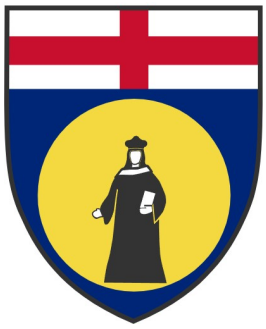
UNIVERSITÀ DEGLI STUDI DI GENOVA

SCUOLA POLITECNICA

DIME

Dipartimento di Ingegneria Meccanica, Energetica, Gestionale e dei Trasporti



TESI DI LAUREA MAGISTRALE

IN

INGEGNERIA MECCANICA - ENERGIA E AERONAUTICA

High-Performance Distributed Dynamic Mode Decomposition for Large-Scale Flow Separation Data

Relatori:
Prof. Ing. Davide Lengani, UniGe
Dr. Matteo Dellacasagrande, UniGe

Candidata:
Giulia Maria Podetti

Correlatori:
Dr. Ardeshir Hanifi, KTH
Dr. Mohammad Moniripiri, KTH

Luglio 2026

Abstract

The pursuit of higher aerodynamic efficiency in modern turbomachinery often requires an increasing aerodynamic loading of individual blade rows, heightening the risk of boundary layer separation on the suction side of highly loaded blades. Understanding the mechanisms governing separation and the subsequent laminar-to-turbulent transition is therefore essential for future low-pressure turbine design. Direct Numerical Simulation (DNS) offers the resolution required to capture these phenomena, but the resulting datasets, comprising thousands of three-dimensional snapshots, exceed the memory capacity of a single compute node and cannot be analysed with conventional serial tools.

This thesis presents a distributed-memory parallel framework, implemented in Python with MPI, for applying Dynamic Mode Decomposition (DMD) to large-scale DNS datasets describing a flat-plate boundary layer flow with separation. The framework first partitions the data in time to enable efficient parallel I/O and in-place spectral decomposition. An MPI "All-to-Allv" communication step is used to reorganize the data so that each rank stores the full temporal history of a spatial subdomain. This approach enables a windowed Exact DMD on each streamwise subdomain and spectral wavenumber.

The implementation was rigorously validated: data redistribution introduced no measurable numerical perturbation, and the DMD algorithm matched a reference MATLAB implementation to within double-precision tolerance on a classical benchmark. A strong scaling analysis on the Leonardo supercomputer at CINECA identified the best performing configuration at approximately 20 MPI ranks, beyond which communication overhead progressively offsets the benefits of parallelism, consistently with Amdahl's Law. The validated framework was then applied to a large-scale DNS dataset describing a separated flat-plate boundary layer. This preliminary application revealed a well-defined growth-saturation-decay trend in the dominant DMD eigenvalue modulus along the streamwise direction, qualitatively consistent with the classical Kelvin-Helmholtz instability mechanism and corroborated by an independent analysis of the streamwise evolution of the fluctuation energy. The resulting framework thus provides a validated, scalable tool for the modal analysis of large-scale separated flow datasets, capable of producing physically interpretable results on real DNS data.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. Davide Lengani and Dr. Matteo Dellacasagrande, for their constant guidance, valuable feedback, and the time they dedicated to this work throughout its development.

I am equally grateful to Dr. Ardeshir Hanifi and Dr. Mohammad Moniripiri, at KTH Royal Institute of Technology, for their supervision during my period abroad. Their support was instrumental in making my thesis experience abroad a highly positive and formative one.

Contents

List of Figures

List of Tables

List of Acronyms

CFD	Computational Fluid Dynamics
DFT	Discrete Fourier Transform
DMD	Dynamic Mode Decomposition
DNS	Direct Numerical Simulation
FFT	Fast Fourier Transform
FSTI	Free-Stream Turbulence Intensity
HDF5	Hierarchical Data Format, version 5
HPC	High-Performance Computing
K-H	Kelvin-Helmholtz (instability)
LES	Large-Eddy Simulation
LPT	Low-Pressure Turbine
LSB	Laminar Separation Bubble
MPI	Message Passing Interface
PIV	Particle Image Velocimetry
POD	Proper Orthogonal Decomposition
RAM	Random Access Memory
RANS	Reynolds-Averaged Navier-Stokes
rFFT	Real Fast Fourier Transform
SVD	Singular Value Decomposition

List of Symbols

Latin Symbols

A, $\tilde{A}$	Full-order and reduced-order DMD linear operator
b_i, $\mathbf{b}$	DMD mode amplitude, amplitude vector
C_f	Skin friction coefficient
d	Bar diameter (wake generator)
f	Frequency
f^+	Reduced wake passing frequency
k	Discrete wavenumber index
k_z	Spanwise wavenumber
K	Acceleration parameter
L_z	Domain extent along the spanwise direction
N_t	Number of temporal snapshots
N_x, N_y, N_z	Number of grid points along x, y, z
$N_{z,\mathrm{rfft}}$	Number of unique spectral coefficients (rFFT)
p	Pressure; also, parallelisable fraction (Amdahl's Law)
r	SVD truncation rank
Re, Re_x, Re_θ	Reynolds number, local and momentum-thickness-based
S_N	Parallel speedup with N processors
t	Time
T_1, T_N	Serial and N-processor execution time
Tu	Free-stream turbulence intensity (%)
u, v, w	Streamwise, wall-normal, spanwise velocity components
U, Σ, V	SVD factor matrices
U_e	Free-stream (edge) velocity
$\mathbf{w}_i$, W	Eigenvector, eigenvector matrix of $\tilde{A}$
X, X'	DMD snapshot matrix and time-shifted snapshot matrix
x, y, z	Streamwise, wall-normal, spanwise coordinates
Y_p	Total pressure loss coefficient

Greek Symbols

δ, δ^*	Boundary layer thickness, displacement thickness
θ	Momentum thickness
λ_i	Continuous DMD eigenvalue
λ_θ	Pressure gradient parameter

μ_i	Discrete DMD eigenvalue
ν	Kinematic viscosity
ρ	Density
σ_i	Growth/decay rate (real part of λ_i); also singular value
τ_w	Wall shear stress
Φ_i, Φ	DMD spatial mode, mode matrix
ω_i	Angular frequency (imaginary part of λ_i)

1 Introduction

Turbomachines represent the core of energy conversion in both aeronautical propulsion and power generation. Aircraft engines, gas turbines for electricity production, and marine propulsion systems all rely on the same operating principle, entrusting compressors and turbines with the task of exchanging energy with the working fluid through rotor and stator stages. Although these are mature technologies, the field continues to be the subject of intense research and innovation efforts, driven by the constant need to increase their efficiency [1].

This trend does not stem solely from economic considerations, related to the reduction of fuel consumption and operating costs, but is also strongly influenced by increasingly stringent regulatory and environmental constraints. International agreements on the reduction of CO2 and pollutant emissions, together with the decarbonization targets set by the aviation and energy industries, impose a path of continuous performance improvement for turbomachinery, in which even seemingly marginal efficiency gains have a significant impact at fleet or plant scale.

In this context, every component of the machine, from compressors to high- and low-pressure turbines, undergoes a continuous optimization process, in which small aerodynamic gains translate into concrete benefits in terms of fuel consumption, emissions, and reliability [2, 3]. This constant pursuit of maximum efficiency motivates the need to push the aerodynamic performance of individual components closer to their physical limits.

The pursuit of ever higher efficiency requires increasing the aerodynamic loading of individual stages, while simultaneously reducing the number of blades and vanes required to perform the same work. This trend responds primarily to the need for reducing the overall weight of the machine: a lower component count, for the same required performance, directly results in a lighter and more compact engine, with significant benefits especially in aeronautical applications, where weight has a direct impact on aircraft fuel consumption and performance [4].

At the same time, the push toward more compact and lightweight configurations cannot disregard the need to maintain, if not improve, the aerodynamic efficiency of the individual components. More heavily loaded blades, in fact, entail steeper pressure gradients along the blade surface, with a consequent increase in the risk of flow separation, particularly on the rear part of the suction side [5, 6]. The design challenge therefore becomes achieving the highest possible aerodynamic loading while minimizing the losses associated with these viscous phenomena, which represent one of the main

sources of inefficiency in modern turbomachinery.

