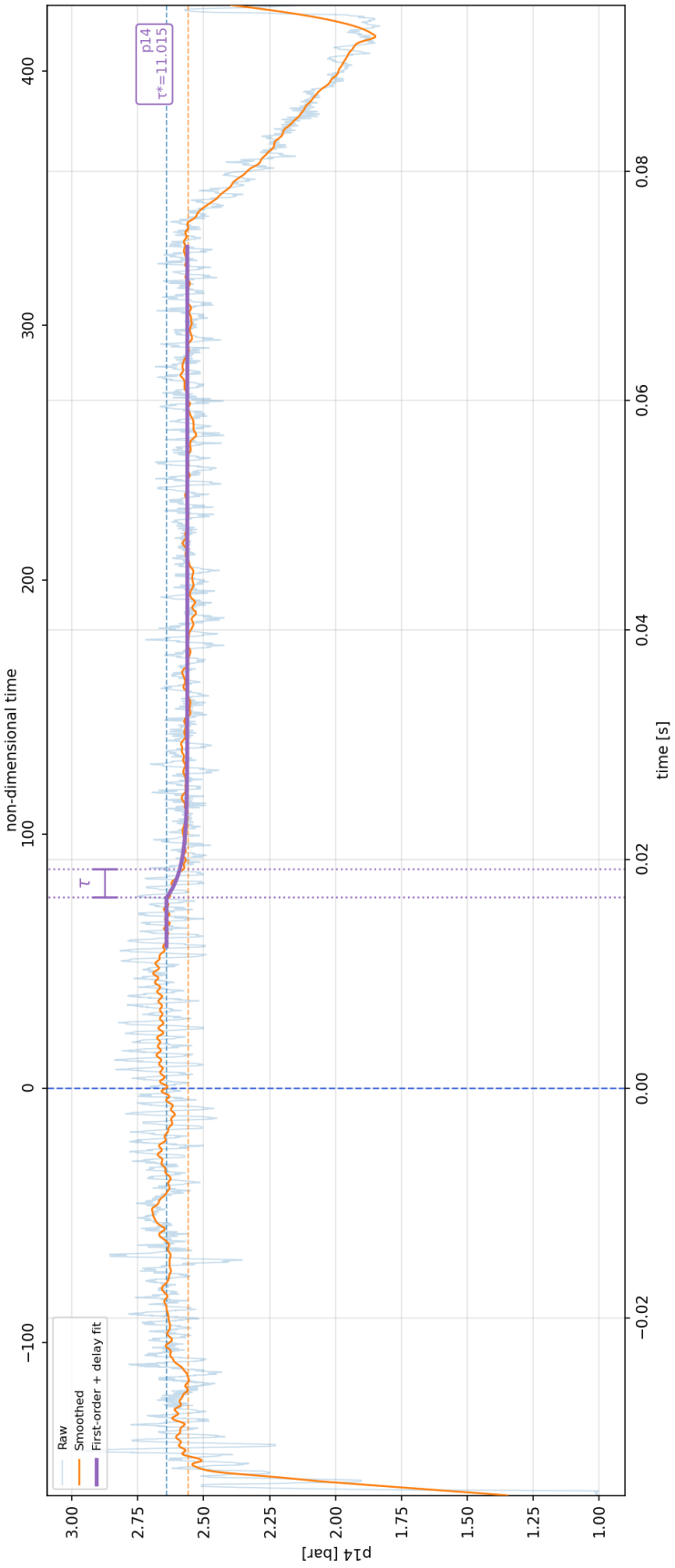


FIGURE J.5: Raw and optimized values of wall pressure at point p_{14} over time and dimensionless time

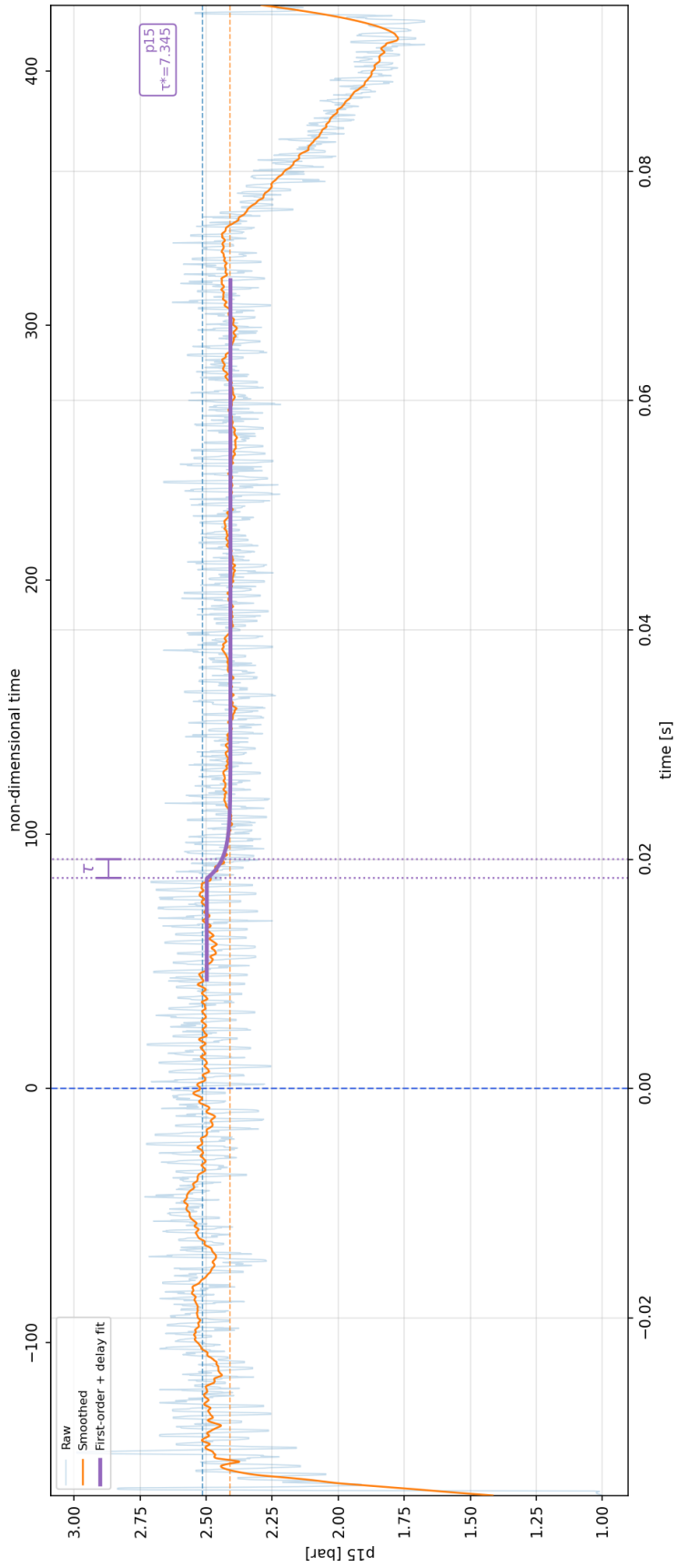


FIGURE J.6: Raw and optimized values of wall pressure at point p_{15} over time and dimensionless time