This need to balance aerodynamic loading, weight, and viscous losses has driven the scientific and industrial community to explore alternative design solutions, capable of sustaining higher diffusion levels without incurring a significant loss of performance.

In order to meet this demand for increasingly higher aerodynamic loading without compromising performance, the scientific community has explored, over the years, blade configurations and design strategies alternative to more conservative approaches. Increasing the aerodynamic loading inevitably entails steeper adverse pressure gradients along the blade surface, particularly on the rear part of the suction side, where the risk of boundary layer separation becomes significant.

Fully exploiting these aerodynamic loading margins therefore requires the ability to understand and predict the boundary-layer response to adverse pressure gradients. It is necessary to correctly capture the onset of separation and the subsequent laminar-to-turbulent transition mechanisms. This kind of detailed physical knowledge is essential in order to reliably explore the space of possible design architectures, assessing which configurations can sustain the desired loading while containing the losses associated with these viscous phenomena.

This is the origin of the need for increasingly refined investigation tools, capable of describing with growing accuracy the evolution of the boundary layer and the associated separation and transition phenomena, discussed in the following.

The need to understand in detail the boundary layer separation and transition mechanisms has driven, over the decades, a continuous evolution of fluid dynamic investigation tools, both experimental and numerical. Measurement techniques have progressively evolved from pointwise, time-averaged approaches, such as hot-wire anemometry or static wall pressure measurements, to techniques capable of providing spatially resolved information, such as Particle Image Velocimetry (PIV), which can reconstruct the entire instantaneous velocity field over a measurement plane.

At the same time, the development of computational fluid dynamics has made available numerical tools of increasing complexity, ranging from Reynolds-Averaged Navier-Stokes (RANS) simulations, historically the backbone of industrial design, to higher-fidelity approaches such as Large-Eddy Simulation (LES) and Direct Numerical Simulation (DNS), capable of explicitly resolving, without modeling, the turbulent scales of motion responsible for transition [7, 8].

This joint evolution of experimental and numerical tools has made it possible to address

the study of separation and transition phenomena with increasing accuracy, while at the same time laying the groundwork for a new need: that of obtaining not only time-averaged information, but a genuinely space- and time-resolved description of the evolution of the coherent structures responsible for transition, discussed in the following.

The use of DNS and LES simulations resolved in space and time entails, as a counterpart to the physical detail obtained, the generation of extremely large amounts of data: for each time instant the entire three-dimensional flow field must be stored, and the temporal resolution required to properly capture the coherent structures responsible for transition typically results in a number of snapshots on the order of thousands. The mere storage and extraction of physically relevant information from datasets of this size represents today a challenge no less significant than the simulation itself.

Fully exploiting these aerodynamic loading margins therefore requires the ability to understand and predict the boundary-layer response to adverse pressure gradients. It is necessary to correctly capture the onset of separation and the subsequent laminar-to-turbulent transition mechanisms. Modal decomposition techniques such as Proper Orthogonal Decomposition (POD) [9, 10] and Dynamic Mode Decomposition (DMD) [11] address the former need, while high-performance parallel computing (HPC), typically based on distributed-memory architectures and programming paradigms such as the Message Passing Interface (MPI)[12, 13], addresses the latter.

While the physical mechanisms discussed above are most directly observed on the suction side of highly loaded turbine blades, their fundamental features, namely the onset of separation under an adverse pressure gradient and the subsequent laminar-to-turbulent transition, can be studied in a canonical, geometry-independent configuration: a flat plate boundary layer subjected to a controlled adverse pressure gradient. This simplified setting isolates the essential physics of interest from the additional complexity introduced by blade curvature, secondary flows, and rotor-stator interaction, while remaining directly representative of the near-wall flow behaviour relevant to turbomachinery design.The dataset analysed in this thesis is therefore a DNS of this canonical configuration.

It is within this context that the present thesis work is framed, aiming at the development of a parallel framework for the application of DMD to large-scale DNS datasets describing a boundary layer flow with separation, implemented on the Leonardo high-performance computing cluster at CINECA.

The remainder of this thesis is organised as follows. Chapter 2 reviews the relevant

literature on boundary layer transition and separation, with particular focus on the coexistence of streaky structures and the Kelvin-Helmholtz instability within laminar separation bubbles, and on modal decomposition techniques employed for their identification. Chapter 3 introduces the theoretical background underlying the methodology developed in this work, covering Dynamic Mode Decomposition and its foundation in Koopman operator theory, together with the spectral analysis tools employed for the wavenumber decomposition of the velocity field. Chapter 4 describes the architecture and implementation of the distributed parallel DMD framework, including the fundamentals of high-performance distributed-memory computing, the strategy adopted for data redistribution across MPI ranks, and the algorithmic details of the spectral and modal decomposition pipeline. Chapter 5 presents the validation of the implemented framework, both in terms of numerical correctness against a serial reference implementation and an established benchmark from the literature, and in terms of computational performance, through a strong scaling analysis conducted on the Leonardo cluster.

2 Literature Review

2.1 Boundary Layer Fundamentals

A fundamental concept in the study of real fluids is that of the boundary layer, first introduced by Ludwig Prandtl in 1904 [14]; according to this theory, when a fluid flows past a wall, two distinct regions can be identified: one, sufficiently far from the wall, where viscous effects are negligible and the flow field is well approximated by the Euler equations; the other, close to the wall, where viscous effects are instead predominant and high velocity gradients are observed in the wall-normal direction. This latter region is referred to as the boundary layer, conventionally defined as the portion of fluid in which the velocity reaches 99% of the free-stream asymptotic value.

Depending on the geometry and the streamwise velocity distribution, the boundary layer evolves differently in extent and structure. In the simplest configuration, that of a flat plate exposed to a uniform stream, three successive states can be identified along the streamwise direction: a laminar state, in which the fluid motion is well organized and particles move along parallel trajectories; a transitional state, in which the flow is partly laminar and partly turbulent, exhibiting fluctuations due to the intermittent switching between the two conditions; and finally a turbulent state, characterized by disorganized fluctuations that give rise to vortical structures of different scales, although a thin viscous sub-layer close to the wall remains, where viscous effects are still dominant.

Fig. 2.1 illustrates this evolution schematically, showing the growth of the boundary layer thickness $\delta(x)$ along a flat plate as the flow transitions from the laminar to the turbulent state.

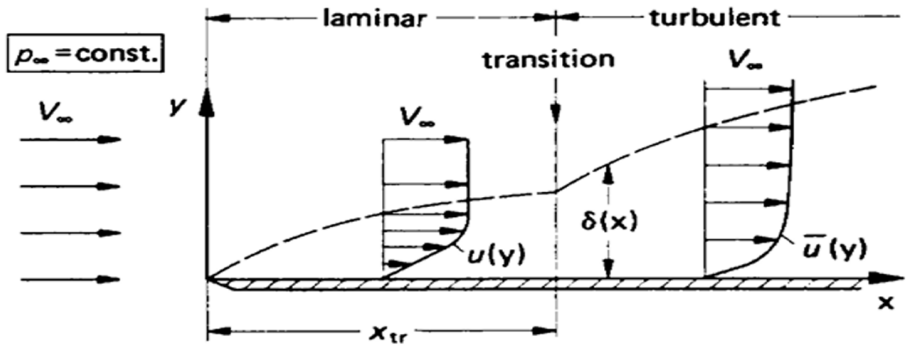


Figure 2.1: Boundary layer evolution over a flat plate, showing the laminar, transitional, and turbulent regions.