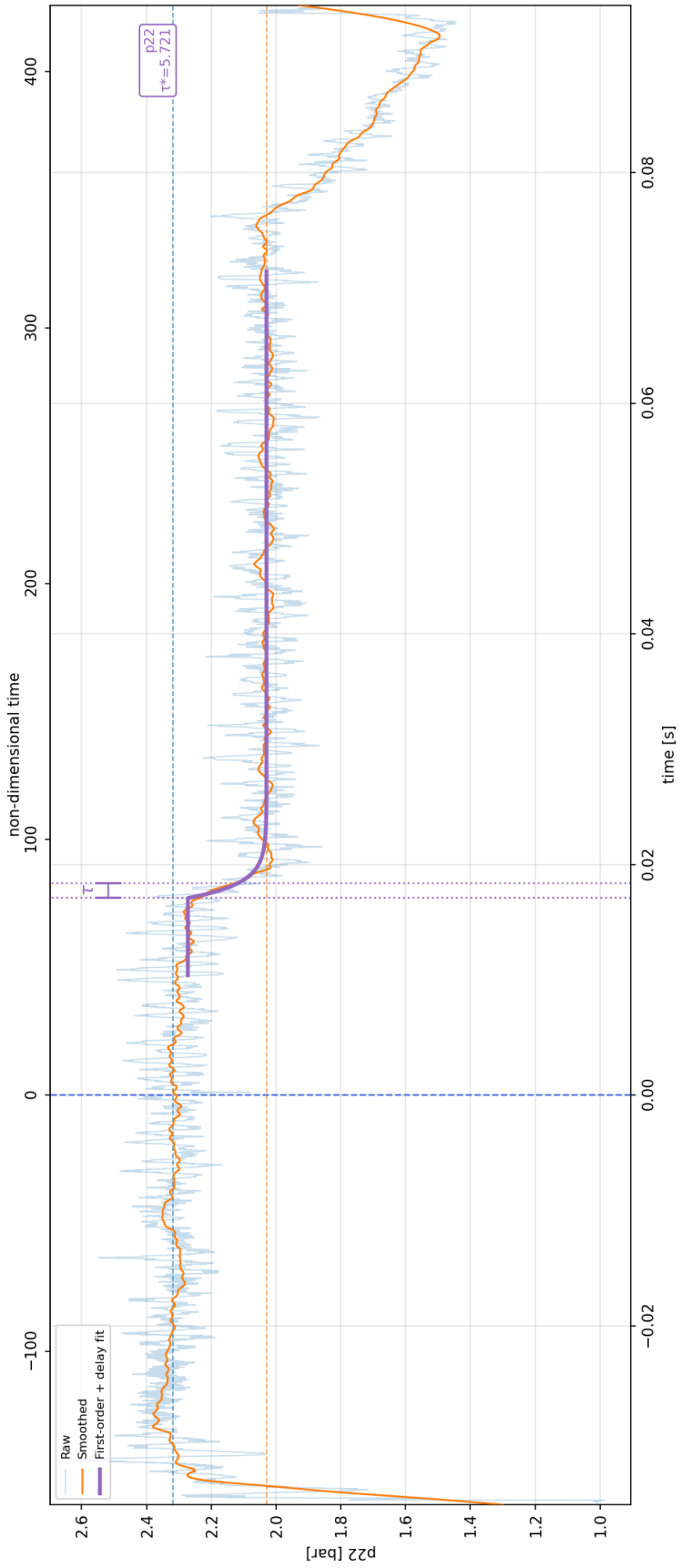


FIGURE J.7: Raw and optimized values of wall pressure at point p_{22} over time and dimensionless time

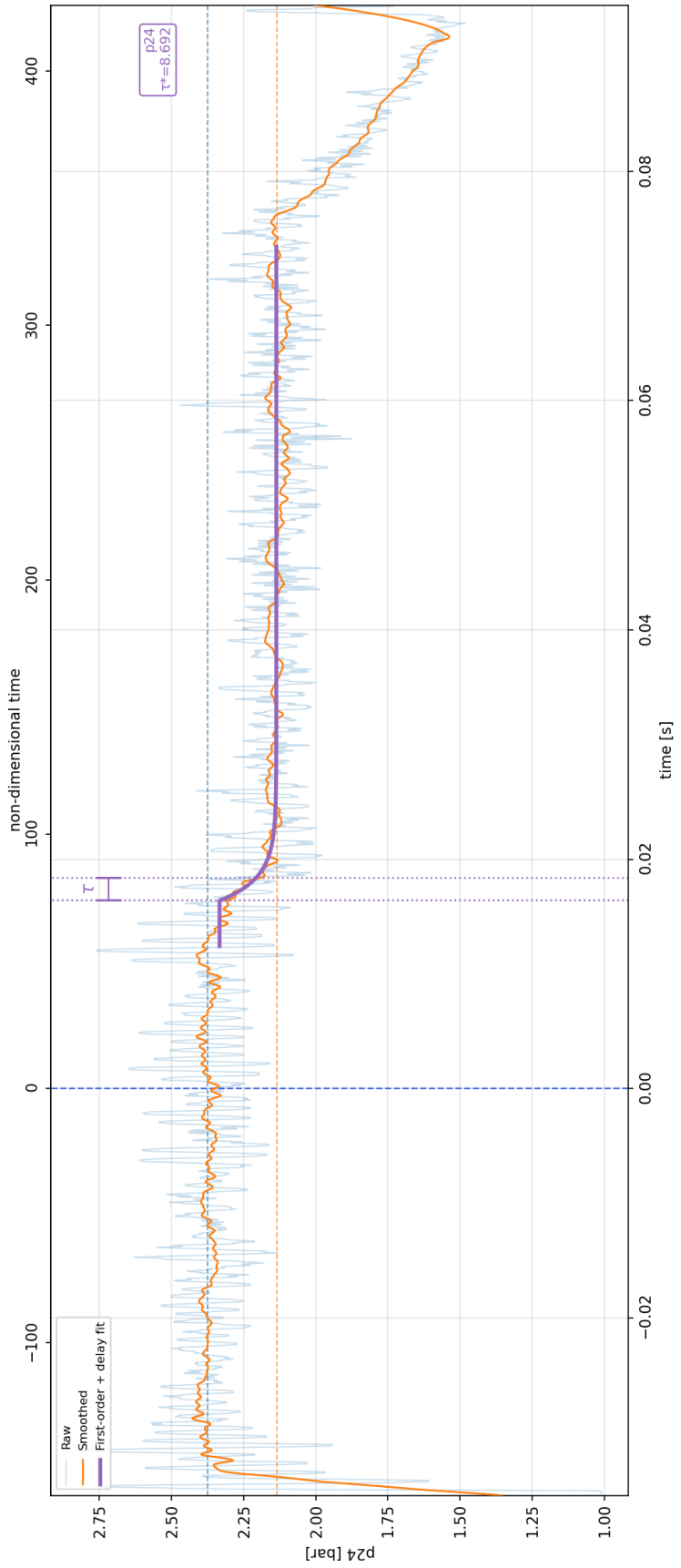


FIGURE J.8: Raw and optimized values of wall pressure at point p_{24} over time and dimensionless time