The motion of an incompressible viscous fluid is governed, in the most general case, by

the full continuity and Navier-Stokes equations. In the two-dimensional, steady case, these reduce to:

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \qquad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{1}$$

Within the boundary layer, however, an order-of-magnitude analysis of the individual terms, first carried out by Prandtl, shows that gradients in the wall-normal direction are significantly stronger than those in the streamwise direction, and that the wall-normal velocity component v is negligible compared to u. These considerations allow the system to be simplified, neglecting the diffusive term in the x-direction and observing that pressure does not vary in the wall-normal direction ($\partial p/\partial y = 0$, i.e. the pressure within the boundary layer is imposed by the external, inviscid flow), thereby yielding the boundary layer equations [15]:

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\frac{\partial^2 u}{\partial y^2}, \qquad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2}$$

$$2f'''(\eta) + f''(\eta)f(\eta) = 0 \tag{3}$$

to be solved with the boundary conditions $f(0) = 0$, $f'(0) = 0$ and $f'(\infty) = 1$, corresponding respectively to the no-slip condition at the wall and to the asymptotic recovery of the free-stream velocity. It is from this solution, as already mentioned, that the well-known growth of the boundary layer thickness $\delta(x) \sim x^{1/2}$ follows.

For the turbulent boundary layer, the thickness instead grows as $\delta(x) \sim x^{4/5}$ [15], a faster rate compared to the laminar case, a consequence of the more intense momentum exchange between adjacent fluid layers under turbulent conditions.

In the case of turbulent flow, the Navier-Stokes equations take on a more complex formulation, made necessary by the chaotic, multi-scale nature of the flow field. The classical approach to their study consists in the Reynolds decomposition, according to which each quantity of the flow field, for instance a velocity component u, is decomposed into the sum of a time-averaged value $\bar{u}$ and a fluctuating component u':

$$u(t) = \bar{u} + u'(t) \tag{4}$$

Substituting this decomposition into the Navier-Stokes equations and time-averaging the entire system yields the Reynolds-Averaged Navier-Stokes (RANS) equations, which,

in addition to the classical terms already present in the equations for laminar flow, contain an additional term, known as the Reynolds stress tensor, accounting for the momentum exchange induced by turbulent fluctuations [15]. In the two-dimensional case, the tangential component of this tensor, equal to $-\rho\overline{u'v'}$, represents the dominant term in the description of turbulent wall stresses, and it is precisely the correlation between the fluctuations of the streamwise and wall-normal velocity components that governs the production of turbulent kinetic energy within the boundary layer.

This correlation between the different fluctuating velocity components is not only relevant to the closure of the RANS equations, but also constitutes, as will be shown in Chapter 4, a relevant diagnostic indicator within the context of modal decomposition techniques: the joint analysis of the u, v, and w components of a velocity field indeed makes it possible to verify whether an instability identified in one component is intrinsically linked to another, as occurs, for instance, in the lift-up mechanism discussed in Sec. 2.3, in which wall-normal velocity fluctuations are directly responsible for the generation of streamwise streaks.

2.2 Laminar-to-Turbulent Transition Mechanisms

The transition of the boundary layer from the laminar to the turbulent state, referred to as transition, is primarily governed by the local Reynolds number, defined as the ratio between inertial and viscous forces. Once this parameter exceeds a critical value, small disturbances present within the boundary layer begin to amplify, giving rise to localized regions of turbulent flow (turbulent spots) that, by propagating and spreading, eventually lead to a fully turbulent boundary layer [16].

Fig. 2.2 shows an instantaneous streamwise velocity field within the boundary layer, in which the alternating high- and low-speed streaks characteristic of Klebanoff structures are clearly visible along the spanwise direction.

Depending on the boundary conditions and the level of disturbance present in the external flow, the transition mechanism can occur according to three main modes: natural transition, by-pass transition, and separation-induced transition [16].

In natural transition, typical of environments with very low external turbulence intensity, the laminar boundary layer selectively amplifies small disturbances at a well-defined frequency, giving rise to Tollmien-Schlichting waves; these, through a secondary instability process, eventually degenerate into turbulent spots [18]. In by-pass transition, which instead occurs in the presence of high external turbulence or surface roughness, the Tollmien-Schlichting wave mechanism is bypassed: low-frequency distur-

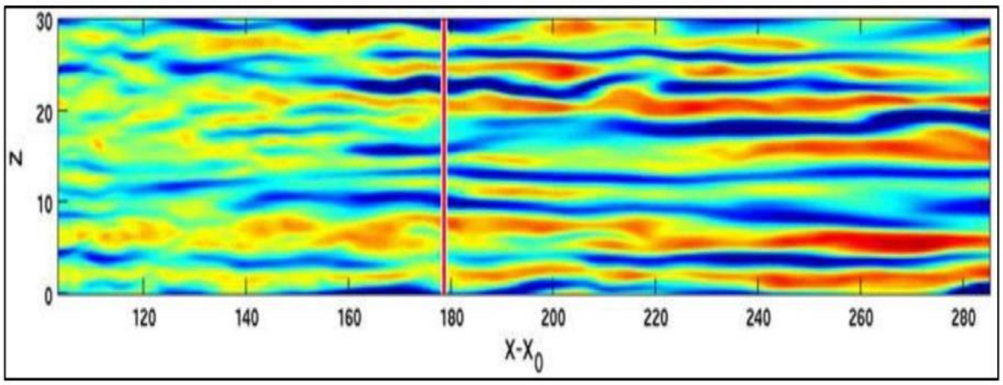


Figure 2.2: Klebanoff streaks: instantaneous streamwise velocity contours showing alternating high- and low-speed streamwise-elongated structures within the boundary layer (adapted from [17]).

bances penetrate directly into the boundary layer, giving rise to streamwise-elongated structures (streaks), whose breakdown directly generates turbulent spots. Finally, separation-induced transition occurs when the laminar boundary layer, subjected to a sufficiently strong adverse pressure gradient, separates from the wall before completing transition; the resulting separated shear layer, intrinsically unstable, transitions rapidly to turbulence, often promoting the reattachment of the boundary layer [19]. Fig. 2.3 compares the natural and by-pass transition routes, highlighting how, under by-pass conditions, the initial stages of the natural transition process (Tollmien-Schlichting waves, spanwise vorticity) are skipped in favour of a direct route to turbulent spot formation.

Given the centrality of this latter mechanism to the present thesis work, separation-induced transition is further examined separately in Sec. 2.4, while Sec. 2.3 briefly recalls the essential features of by-pass transition, relevant to understanding the streaky structures that, as will be shown, can coexist with the instability mechanisms typical of separated flow.

2.3 By-Pass Transition and Streaky Structures

By-pass transition represents the most common transition mechanism in turbomachinery applications, owing to the high level of external turbulence that typically characterises these environments. Its origin lies in the formation and subsequent breakdown of structures elongated in the streamwise direction, known as Klebanoff streaks, that is, filaments of fluid alternating between higher and lower velocity relative to the mean value, distributed along the spanwise direction within the boundary layer [17].

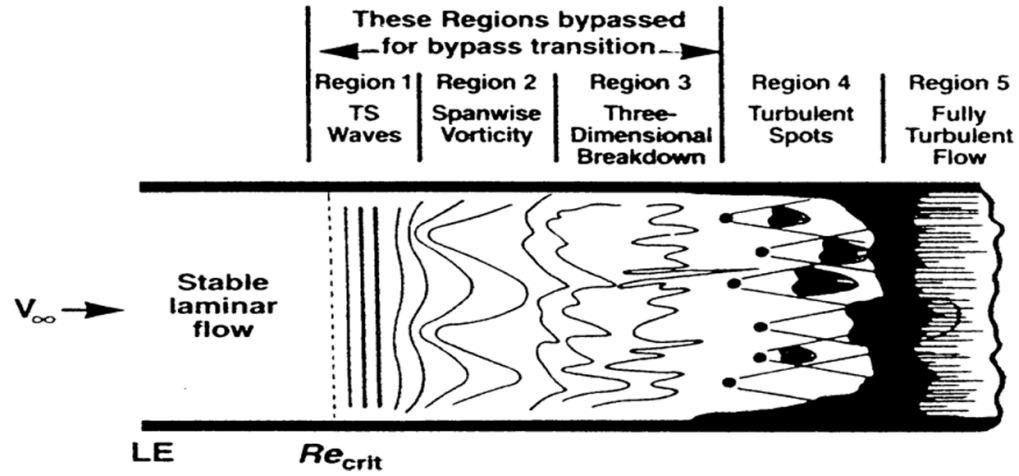


Figure 2.3: Comparison between natural and by-pass transition routes: in by-pass transition, the intermediate stages of the natural mechanism (Tollmien-Schlichting waves, spanwise vorticity, three-dimensional breakdown) are bypassed in favour of a direct formation of turbulent spots (adapted from [19]).