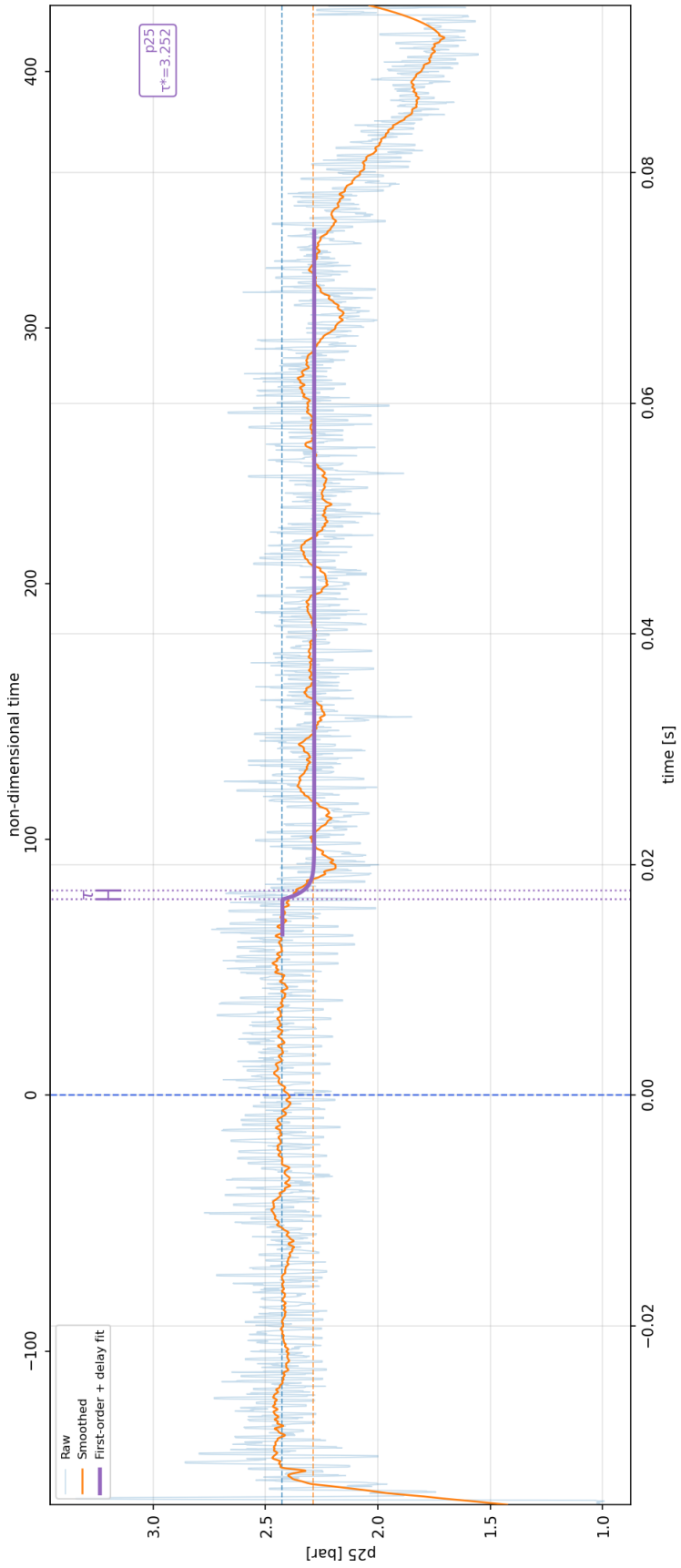


FIGURE J.9: Raw and optimized values of wall pressure at point p_{25} over time and dimensionless time

Appendix K

Wall Plots

```
"""
TVC_WallPressure_MeanProfiles_0.2.py

W.Hoffmann

"""

import pathlib
import numpy as np
import matplotlib.pyplot as plt


# ================== PATHS ==================
BASE_DIR    = pathlib.Path(r"C:/Users/hoffm/DTN/_out_20241030_1")
INPUT_DIR   = BASE_DIR
SUMMARY_DIR = BASE_DIR / "_analysis_from_precut" / "_npy"
OUT_DIR     = BASE_DIR / "_analysis_from_precut" / "_winplots"
OUT_DIR.mkdir(parents=True, exist_ok=True)


# ================== CHANNELS ==================
labels = ['p7','p1','p3','p6','p11','p12','p13','p14','p22','p24','p25','p15']
idx = {lab: i for i, lab in enumerate(labels)}

UPPER_CH = ['p22', 'p24', 'p25']
LOWER_CH = ['p11', 'p12', 'p13', 'p14', 'p15']

# ================== NOZZLE GEOMETRY ==================
# proste odcinki obrysu (mm); wysokość szczeliny h = 30 mm -> y/h = ±0.5 w
↪  wlocie/wylocie
GEO_UPPER_WALL = np.array([[-9.0,  15.00], [0.0, 15.00], [49.4, 23.71], [65.0,
↪  15.00]], dtype=float)
GEO_LOWER_WALL = np.array([[-9.0, -15.00], [0.0,-15.00], [49.4,-23.71],
↪  [65.0,-15.00]], dtype=float)
X_L  = 65.0   # mm (l)
H_MM = 30.0   # mm (d)

POS_X = {'p11':21.1, 'p12':28.2, 'p13':35.2, 'p14':42.3, 'p15':57.6,
```

```
         'p22':21.1, 'p24':42.3, 'p25':57.6}

X_MM = POS_X


# ================== CALIBRATION ==================
DIVIDERS_ACTUAL = np.array([
    0.49590, 0.83396, 0.83000, 0.49807,   # p7, p1, p3, p6
    0.00082, 0.00431, 0.00515, 0.00480,   # p11..p14
    0.01429, 0.01467, 0.01456, 0.01446    # p22,p24,p25,p15
], dtype=float)

pOt = 1.01000  # [bar] ABS


def compute_pressures_from_precut(U, zero_offsets):
    U = U.copy()
    zero_offsets = zero_offsets.copy()

    U[:, 8:12] = U[:, 8:12] / 50.0
    zero_offsets[8:12] = zero_offsets[8:12] / 50.0

    return (U - zero_offsets) / DIVIDERS_ACTUAL + pOt


def _window_stats_nd_pressure(P, channels, window_idx):
    i0, i1 = map(int, window_idx)
    sl = slice(i0, i1 + 1)

    p3 = P[sl, idx['p3']].astype(float)

    x = []
    mean = []
    std = []

    for ch in channels:
        pch = P[sl, idx[ch]].astype(float)
        nd = pch / p3

        x.append(X_MM[ch])
        mean.append(float(np.nanmean(nd)))
        std.append(float(np.nanstd(nd, ddof=1)))

    return np.asarray(x), np.asarray(mean), np.asarray(std)


def _sort_by_x(x, mean, std):
    order = np.argsort(x)
    return x[order], mean[order], std[order]


def _label_points(ax, x_vals, ch_list, y_frac=0.96):
    y0, y1 = ax.get_ylim()
    y = y0 + y_frac * (y1 - y0)
    for x, ch in zip(x_vals, ch_list):
```

```
91          ax.text(x, y, ch, ha='center', va='top', fontsize=8)
92
93
94  def _plot_geometry_panel(ax):
95      ax.plot(GEO_UPPER_WALL[:, 0], GEO_UPPER_WALL[:, 1] / H_MM, color='0.25', lw=1.4)
96      ax.plot(GEO_LOWER_WALL[:, 0], GEO_LOWER_WALL[:, 1] / H_MM, color='0.25', lw=1.4)
97
98      # punkty pomiarowe
99      for ch in UPPER_CH:
100         x = POS_X[ch]
101         y = np.interp(x, GEO_UPPER_WALL[:, 0], GEO_UPPER_WALL[:, 1]) / H_MM
102         ax.plot(x, y, marker='o', ms=4, color='tab:blue')
103         ax.text(x, y + 0.055, ch, ha='center', va='bottom', fontsize=8,
            ↪  color='tab:blue')
104
105     for ch in LOWER_CH:
106         x = POS_X[ch]
107         y = np.interp(x, GEO_LOWER_WALL[:, 0], GEO_LOWER_WALL[:, 1]) / H_MM
108         ax.plot(x, y, marker='o', ms=4, color='tab:orange')
109         ax.text(x, y - 0.055, ch, ha='center', va='top', fontsize=8,
            ↪  color='tab:orange')
110
111     ax.axhline(0.0, color='0.65', lw=0.8, linestyle='--')
112     ax.set_ylabel(r"$y/h$ [-]")
113     ax.set_title("Nozzle geometry and pressure tap positions")
114     ax.grid(True, alpha=0.25)
115     ax.set_ylim(-0.9, 0.9)
116
117
118 def _scale_to_band(y, y_min, y_max, band_min, band_max):
119     y = np.asarray(y, dtype=float)
120     if not np.isfinite(y_min) or not np.isfinite(y_max) or np.isclose(y_max, y_min):
121         return np.full_like(y, 0.5 * (band_min + band_max), dtype=float)
122     return band_min + (y - y_min) / (y_max - y_min) * (band_max - band_min)
123
124
125 def _err_to_band(err, y_min, y_max, band_min, band_max):
126     err = np.asarray(err, dtype=float)
127     if not np.isfinite(y_min) or not np.isfinite(y_max) or np.isclose(y_max, y_min):
128         return np.zeros_like(err, dtype=float)
129     return err / (y_max - y_min) * (band_max - band_min)
130
131
132 def plot_wall_pressure_profiles(P, w_before, w_after, stem, out_png):
133     x_l_b, m_l_b, s_l_b = _window_stats_nd_pressure(P, LOWER_CH, w_before)
134     x_l_a, m_l_a, s_l_a = _window_stats_nd_pressure(P, LOWER_CH, w_after)
135
136     x_u_b, m_u_b, s_u_b = _window_stats_nd_pressure(P, UPPER_CH, w_before)
137     x_u_a, m_u_a, s_u_a = _window_stats_nd_pressure(P, UPPER_CH, w_after)
138
139     x_l_b, m_l_b, s_l_b = _sort_by_x(x_l_b, m_l_b, s_l_b)
140     x_l_a, m_l_a, s_l_a = _sort_by_x(x_l_a, m_l_a, s_l_a)
141     x_u_b, m_u_b, s_u_b = _sort_by_x(x_u_b, m_u_b, s_u_b)
142     x_u_a, m_u_a, s_u_a = _sort_by_x(x_u_a, m_u_a, s_u_a)
143
```

```
    # osobne skale danych dla lower i upper
    lower_vals = np.r_[m_l_b - s_l_b, m_l_b + s_l_b, m_l_a - s_l_a, m_l_a + s_l_a]
    upper_vals = np.r_[m_u_b - s_u_b, m_u_b + s_u_b, m_u_a - s_u_a, m_u_a + s_u_a]

    l_min = float(np.nanmin(lower_vals))
    l_max = float(np.nanmax(lower_vals))
    u_min = float(np.nanmin(upper_vals))
    u_max = float(np.nanmax(upper_vals))

    def _pad_limits(vmin, vmax, pad_frac=0.12):
        span = vmax - vmin
        if not np.isfinite(span) or span <= 0:
            span = max(abs(vmin), 1.0) * 0.02
        return vmin - pad_frac * span, vmax + pad_frac * span

    l_min, l_max = _pad_limits(l_min, l_max)
    u_min, u_max = _pad_limits(u_min, u_max)

    LOWER_BAND = (0.05, 0.45)
    UPPER_BAND = (0.55, 0.95)

    yl_b = _scale_to_band(m_l_b, l_min, l_max, *LOWER_BAND)
    yl_a = _scale_to_band(m_l_a, l_min, l_max, *LOWER_BAND)
    yu_b = _scale_to_band(m_u_b, u_min, u_max, *UPPER_BAND)
    yu_a = _scale_to_band(m_u_a, u_min, u_max, *UPPER_BAND)

    el_b = _err_to_band(s_l_b, l_min, l_max, *LOWER_BAND)
    el_a = _err_to_band(s_l_a, l_min, l_max, *LOWER_BAND)
    eu_b = _err_to_band(s_u_b, u_min, u_max, *UPPER_BAND)
    eu_a = _err_to_band(s_u_a, u_min, u_max, *UPPER_BAND)

    fig, ax_p = plt.subplots(1, 1, figsize=(11.0, 6.5))
    fig.subplots_adjust(left=0.12, right=0.80, top=0.91, bottom=0.13)

    # prawa oś: geometria
    ax_g = ax_p.twinx()
    ax_g.patch.set_visible(False)

    ax_g.plot(
        GEO_UPPER_WALL[:, 0], GEO_UPPER_WALL[:, 1] / H_MM,
        color='0.72', lw=2.0, zorder=0
    )
    ax_g.plot(
        GEO_LOWER_WALL[:, 0], GEO_LOWER_WALL[:, 1] / H_MM,
        color='0.72', lw=2.0, zorder=0
    )
    ax_g.fill_between(
        GEO_UPPER_WALL[:, 0],
        np.interp(GEO_UPPER_WALL[:, 0], GEO_LOWER_WALL[:, 0], GEO_LOWER_WALL[:, 1]) /
        ↪  H_MM,
        GEO_UPPER_WALL[:, 1] / H_MM,
        color='0.92', alpha=0.45, zorder=0
    )

    for ch in UPPER_CH:
```

```
198         x = POS_X[ch]
199         y = np.interp(x, GEO_UPPER_WALL[:, 0], GEO_UPPER_WALL[:, 1]) / H_MM
200         ax_g.plot(x, y, marker='o', ms=4, color='0.45', zorder=1)
201         ax_g.text(x, y + 0.055, ch, ha='center', va='bottom', fontsize=8,
            ↪  color='0.35')
202
203     for ch in LOWER_CH:
204         x = POS_X[ch]
205         y = np.interp(x, GEO_LOWER_WALL[:, 0], GEO_LOWER_WALL[:, 1]) / H_MM
206         ax_g.plot(x, y, marker='o', ms=4, color='0.45', zorder=1)
207         ax_g.text(x, y - 0.055, ch, ha='center', va='top', fontsize=8, color='0.35')
208
209     ax_g.axhline(0.0, color='0.78', lw=0.8, linestyle='--', zorder=0)
210     ax_g.set_ylabel(r"geometry $y/h$ [-]", color='0.35')
211     ax_g.tick_params(axis='y', labelcolor='0.35')
212     ax_g.set_ylim(-0.9, 0.9)
213
214     # główna oś: przeszkalowane pasma 0..1
215     ax_p.set_ylim(0, 1)
216     ax_p.axhline(0.50, color='0.60', lw=0.9, linestyle='--')
217     ax_p.text(0.01, 0.94, "UPPER wall scale", transform=ax_p.transAxes,
218               ha='left', va='top', fontsize=9, color='0.25')
219     ax_p.text(0.01, 0.44, "LOWER wall scale", transform=ax_p.transAxes,
220               ha='left', va='top', fontsize=9, color='0.25')
221
222     # pressure profiles
223     ax_p.errorbar(
224         x_u_b, yu_b, yerr=eu_b,
225         fmt='o-', capsize=4, lw=1.5, ms=5,
226         label='upper before', zorder=6
227     )
228     ax_p.errorbar(
229         x_u_a, yu_a, yerr=eu_a,
230         fmt='o--', capsize=4, lw=1.5, ms=5,
231         label='upper after', zorder=6
232     )
233
234     ax_p.errorbar(
235         x_l_b, yl_b, yerr=el_b,
236         fmt='s-', capsize=4, lw=1.5, ms=5,
237         label='lower before', zorder=7
238     )
239     ax_p.errorbar(
240         x_l_a, yl_a, yerr=el_a,
241         fmt='s--', capsize=4, lw=1.5, ms=5,
242         label='lower after', zorder=7
243     )
244
245     ax_p.set_xlabel("x [mm]")
246     ax_p.grid(True, alpha=0.25)
247
248     # ===== oś Y  =====
249     # Dolna połowa osi: skala LOWER
250     # Górna połowa osi: skala UPPER
251     n_ticks_lower = 5
```

```
    n_ticks_upper = 5

    lower_tick_vals = np.linspace(l_min, l_max, n_ticks_lower)
    upper_tick_vals = np.linspace(u_min, u_max, n_ticks_upper)

    lower_tick_pos = _scale_to_band(lower_tick_vals, l_min, l_max, *LOWER_BAND)
    upper_tick_pos = _scale_to_band(upper_tick_vals, u_min, u_max, *UPPER_BAND)

    ax_p.set_yticks(np.r_[lower_tick_pos, upper_tick_pos])
    ax_p.set_yticklabels(
        [f"{v:.3f}" for v in lower_tick_vals] +
        [f"{v:.3f}" for v in upper_tick_vals]
    )


    for i, tick_label in enumerate(ax_p.get_yticklabels()):
        if i < n_ticks_lower:
            tick_label.set_color('tab:orange')
        else:
            tick_label.set_color('tab:blue')

    ax_p.tick_params(axis='y', labelsize=8)
    ax_p.set_ylabel(r"$p_{i}/p_3$ [-]")

    all_x = np.r_[GEO_UPPER_WALL[:, 0], GEO_LOWER_WALL[:, 0],
    ↪  np.array(list(X_MM.values()), dtype=float)]
    x_pad = 0.04 * (np.max(all_x) - np.min(all_x))
    ax_p.set_xlim(np.min(all_x) - x_pad, np.max(all_x) + x_pad)

    h1, l1 = ax_p.get_legend_handles_labels()
    ax_p.legend(h1, l1, loc='center right', fontsize=9)

    fig.savefig(out_png, dpi=170, bbox_inches='tight')
    plt.close(fig)

    out_csv = pathlib.Path(out_png).with_suffix('.csv')
    with open(out_csv, 'w', encoding='utf-8') as f:
        f.write("wall,channel,x_mm,mean_before,std_before,mean_after,std_after\n")

        for ch in UPPER_CH:
            _, m_b, s_b = _window_stats_nd_pressure(P, [ch], w_before)
            _, m_a, s_a = _window_stats_nd_pressure(P, [ch], w_after)
            f.write(f"upper,{ch},{X_MM[ch]},{m_b[0]},{s_b[0]},{m_a[0]},{s_a[0]}\n")

        for ch in LOWER_CH:
            _, m_b, s_b = _window_stats_nd_pressure(P, [ch], w_before)
            _, m_a, s_a = _window_stats_nd_pressure(P, [ch], w_after)
            f.write(f"lower,{ch},{X_MM[ch]},{m_b[0]},{s_b[0]},{m_a[0]},{s_a[0]}\n")

    return out_csv

def process_pair(precut_path: pathlib.Path):
    stem = precut_path.stem