The formation process of these structures originates from the disturbances inherently present in the free-stream flow. Since the boundary layer responds selectively only to the low-frequency components of such disturbances, filtering out the high-frequency components through a mechanism known as shear-sheltering, the low-frequency disturbances that penetrate the boundary layer are amplified through an algebraic transient-growth mechanism, known as the lift-up mechanism, giving rise to the observed streaks [20].

The streaks thus formed are subject to an instability process that determines their breakdown, with the consequent generation of turbulent spots. Two main instability mechanisms have been identified: a sinuous one, in which the streak undergoes a wave-like motion in the spanwise plane, and a varicose one, characterised by successive contractions and expansions of the structure in all directions. In the case of sinuous instability, the low-speed streak undergoes a lift-up and ejection from the wall, giving rise to a secondary vortical structure known as a hairpin vortex, typical of the turbulent boundary layer. The subsequent formation of further hairpin vortices induces normal and spanwise velocity fluctuations that sustain turbulent activity [20].

2.4 Separated-Flow Transition and Kelvin-Helmholtz Instability

When the adverse pressure gradient imposed on the flow is sufficiently strong, the laminar boundary layer is unable to remain attached to the wall and separates, even

before completing transition to turbulence. A recirculating flow region is thus formed, bounded by the separation and reattachment positions, known as a Laminar Separation Bubble (LSB), characterised by low wall shear stress levels and an approximately constant static pressure along its extent [21].

Fig. 2.4 shows the time-averaged structure of a laminar separation bubble, highlighting the laminar shear layer upstream of separation, the transition region within the bubble, and the resulting turbulent shear layer downstream of reattachment.

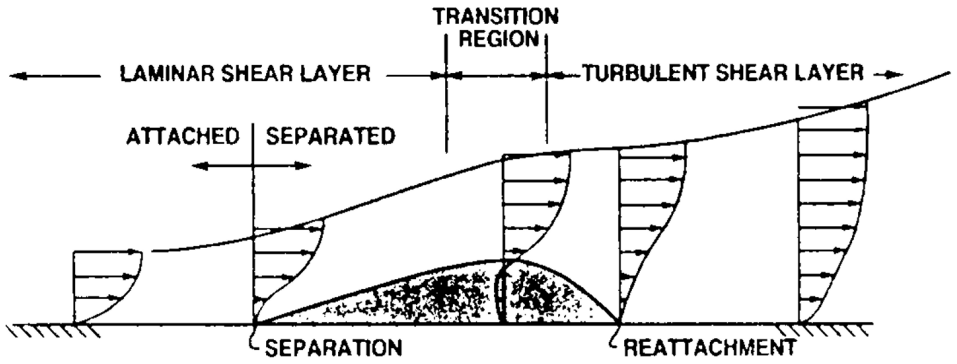


Figure 2.4: Time-averaged structure of a laminar separation bubble (adapted from [21]) .

The flow field around laminar separation bubbles has often been investigated mainly through PIV measurements. Its statistical representation has also relied on modal decompositions such as POD and DMD. However, the evolution of computational fluid dynamics has made it possible to study these phenomena numerically as well, through high-fidelity simulations. In particular, the use of Direct Numerical Simulation (DNS) allows the entire spectrum of turbulent scales involved in transition to be captured without any modelling, offering a level of spatio-temporal detail unattainable by experimental means, albeit at a considerably higher computational cost. By jointly applying POD and DMD to DNS datasets describing laminar separation bubbles, it has been shown that, for moderate-to-high levels of free-stream turbulence, the inviscid Kelvin-Helmholtz instability and the low-frequency Klebanoff modes interact, contributing simultaneously to the transition of the separated shear layer; the energetically dominant POD components are indeed found to describe both the K-H vortices and the low-frequency streaky structures induced by free-stream turbulence, with higher-order modes exhibiting a mixed nature, preserving the spanwise wavelength characteristic of the streaks together with a streamwise wavelength close to that of the K-H vortices [22]. This kind of numerical evidence, complementary to the experimental evidence discussed above, further reinforces the generality of the streak/K-H roll interaction mechanisms

described in this chapter, regardless of the experimental or numerical nature of the analysed dataset, an aspect of direct relevance to the present thesis, which is based entirely on a DNS dataset.

Boundary layer separation gives rise, within the recirculating region, to an inflectional velocity profile, a necessary condition for the onset of an inviscid instability known as the Kelvin-Helmholtz instability. Unlike the natural or by-pass transition mechanisms discussed previously, which rely on viscous instabilities, the Kelvin-Helmholtz instability acts directly on the separated shear layer, exponentially amplifying velocity disturbances within a frequency range well predicted by linear stability theory [23]. The amplification process leads, after an initial linear stage, to disturbance saturation near the position of maximum bubble displacement, where large-scale coherent vortical structures are observed to form, generated by the roll-up of the separated shear layer [24].

For low levels of external turbulence, these vortices retain a substantially two-dimensional nature and are convected downstream at a shedding frequency coinciding with that of the most amplified disturbances in the shear layer. Moving further downstream, however, the vortices begin to exhibit spanwise distortion, eventually breaking down into progressively finer-scale structures: it is precisely this breakdown process that drives boundary layer transition and, consequently, its turbulent reattachment to the wall [25].

The physical mechanism responsible for the three-dimensional breakdown of the initially two-dimensional Kelvin-Helmholtz vortices deserves specific attention. Although the roll-up of the separated shear layer gives rise, to a first approximation, to vortical structures aligned along the spanwise direction, these vortices do not remain two-dimensional indefinitely. Moving further downstream, they develop a periodic modulation along the spanwise direction, whose origin has been attributed to a secondary instability of elliptic nature. This mechanism, well known within the field of vortex dynamics, acts when a vortex core with an elliptical, rather than circular, cross-section develops a three-dimensional, wave-like deformation along its own axis. This deformation typically has a characteristic wavelength that is an integer multiple of the streamwise wavelength characterising the shedding of the vortices themselves. This mechanism provides a direct physical explanation for the quantitative result discussed in Sec. 2.5: in cases where the spanwise periodicity observed in the breakup region collapses onto the same characteristic value as the streamwise periodicity of the Kelvin-Helmholtz vortices, the observed phenomenon can indeed be traced back to the elliptic deformation of the vortex core, rather than to a streaky mechanism. The consistency between the spanwise wavelength of the deformation and the streamwise

wavelength of the shedding therefore constitutes further evidence supporting the purely hydrodynamic origin, not necessarily linked to the presence of external disturbances, of the three-dimensional breakdown observed under conditions of low free-stream turbulence [26, 27].

Fig. 2.5 shows instantaneous PIV snapshots acquired in wall-parallel planes at different distances from the wall, illustrating how streaky structures dominate the flow close to the wall, while Kelvin-Helmholtz rolls and their spanwise breakdown are captured in the plane located farther from it.

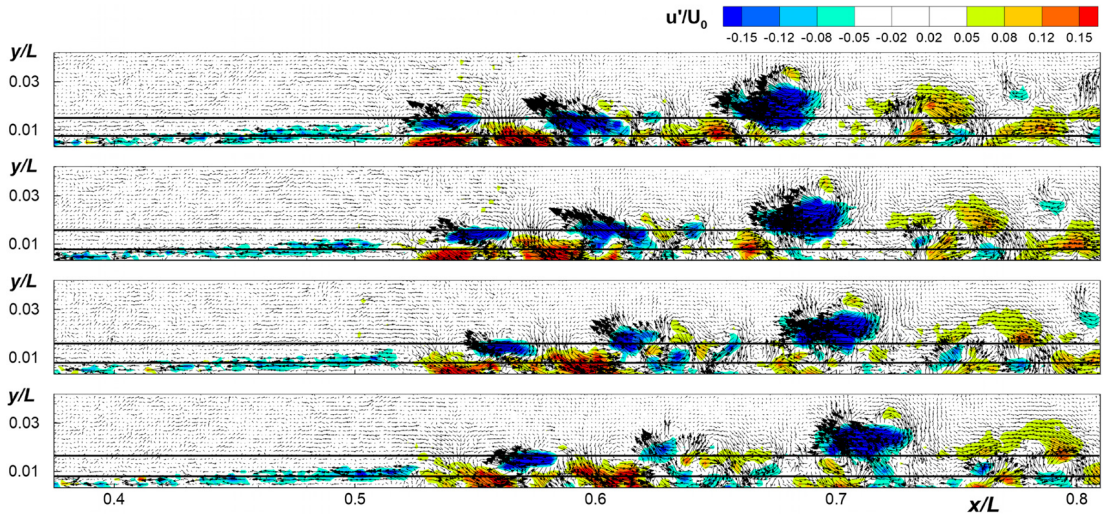


Figure 2.5: PIV snapshots acquired in the wall-parallel plane located close to (left) and far from (right) the wall, showing streaky structures and Kelvin-Helmholtz rolls, respectively (reproduced from [28]).