```

```
305     off_path = precut_path.with_name(stem.replace("_precut", "_pre_zerooffset") +
        ↪  ".npy")
306     summary_path = SUMMARY_DIR / f"{stem}_summary.npy"
307
308     if not summary_path.exists():
309         print(f"[WARN] Brak summary dla {stem}, pomijam.")
310         return
311
312     if not off_path.exists():
313         print(f"[WARN] Brak zerooffset dla {stem}, pomijam.")
314         return
315
316     U = np.load(precut_path)
317     zero = np.load(off_path).astype(float)
318
319     if U.ndim != 2 or U.shape[1] != 12:
320         print(f"[ERROR] {precut_path.name}: oczekuję (N,12), mam {U.shape}")
321         return
322
323     if zero.shape[0] != 12:
324         print(f"[ERROR] {off_path.name}: oczekuję (12,), mam {zero.shape}")
325         return
326
327     summary = np.load(summary_path, allow_pickle=True).item()
328     w_before = tuple(summary["windows"]["before_idx"])
329     w_after  = tuple(summary["windows"]["after_idx"])
330
331     P = compute_pressures_from_precut(U, zero)
332
333     out_png = OUT_DIR /
        ↪  f"{stem}_wall_pressure_mean_profiles_combo_half_axis_scales.png"
334     out_csv = plot_wall_pressure_profiles(P, w_before, w_after, stem, out_png)
335
336     print(f"[OK] {out_png.name}")
337     print(f"[OK] {out_csv.name}")
338
339
340 def main():
341     files = sorted(INPUT_DIR.glob("_DTN_20241030_145052*_precut.npy"))
342
343     # files = sorted(INPUT_DIR.glob("*_precut.npy"))
344
345     if not files:
346         print(f"[WARN] Brak plików *_precut.npy w {INPUT_DIR}")
347         return
348
349     for f in files:
350         try:
351             process_pair(f)
352         except Exception as e:
353             print(f"[ERROR] {f.name}: {e}")
354
355 if __name__ == "__main__":
356     main()
```