The behaviour of the separation bubble is strongly influenced by the main flow parameters, in particular the Reynolds number and the external turbulence intensity, which act by differently modifying the coherent structures responsible for transition, as discussed in Sec. 2.5. A further aspect of particular relevance, extensively documented in the literature, concerns the possible coexistence, within the same separation bubble, of the Kelvin-Helmholtz instability mechanisms just described with the streaky structures typical of by-pass transition discussed in Sec. 2.3: for sufficiently high levels of external turbulence, streaks generated in the attached portion of the boundary layer can indeed propagate into the separated region as well, interacting with the Kelvin-Helmholtz vortices and inducing a spanwise distortion that anticipates their breakdown [28].

Based on the relative position between the onset of transition and the separation location, the literature further classifies different modes of separation-induced transition, distinguishing between a transitional mode, in which transition already begins within

the still-attached boundary layer, and a laminar separation mode, in which transition occurs entirely within the bubble, further subdivided, depending on the extent of the bubble itself, into a short mode and a long mode [19].

2.5 Effects of Reynolds Number, Free-Stream Turbulence and Pressure Gradient

The main flow parameters governing the evolution of the laminar separation bubble, namely the Reynolds number, the free-stream turbulence intensity, and the imposed adverse pressure gradient, act by substantially modifying both the geometrical dimensions of the bubble and the nature of the coherent structures responsible for its transition, as extensively documented in the literature [27].

As the Reynolds number increases, the boundary layer thickness at the separation position decreases, resulting in a reduction of both the length and the height of the bubble. This reduction in thickness at separation is directly reflected in the characteristic wavelength of the Kelvin-Helmholtz vortices generated by the roll-up of the shear layer, which becomes shorter as the Reynolds number increases. The free-stream turbulence intensity instead acts through a different mechanism: an increase in turbulence intensity anticipates transition, thereby reducing the extent of the bubble, without however significantly altering the boundary layer thickness at separation, and hence the characteristic wavelength of the Kelvin-Helmholtz vortices [27].

Free-stream turbulence also modifies the nature of the dominant structures within the bubble. At low turbulence levels, transition is found to be governed almost exclusively by the inviscid Kelvin-Helmholtz instability, whereas as free-stream turbulence increases, streaky structures appear in the forward part of the bubble, whose interaction with the Kelvin-Helmholtz vortices anticipates their breakdown.

A result of particular interest, which effectively summarises the combination of effects just described, concerns the behaviour of the characteristic wavelengths once appropriately non-dimensionalised. Rescaling both the spanwise wavelength of the streaks and the streamwise wavelength of the Kelvin-Helmholtz vortices with respect to the boundary layer momentum thickness computed at the separation position, these wavelengths have been observed to collapse onto similar values, equal to $\lambda_z/\theta_{sep} \approx 22$ for the spanwise periodicity and $\lambda_x/\theta_{sep} \approx 35$ for the streamwise one, regardless of the Reynolds number and the free-stream turbulence level considered, provided that both families of structures are present in the flow. In cases where free-stream turbulence is sufficiently low to suppress the formation of streaks, and the flow is dominated ex-

clusively by Kelvin-Helmholtz vortices, the spanwise periodicity also collapses onto the same characteristic value as the streamwise periodicity of the vortices themselves. These findings confirm that, under these conditions, the spanwise spacing observed in the breakup region is itself a direct consequence of the three-dimensional elliptic instability of the Kelvin-Helmholtz vortices, rather than of a streaky mechanism [27].

Beyond these observations specifically referring to the separation bubble, the effect of free-stream turbulence and pressure gradient on boundary layer transition has historically been summarised, for the more general case of by-pass transition, through empirical correlations. For free-stream turbulence intensity alone, the Reynolds number based on the momentum thickness at the onset of transition, $Re_{\theta s}$, has been related to the turbulence intensity Tu (expressed as a percentage) through Mayle's correlation [16]:

$$Re_{\theta s} = 400\, Tu^{-5/8} \tag{5}$$

which shows how an increase in free-stream turbulence reduces the critical value of $Re_{\theta s}$, thereby anticipating the onset of transition.
To also account for the effect of the pressure gradient, typically expressed through the dimensionless acceleration parameter

$$\lambda_\theta = \frac{\theta^2}{\nu}\frac{dU_e}{dx} \tag{6}$$

Abu-Ghannam and Shaw proposed a more general correlation [29]:

$$Re_{\theta s} = 163 + \exp\left[F(\lambda_\theta) - \frac{F(\lambda_\theta)}{6.91}Tu\right] \tag{7}$$

where $F(\lambda_\theta)$ is a polynomial function of λ_θ, distinct depending on whether the pressure gradient is favourable ($\lambda_\theta > 0$) or adverse ($\lambda_\theta < 0$). This correlation highlights a relevant aspect: for a given turbulence level, the effect of an adverse pressure gradient in promoting transition is more pronounced than the ability of a favourable pressure gradient to delay it, and moreover the influence of the pressure gradient progressively diminishes as the free-stream turbulence intensity increases, a behaviour qualitatively consistent with what was observed for separation-induced transition discussed previously.

Fig. 2.6 shows the momentum-thickness Reynolds number at transition onset predicted

by Eq. 7 as a function of the acceleration parameter λ_θ, for different levels of free-stream turbulence: the curves clearly illustrate how the destabilising effect of an adverse pressure gradient is asymmetric with respect to the stabilising effect of a favourable one, and how this asymmetry is progressively damped as Tu increases.

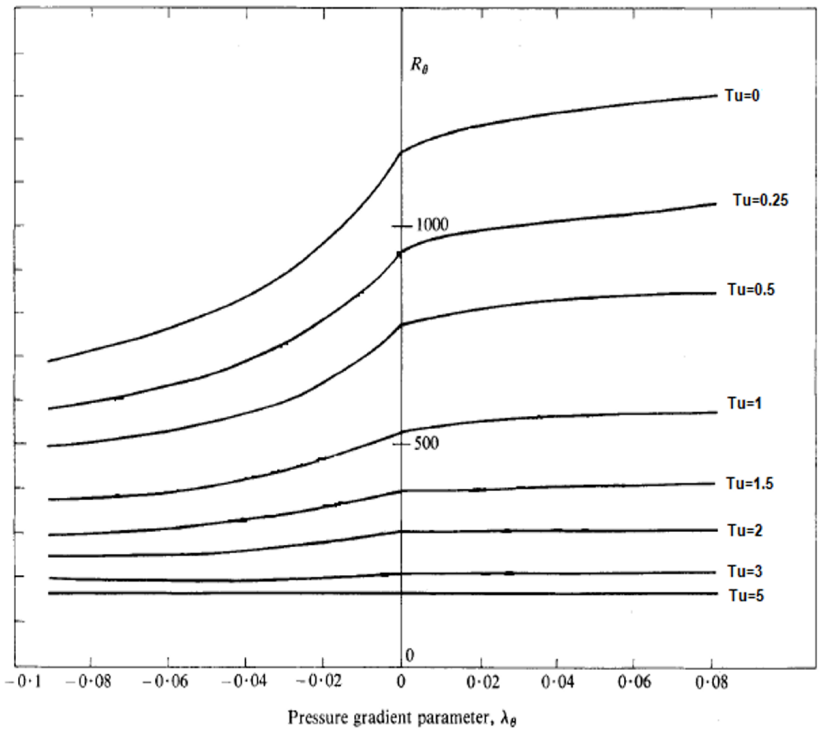


Figure 2.6: Momentum thickness Reynolds number computed at the transition onset for different values of λ_θ and free-stream turbulence, according to the correlation of [29].

A further relevant correlation concerns the prediction of the laminar separation position itself: according to Thwaites' criterion, separation occurs when the Reynolds number based on the momentum thickness and the acceleration parameter λ_θ satisfy the relation [30]:

$$Re_{\theta,s}^2 \, K \leq -0.0082 \tag{8}$$

where the acceleration parameter K is defined as

$$K = \frac{\lambda_\theta}{Re_\theta^2} \tag{9}$$

This criterion, although derived for the laminar boundary layer under general conditions, underlines how the separation position is uniquely determined, for a given

pressure gradient profile, by the upstream history of the boundary layer along the surface, rather than by the local position alone.

Finally, the adverse pressure gradient imposed on the flow acts in a manner analogous to a reduction of the Reynolds number, promoting further boundary layer thickening and, consequently, an earlier separation and a larger bubble extent; its effect in promoting transition, however, becomes progressively less pronounced as the free-stream turbulence intensity increases, consistently with what has just been discussed for the more general case of by-pass transition.

2.6 Modal Decomposition Techniques for Coherent Structure Identification

The coexistence, within the separation bubble, of instability mechanisms of different nature, such as the streaky structures typical of by-pass transition and the Kelvin-Helmholtz vortices generated by the inviscid instability of the separated shear layer, often makes their clear identification impossible through simple visual inspection of instantaneous velocity fields or time-averaged statistical quantities. For this reason, the scientific community has made extensive use, over the past decades, of modal decomposition techniques, capable of objectively extracting the dominant coherent structures from a large dataset.

Proper Orthogonal Decomposition (POD) represents the historically most widespread modal decomposition technique in fluid dynamics. Given a sequence of N realisations of the flow field, for instance N PIV or DNS snapshots, arranged in the columns of a snapshot matrix U, the snapshot method introduced by Sirovich allows the POD temporal coefficients $v(t)$ to be obtained by solving the eigenvalue problem

$$CX = \Lambda X \tag{10}$$

where

$$C = U^T U \tag{11}$$

represents the temporal cross-correlation matrix, Λ is the diagonal matrix of real eigenvalues, and X is the matrix of orthogonal eigenvectors of C. The spatial POD modes Φ are then obtained by projection:

$$\Phi = UX\Lambda^{-1/2} \tag{12}$$

and are ranked according to the energy content associated with their respective eigenvalue, thereby guaranteeing a low-rank representation that concentrates most of the turbulent kinetic energy into a limited number of dominant modes [10].

When applied to the entire measurement domain, however, POD tends to be dominated by the highest-energy structures, typically the Kelvin-Helmholtz vortices in the breakup region, thereby masking the less energetic, yet still relevant, dynamics, such as the streaks in the forward part of the bubble. To overcome this limitation, POD can be applied separately to spatial partitions of the domain, thereby isolating the different families of dominant structures in each flow region. A further extension of the technique, known as Extended POD (E-POD), additionally allows the velocity fluctuations observed in a given region of the domain to be projected onto the modal basis computed in a different region, making it possible to quantify the degree of correlation between apparently distinct dynamics, such as the bursting events observed close to the wall and the Kelvin-Helmholtz structures dominating far from it [27].

Fig. 2.7 shows POD modes computed separately on wall-parallel planes located close to and far from the wall, illustrating how the modal decomposition, once applied to spatially partitioned data, is able to isolate streaky-structure-dominated modes from Kelvin-Helmholtz-related ones.

An intrinsic limitation of POD, however, lies in its purely statistical nature: while providing an energy-based hierarchy of coherent structures, it is unable to associate an explicit temporal dynamics, in terms of oscillation frequency and growth or decay rate, with each mode. Dynamic Mode Decomposition (DMD), introduced by Schmid [11], fills this gap, providing a linear approximation of the system dynamics directly from data, without requiring any prior knowledge of the equations governing the observed phenomenon. Unlike POD, each DMD mode is associated with a precise complex eigenvalue, whose phase and modulus respectively provide the oscillation frequency and growth rate of the corresponding structure, making the technique particularly suited to identifying the instabilities responsible for transition.

In the classical formulation of DMD, discussed in detail in Chapter 3, the columns of the snapshot matrix X correspond to successive time instants, separated by a constant interval Δt, and the discrete eigenvalues μ_i obtained from the decomposition are interpreted in terms of the oscillation frequency and the temporal growth or decay rate of the corresponding structure.

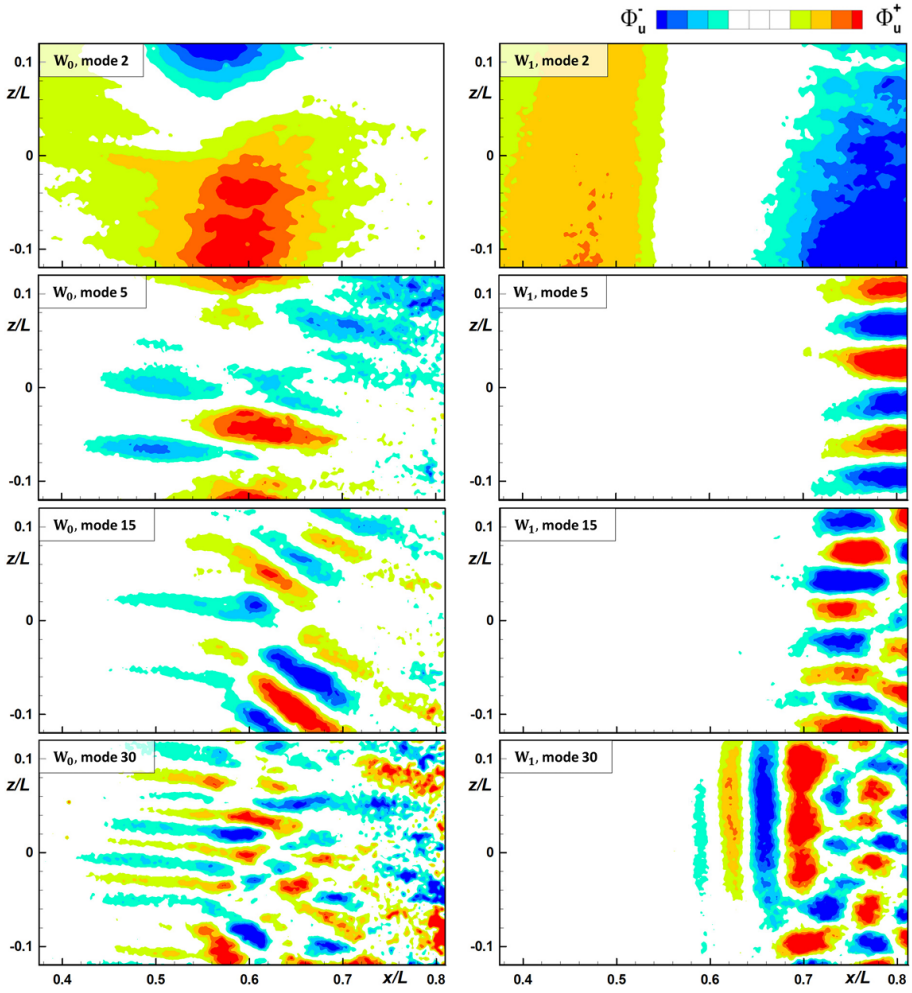


Figure 2.7: POD modes of streamwise velocity computed in the wall-parallel plane located close to (left) and far from (right) the wall (adapted from [27]) .