Bibliography

[1] S. Afridi, T. A. Khan, S. I. A. Shah, D. J. Kukulka, et al. Techniques of fluidic thrust vectoring in jet engine nozzles: A review. *Energies*, 16(15):5721, July 2023.

[2] S. Ahmad, S. Siddique, M. Yousaf, M. Tariq, M. Khan, and M. Alam. Computational and experimental investigation of fluidic thrust vectoring actuator. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 40:315, 2018.

[3] ANSYS Inc., Canonsburg, PA, USA. *ANSYS Fluent Theory Guide*, 2024.

[4] Baku13. Messerschmitt Me 163 Engine "HWK 109-509 A". Wikimedia Commons, Aug. 2005. Photograph of HWK 109-509 A rocket engine on display at the Luftwaffenmuseum, Berlin-Gatow, Germany. Licensed under CC BY-SA 3.0, CC BY 2.5 and GFDL.

[5] L. T. Bankston and H. M. Larsen. Thrust vectoring experiments with gas injection. NAVORD Report 6548, China Lake, May 1959.

[6] M. Bartlett. Smoothing periodograms from time-series with continuous spectra. *Nature*, 161:686–687, 5 1948.

[7] L. Bell. Stanley r. mickelsen safeguard complex: Introduction. Stanley R. Mickelsen Safeguard Complex website, 2026. Privately maintained historical website; use only for illustrative/historical context.

[8] E. G. Bellandi and A. J. Slippey. Preliminary analysis and design enhancements of a dual-throat FTV nozzle concept. In *39th AIAA Fluid Dynamics Conference*, San Antonio, TX, USA, June 2009. American Institute of Aeronautics and Astronautics. AIAA Paper 2009-3900.

[9] M. A. Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[10] L. Bridgman, editor. *Jane's All the World's Aircraft 1958–59*. Sampson Low, Marston & Co., London, 1958.

[11] J. R. Chambers. *McDonnell Douglas F-15 Eagle*. The NASA History Series. NASA, 2000. Retrieved 20 April 2010.

[12] Q. Y. Chen, D. Gottlieb, and J. S. Hesthaven. Uncertainty analysis for the steady-state flows in a dual throat nozzle. *Journal of Computational Physics*, 204:378–398, 3 2005.

[13] T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6(2):418–445, 1996.

[14] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[15] K. A. Deere. Summary of fluidic thrust vectoring research conducted at NASA langley research center. In *21st AIAA Applied Aerodynamics Conference*, Orlando, FL, USA, June 2003. American Institute of Aeronautics and Astronautics. AIAA Paper 2003-3800.

[16] K. A. Deere, B. L. Berrier, J. D. Flamm, and S. K. Johnson. A computational study of a dual throat fluidic thrust vectoring nozzle concept. In *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, Tucson, AZ, USA, July 2005. American Institute of Aeronautics and Astronautics. AIAA Paper 2005-3502.

[17] K. A. Deere, J. D. Flamm, B. L. Berrier, and S. K. Johnson. Computational study of an axisymmetric dual throat fluidic thrust vectoring nozzle for a supersonic aircraft application. In *43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, Cincinnati, OH, USA, July 2007. American Institute of Aeronautics and Astronautics. AIAA Paper 2007-5085.

[18] Defence Research and Development Organisation. Man-Portable Anti-Tank Guided Missile (MPATGM). `https://www.drdo.gov.in/man-portable-anti-tank-guided-missile`, 2024. [Online; accessed 26-January-2024].

[19] Defence Research and Development Organisation. Quick Reaction Surface to Air Missile (QRSAM). `https://www.drdo.gov.in/qrsam`, 2024. [Online; accessed 26-January-2024].

[20] S. R. Doty. Kunsan pilots improve capability with AIM-9X missile. Air Force Link, Feb. 2008.

[21] B. Dunbar. Solid rocket boosters. NASA, Mar. 2006. Archived NASA page.

[22] Eclipse, Inc. *Combustion Engineering Guide*. Chapter 1: Orifices & Flows, coefficients of discharge for various types of orifices.

[23] Engineers for Engineers. Orifices & flows, Oct. 2021. Engineering blog article.

[24] M. F. *Harrier*. Patrick Stephens Ltd, Cambridge, 1 edition, 1986.

[25] FAST Research Group. Experimental validation: Mach 7 ludwieg tube. The University of Texas at Austin, Aug. 2021.

[26] Federation of American Scientists. C-4 submarine-launched ballistic missile (slbm). `https://nuke.fas.org/guide/usa/slbm/c-4.htm`, 2 1997. Accessed: February 12, 2024.

[27] Federation of American Scientists. X-36. `https://man.fas.org/dod-101/sys/ac/x-36.htm`, 2024. Accessed: February 7, 2024.

[28] M. Ferlauto and R. Marsilio. Open and closed-loop responses of a dual-throat nozzle during fluidic thrust vectoring. In *52nd AIAA/SAE/ASEE Joint Propulsion Conference*, Salt Lake City, UT, USA, July 2016. American Institute of Aeronautics and Astronautics. AIAA Paper 2016-4504.

[29] M. Ferlauto and R. Marsilio. Numerical investigation of the dynamic characteristics of a dual-throat-nozzle for fluidic thrust-vectoring. *AIAA Journal*, 55:86–98, 2017.

[30] J. D. Flamm, K. A. Deere, B. L. Berrier, S. K. Johnson, and M. L. Mason. Experimental study of a dual-throat fluidic thrust-vectoring nozzle concept. In *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, Tucson, AZ, USA, July 2005. American Institute of Aeronautics and Astronautics. AIAA Paper 2005-3503.