In the specific context of analysing convectively unstable flows, such as the separated shear layer responsible for the Kelvin-Helmholtz instability, an alternative formulation of the technique, known as spatial DMD, proves more natural: here, the columns of the snapshot matrix no longer represent time instants, but successive positions along the streamwise direction of the domain, separated by a constant spatial step Δx. In this case, the entire DMD formalism remains unchanged, but the physical meaning of the discrete eigenvalues μ_i (or, equivalently, of the continuous eigenvalues λ_i obtained through the relation $\lambda_i = \ln(\mu_i)/\Delta x$) shifts from the temporal domain to the spatial one: the phase of the eigenvalue now provides the characteristic spatial wavelength of the identified structure, rather than its oscillation frequency, while the modulus, or equivalently the real part of the continuous eigenvalue, describes its spatial amplification or attenuation rate along the streamwise direction of the flow, rather than its evolution in time.

This reinterpretation proves particularly effective in characterising the Kelvin-Helmholtz instability, which, as discussed in Sec. 2.4, develops spatially along the separated shear layer, progressively amplifying as disturbances are convected downstream: an eigenvalue with positive real part thus directly identifies the region of the domain in which the instability is growing, while an eigenvalue with negative or null real part respectively signals its attenuation or neutral condition. It is precisely this spatial formulation of DMD, applied incrementally along the streamwise direction, that constitutes the foundation of the three-regime identification procedure described in the following [31].

The direct application of DMD to the entire dataset describing the evolution of a separation bubble is nonetheless problematic, since the flow passes through markedly different dynamical regimes along its evolution: initially stable in the pre-transitional laminar region, then unstable during the linear amplification stage of disturbances, and finally stable again once the fully turbulent condition is reached. Applying DMD over the entire domain in these cases tends to yield a globally stable behaviour, masking the linear instability stage responsible for transition.

To address this issue, a procedure has been proposed that applies DMD incrementally over progressively increasing portions of the streamwise domain, rather than to the entire dataset at once. For each extent k of the domain considered, DMD is applied by solving the least-squares problem

$$\min_S \|V_{k+1} - V_k S\|_F^2 \tag{13}$$

where S is the linear transformation matrix that best approximates the evolution of the

k columns of the data matrix. The average residual r_k of the approximation, computed as the mean Euclidean norm of the columns of the residual matrix

$$R_k = V_{k+1} - V_k S_k \tag{14}$$

is then monitored as k increases. The typical residual trend shows a decreasing segment, corresponding to the pre-transitional laminar regime, followed by an increasing segment, associated with the unstable regime dominated by the Kelvin-Helmholtz instability, and finally a change in slope, marking the onset of the stable turbulent condition. The two switching points k_1 and k_2 between these three regimes are identified automatically: the first as the minimum of the residual curve, the second through a least-squares fit of the residual with two straight lines, identifying the intersection point that minimises the overall mean squared error. By applying DMD exclusively to the portion of the domain identified as unstable, it is possible to accurately extract the eigenvalue associated with the maximum growth rate, and hence the characteristic frequency and wavelength of the Kelvin-Helmholtz instability, in cases where applying the technique over the entire domain would have been unable to isolate any unstable eigenvalue at all [31].

Fig. 2.8 shows the Ritz eigenvalues with the largest real part for each streamwise extent k, plotted in the complex plane: eigenvalues belonging to the stable pre-transitional and turbulent regimes remain close to the unit circle, while those identified within the unstable region (highlighted in red) clearly migrate toward a well-defined location in the phase space, confirming the robustness of the switching-point identification described above.

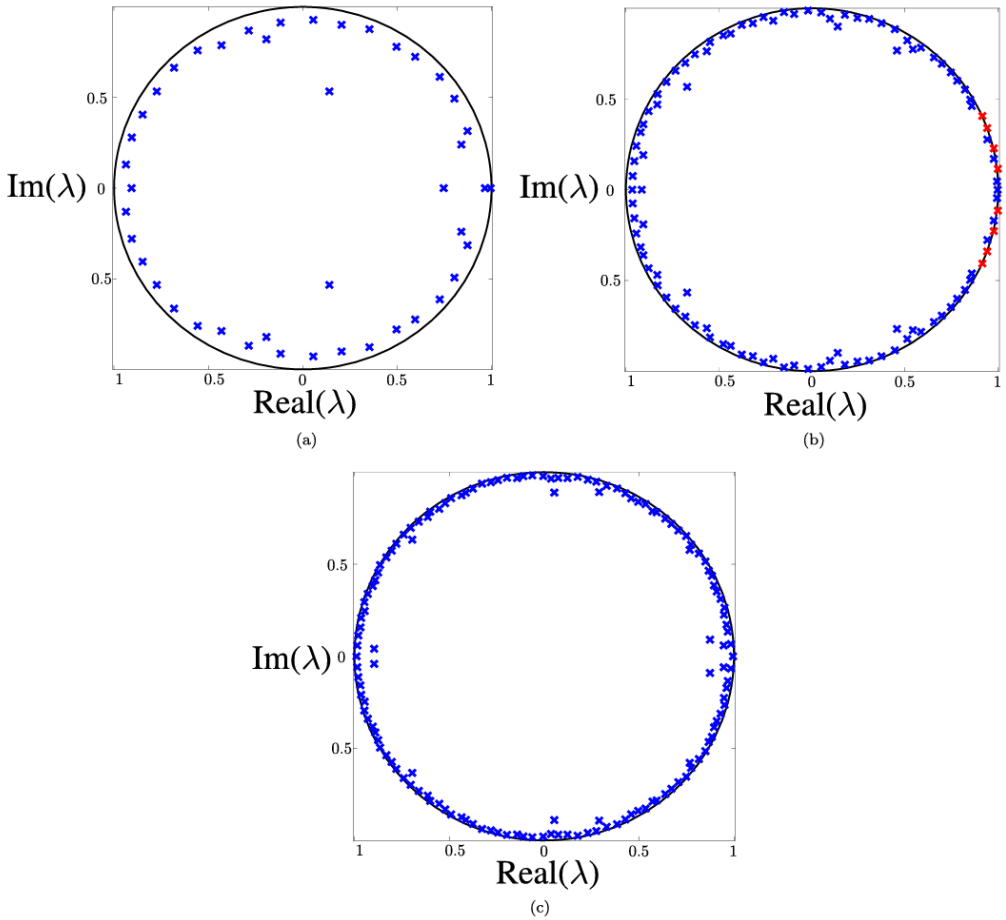


Figure 2.8: Ritz eigenvalues with the maximum real part for each streamwise extent k, colour-coded by flow regime (stable regime in blue/green/black, unstable regime in red) (adapted from [31]).

Fig. 2.9 shows the trend of the DMD residual as a function of the streamwise extent of the analysed dataset, used to automatically identify the boundaries between the stable, unstable, and turbulent flow regimes.

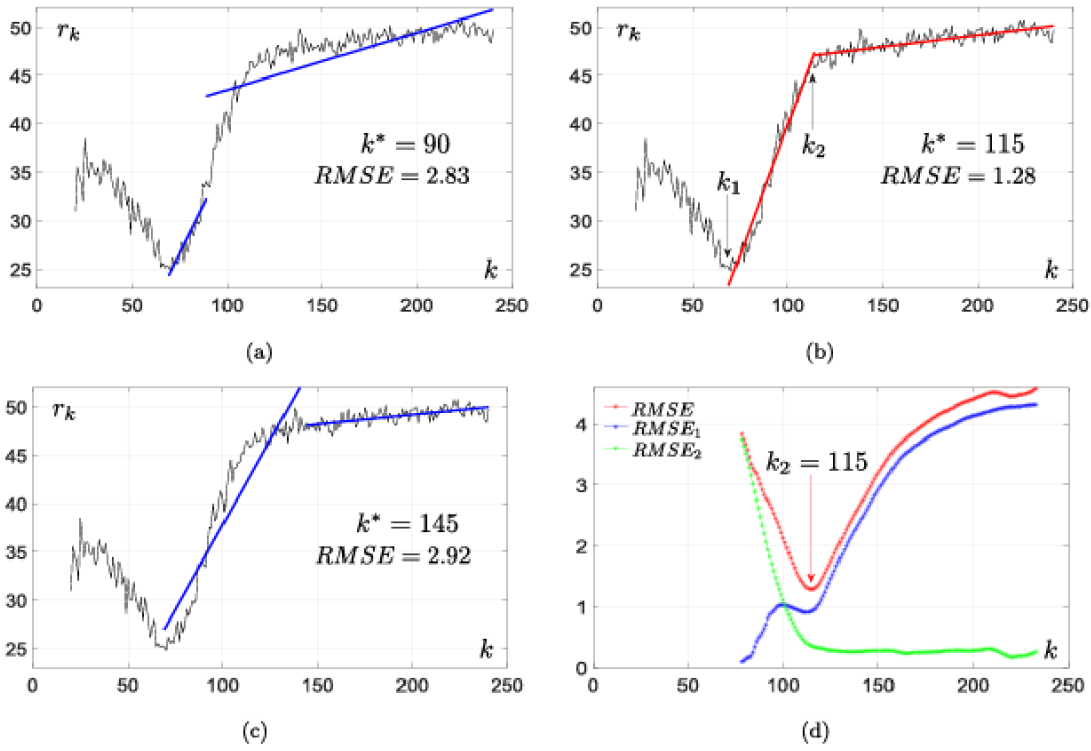


Figure 2.9: Residuals of the incremental DMD procedure for different values of the switching parameter k^* (adapted from [31]).