[31] J. D. Flamm, K. A. Deere, M. L. Mason, B. L. Berrier, and S. K. Johnson. Design enhancements of the two-dimensional, dual throat fluidic thrust vectoring nozzle concept. In *3rd AIAA Flow Control Conference*, San Francisco, CA, USA, June 2006. American Institute of Aeronautics and Astronautics. AIAA Paper 2006-3701.

[32] J. D. Flamm, K. A. Deere, M. L. Mason, B. L. Berrier, and S. K. Johnson. Experimental study of an axisymmetric dual throat fluidic thrust vectoring nozzle for supersonic aircraft application. In *43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, Cincinnati, OH, USA, July 2007. American Institute of Aeronautics and Astronautics. AIAA Paper 2007-5084.

[33] H. P. Gavin. The levenberg–marquardt algorithm for nonlinear least squares curve-fitting problems. Technical report, Department of Civil and Environmental Engineering, Duke University, Durham, NC, USA, Mar. 2026.

[34] GlobalSecurity.org. Ss-26 (9k720 iskander) program. `https://www.globalsecurity.org/wmd/world/russia/ss-26-program.htm`, 2024. Accessed: February 7, 2024.

[35] R. H. Goddard. Mechanism for directing flight. US Patent 1,879,187, Sept. 1932.

[36] R. H. Goddard. *Rocket Development: Liquid-Fuel Rocket Research, 1929–1941.* Prentice-Hall, New York, 1948.

[37] J. C. Goodall. *America's Stealth Fighters and Bombers: B-2, F-117, YF-22 and YF-23.* Motorbooks International, Osceola, WI, USA, 1992.

[38] Y. Gordon. *Sukhoi Su-27 Flanker.* Airlife Publishing Ltd, Shrewsbury, England, 1999.

[39] Y. Gordon and P. Davidson. *Sukhoi Su-27 Flanker*, volume 42 of *Warbird Tech Series.* Specialty Press, 2006.

[40] R. Gu and J. Xu. Dynamic experimental investigations of a bypass dual throat nozzle. *Journal of Engineering for Gas Turbines and Power*, 137, 8 2015.

[41] R. Gu, J. Xu, and S. Guo. Experimental and numerical investigations of a bypass dual throat nozzle. *Journal of Engineering for Gas Turbines and Power*, 136, 2014.

[42] M. H. Hamedi-Estakhrsar and H. Mahdavy-Moghaddam. Experimental evaluation and numerical simulation of performance of the bypass dual throat nozzle. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 235:768–781, 6 2021.

[43] F. R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.

[44] W. Hoffmann. Transoniczny tunel aerodynamiczny w konfiguracji tuby ludwiega. In *Nauka dla obronności. Perspektywy logistyki wojskowej i cywilnej w kontekście zapewnienia odporności energetycznej siłom zbrojnym.* Instytut Techniczny Wojsk Lotniczych, 2022. In Polish.

[45] W. Hoffmann and A. Wróblewska. Analiza problemu stateczności przepływu w dyszach typu dual-throat. In *Nauka dla obronności. Perspektywy logistyki wojskowej i cywilnej w kontekście zapewnienia odporności energetycznej siłom zbrojnym. Tom 3*, pages 23–29. Instytut Techniczny Wojsk Lotniczych, 2024. In Polish.

[46] O. Igra and F. Seiler, editors. *Experimental Methods of Shock Wave Research*, volume 9 of *Shock Wave Science and Technology Reference Library.* Springer International Publishing, Cham, 2016.

[47] M. Kato and B. E. Launder. The modelling of turbulent flow around stationary and vibrating square cylinders. In *Proceedings of the 9th Symposium on Turbulent Shear Flows*, volume 1, pages 10.4.1–10.4.6, Kyoto, Japan, 1993.

[48] G. P. Kennedy. *Vengeance Weapon 2: The V-2 Guided Missile*. Smithsonian Institution Press, Washington, DC, 1983.

[49] M. Khalid and D. J. Jones. Navier-stokes investigation of blunt trailing-edge airfoils using o grids. *Journal of Aircraft*, 30(5):797–800, 1993.

[50] Kulite Semiconductor Products, Inc. *XTL-190 Series Miniature Pressure Transducer*. Kulite Semiconductor Products, Inc.

[51] L. D. Landau and E. M. Lifszyc. *Hydrodynamika*. Fizyka teoretyczna. Wydawnictwo Naukowe PWN, Warszawa, 1994. Tłum. z ros. Paweł Kowalczyk i Ludmiła Skubiszak.

[52] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[53] L. Li and T. Saito. Numerical and experimental investigations of fluidic thrust vectoring mechanism. *International Journal of Aerospace Innovations*, 4(1–2):53–64, 2012.

[54] Liquid Propulsion Systems Centre, ISRO. Propulsion systems: Secondary injection thrust vector control (SITVC) for PS1 stage. Indian Space Research Organisation.

[55] J. M. Logsdon, R. A. Williamson, R. D. Launius, R. J. Acker, S. J. Garber, and J. L. Friedman, editors. *Exploring the Unknown: Selected Documents in the History of the U.S. Civil Space Program, Volume IV: Accessing Space*. Number NASA SP-4407 in The NASA History Series. NASA History Division, Office of Policy and Plans, Washington, DC, 1999.

[56] H. Ludwieg. Ein neues verfahren für intermittierend betriebene hochgeschwindigkeits-windkanäle. Technical report, Aerodynamische Versuchsanstalt (AVA) Göttingen, Göttingen, Germany, 1955.

[57] Y. Maruyama, M. Sakata, and Y. Takahashi. Performance analyses of fluidic thrust vector control system using dual throat nozzle. *AIAA Journal*, 60:1730–1744, 2022.

[58] K. Mańczak. *Technika Planowania Eksperymentu*. Wydawnictwa Naukowo-Techniczne (WNT), Warszawa, 1976.

[59] K. D. McCutcheon. *The Redstone Engine*. Aircraft Engine Historical Society, Huntsville, Alabama, 2016. Archived from the original on 5 March 2016. Retrieved 06 February 2024.

[60] F. R. Menter, M. Kuntz, and R. Langtry. Ten years of industrial experience with the sst turbulence model. In K. Hanjalić, Y. Nagano, and M. Tummers, editors, *Proceedings of the 4th International Symposium on Turbulence, Heat and Mass Transfer*, pages 625–632, New York, 2003. Begell House.

[61] D. N. Miller and J. Catt. Conceptual development of fixed-geometry nozzles using fluidic injection for throat area control. In *31st Joint Propulsion Conference and Exhibit*, San Diego, CA, USA, July 1995. American Institute of Aeronautics and Astronautics. AIAA Paper 95-2603.

[62] D. N. Miller, P. J. Yagle, and J. W. Hamstra. Fluidic throat skewing for thrust vectoring in fixed-geometry nozzles. In *37th Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, Jan. 1999. American Institute of Aeronautics and Astronautics. AIAA Paper 99-0365.

[63] S. B. Moorhead. Refractory air vane and refractory material research and development task ii – refractory materials for a thrust vector control valve. Technical Report AD0530958, Army Materials and Mechanics Research Center, July 1974. Archived from the original on 23 February 2022. Retrieved 23 February 2022.

[64] National Aeronautics and Space Administration. *SLS Solid Rocket Booster Fact Sheet*. National Aeronautics and Space Administration, 2015.

[65] National Aeronautics and Space Administration. *NASA's Space Launch System Reference Guide*. National Aeronautics and Space Administration, 2022.

[66] National Air and Space Museum. Rocket motor, liquid fuel, HWK 109-509 A-1. Smithsonian National Air and Space Museum Collection.

[67] National Instruments. *NI 9234 Specifications*. National Instruments.

[68] National Instruments. *NI 9237 Specifications*. National Instruments.

[69] National Instruments. *NI cRIO-9024 Real-Time Controller: Specifications*. National Instruments.

[70] National Nuclear Security Administration. *Nuclear Weapons Characteristic Handbook*. National Nuclear Security Administration, 2017. Accessed: February 12, 2024.

[71] NumPy Developers. *numpy.mean*. NumPy, 2024.

[72] P. Panzarella. *The Use of a Coanda Nozzle with Parallel Secondary Injection for the Thrust Vectoring of a Two-Dimensional Compressible Fluid.* PhD thesis, Air Force Institute of Technology, Dayton, OH, USA, 1965.

[73] N. Penmetsa. *Experimental and Computational Investigation of a Dual-Throat Fluidic Thrust-Vectoring Nozzle.* PhD thesis, University of Colorado Boulder, 2016.

[74] M. Popejoy. Uss alabama offloads last of c4 trident missiles. US Navy, Nov. 2005. Archived from the original on September 12, 2007. Retrieved May 16, 2012.

[75] O. Reynolds. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical Transactions of the Royal Society of London A*, 186:123–164, 1895.

[76] V. Rodriguez, R. Saurel, G. Jourdan, and L. Houas. Solid-particle jet formation under shock-wave acceleration. *Physical Review E*, 88(6):063011, 2013.

[77] M. R. Salimi, R. Askari, and M. Hasani. Computational investigation of effects of side-injection geometry on thrust-vectoring performance in a fuel-injected dual throat nozzle. *Journal of Applied Fluid Mechanics*, 15:1137–1153, 7 2022.

[78] A. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[79] F. F. J. Schrijer and W. J. Bannink. Description and flow assessment of the delft hypersonic ludwieg tube. *Journal of Spacecraft and Rockets*, 47(1):125–133, 2010.

[80] M. Schulmeister. Static evaluation tests of an oblique shock wave system for rocket exhaust deflection. Technical Report NARTS 77, US Naval Air Rocket Test Station, Dover, NJ, USA, 1955.

[81] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. D. III. *Process Dynamics and Control.* John Wiley & Sons, Hoboken, NJ, 4 edition, 2017.

[82] K. S. Sekhon. *Thrust Vectoring Experiments with Gas Injection.* PhD thesis, The University Of Arizona, Tucson, AZ, USA, 1964.

[83] C. S. Shin, H. D. Kim, T. Setoguchi, and S. Matsuo. A computational study of thrust vectoring control using dual throat nozzle. *Journal of Thermal Science*, 19:486–490, 12 2010.

[84] Space Exploration Technologies Corp. *Falcon User's Guide.* Space Exploration Technologies Corp., 9 2021. Accessed: February 7, 2024.

[85] F. W. Spaid. Two-dimensional jet interaction studies at large values of reynolds and mach numbers. *AIAA Journal*, 13(11):1430–1434, 1975.

[86] S. Starr. The launch of Bumper 8 from the Cape: The end of an era and the beginning of another. In *Proceedings of the 52nd International Astronautical Congress*, Toulouse, France, Oct. 2001.

[87] P. J. Strykowski and A. Krothapalli. The countercurrent mixing layer: Strategies for shear-layer control. In *3rd Shear Flow Conference*, Orlando, FL, USA, July 1993. American Institute of Aeronautics and Astronautics. AIAA Paper 93-3260.

[88] TE Connectivity. *Measurement Specialties XPM4 Miniature Pressure Transducer*. TE Connectivity.

[89] TE Connectivity. *XP5 Pressure Transducer: Data Sheet*. TE Connectivity.

[90] The Engineer. Rolls-royce's liftsystem for the joint strike fighter. `https://web.archive.org/web/20131219050153/http://www.theengineer.co.uk/in-depth/rolls-royces-liftsystem-for-the-joint-strike-fighter/1008008.article`, 2013. Originally published in 2013, accessed via the Wayback Machine on February 7, 2024.

[91] R. J. Thomson. Study of selected thrust vector control systems for solid propellant motors. Technical report, Space-General Corporation, El Monte, CA, USA, 1965.

[92] U.S. Army Missile Command. The soviet surface-to-surface guided missile SCUD (SS-1b) (C). Technical report, U.S. Army Missile Command, 1963. Declassified document available through the CIA Reading Room.

[93] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey,
.I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020.

[94] Y. Wang, J. Xu, and S. Huang. Study of starting problem of axisymmetric divergent dual throat nozzle. *Journal of Engineering for Gas Turbines and Power*, 139(6):062602, 2017.

[95] Y. S. Wang, J. L. Xu, S. Huang, Y. C. Lin, and J. J. Jiang. Experimental and numerical investigation of an axisymmetric divergent dual throat nozzle. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 234:563–572, 3 2020.

[96] D. Washington, F. Alvi, P. Strykowski, and A. Krothapalli. Multiaxis fluidic thrust vector control of a supersonic jet using counterflow. *AIAA Journal*, 34(8):1734–1736, 1996.

[97] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15:70–73, 7 1967.

[98] WIKA Alexander Wiegand SE & Co. KG. *Pressure Calibrator Model CTC1100: Data Sheet CT 11.01*. WIKA, 2023.

[99] D. C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, La Canada, California, USA, 3 edition, 2006.

[100] K. Wu and H. D. Kim. A fluidic thrust vector control using the bypass flow in a dual throat nozzle. *Journal of Mechanical Science and Technology*, 35:3435–3443, 8 2021.

[101] K. X. Wu, T. H. Kim, and H. D. Kim. Numerical study of fluidic thrust vector control using dual throat nozzle. *Journal of Applied Fluid Mechanics*, 14:73–87, 1 2021.

[102] J. Zhao, X. Sima, R. Huang, et al. Design of a hypersonic ludwieg tunnel with a double-bent storage tube. *Acta Aerodynamica Sinica*, 40(4):90–100, 2022.