A further critical aspect of standard DMD concerns its ability to correctly identify the unstable eigenvalues associated with the transition process, especially under conditions of low free-stream turbulence and high Reynolds number. In such cases, it has been shown that an extension of the technique based on Koopman operator theory, obtained by augmenting the state vector with nonlinear observables such as the Reynolds shear stress term uv, is able to systematically identify both the frequency and the characteristic wavelength of the Kelvin-Helmholtz vortices more accurately, in situations where standard DMD fails to extract any unstable eigenvalue at all [26].

Fig. 2.10 compares the spatiotemporal modes obtained from standard DMD and from the Koopman-based approach, showing how the latter more accurately captures the correct Kelvin-Helmholtz shedding frequency where standard DMD fails.

It is worth noting that the procedures just described, while effective in isolating the instabilities of interest, share an implicit limitation: they were conceived and validated on datasets of limited size, at most a few thousand snapshots relating to a single measurement plane or a single two-dimensional simulation, and applied serially, without requiring any distributed computing infrastructure. Such an approach is fully adequate

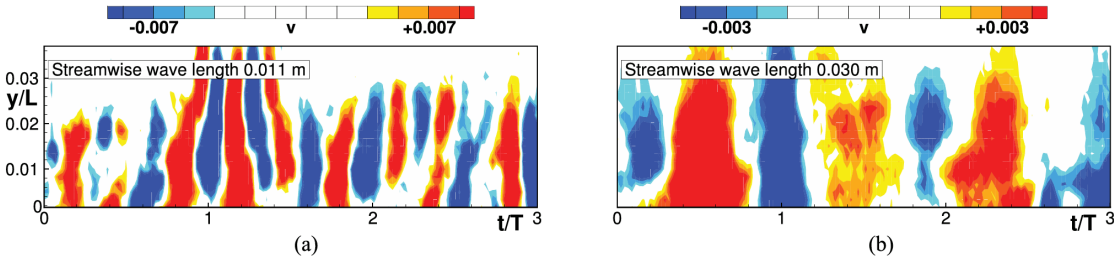


Figure 2.10: Comparison between DMD (left) and Koopman-based (right) modes linked to the most unstable eigenvalue (adapted from [26]).

when the dataset to be analysed can be entirely loaded and processed on a single computing machine. In the case of large-scale three-dimensional DNS simulations, however, such as the one analysed in the present thesis, the resulting data volume makes a serial approach simply infeasible, both in terms of available memory and required computation time. It is precisely from this need, complementary to the modal decomposition techniques reviewed in this section, that the necessity arises to develop a parallel computational framework, capable of applying Dynamic Mode Decomposition, following the same incremental domain-subdivision principle described above, to DNS datasets of a size otherwise intractable, distributing the computational load over a distributed-memory HPC infrastructure.

It is precisely within this family of modal decomposition techniques, and in particular within the Exact DMD formulation introduced by Tu et al. [32], discussed in detail in Chapter 3, that the present thesis is framed, its objective being the development of a parallel framework capable of applying this technique to DNS datasets of a size that would otherwise be intractable with a serial approach.

2.7 Closing Remarks

The transition mechanisms reviewed in this chapter, from the formation of streaks in by-pass transition, to the Kelvin-Helmholtz instability in the separated shear layer, up to their possible coexistence and interaction within the same separation bubble, are not of purely academic interest, but find direct application in the design of modern low-pressure turbines, as already anticipated in Chapter 1. The trend toward increasingly aerodynamically loaded blades makes the occurrence of separation bubbles a far from rare phenomenon under real operating conditions, making a thorough understanding of the mechanisms governing their extent and subsequent turbulent reattachment essential.

In this context, the modal decomposition techniques discussed in Sec. 2.6, and in particular Dynamic Mode Decomposition, represent a particularly effective analysis tool, capable of isolating the coherent structures responsible for transition from large experimental or numerical datasets. It is precisely the application of this technique to a large-scale DNS dataset, describing a flat plate flow with boundary layer separation, that constitutes the subject of the present thesis, whose distributed computational framework is described in detail in Chapter 4.

3 Theoretical and Computational Background

3.1 Dynamic Mode Decomposition

Dynamic Mode Decomposition (DMD) is a data-driven modal analysis technique, originally developed by Schmid in the context of experimental and computational fluid dynamics [11], which allows the dominant coherent spatial structures of a dynamical system to be extracted from a sequence of temporal data, together with their temporal dynamics, namely oscillation frequency and growth or decay rate. Unlike purely statistical techniques such as Proper Orthogonal Decomposition (POD), which ranks spatial structures by energy content without providing explicit information on their temporal evolution, DMD associates each spatial mode with a precise dynamical behaviour, making it a particularly effective tool for studying oscillatory phenomena, convective instabilities, and coherent structures in turbulent or transitional flows [32].

The strength of DMD lies in its equation-free nature: the algorithm requires no prior knowledge of the equations governing the system under investigation, operating exclusively on measured or simulated data, in the form of sequences of instantaneous snapshots of the field of interest. This characteristic makes it particularly suited to the analysis of high-fidelity simulation datasets, such as those produced by DNS simulations, where the governing equations are known but their complexity makes a purely data-driven modal reduction approach preferable.

In the remainder of this section, the theoretical foundation of DMD is first introduced, represented by Koopman operator theory (§3.1.1), which justifies its applicability to intrinsically nonlinear systems. The Exact DMD algorithm is subsequently derived (§3.1.2), in the formulation that constitutes the computational core of the code developed in this thesis, followed by a discussion of the physical interpretation of the DMD spectrum (§3.1.3) .

3.1.1 Koopman Operator Theory

Dynamic Mode Decomposition finds its theoretical foundation in Koopman operator theory, a formalism that allows nonlinear dynamical systems to be analysed through tools typical of linear algebra. The central idea, introduced by Bernard Koopman in 1931 [33], consists of shifting attention from the state space of the system, generally nonlinear and of finite dimension, to a space of observables, scalar functions defined on the system state, where the temporal evolution can be described by a linear operator, albeit of infinite dimension.

Consider a discrete-time dynamical system, described by the nonlinear map:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) \tag{15}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ represents the state of the system at time t_k, and $\mathbf{f}$ is, in general, a nonlinear function. Rather than studying the evolution of the state $\mathbf{x}_k$ directly, Koopman theory introduces the space of observables $g : \mathbb{R}^n \to \mathbb{C}$, scalar functions of the state, and defines the Koopman operator $\mathcal{K}$ as the operator acting on such observables according to the relation:

$$\mathcal{K}g(\mathbf{x}_k) = g(\mathbf{f}(\mathbf{x}_k)) = g(\mathbf{x}_{k+1}) \tag{16}$$

The fundamental characteristic of the Koopman operator is its linearity: although the underlying dynamics $\mathbf{f}$ is nonlinear, the operator $\mathcal{K}$ acts linearly on the (infinite dimensional) space of observables, since for any pair of observables g_1, g_2 and scalars α, β the following holds:

$$\mathcal{K}(\alpha g_1 + \beta g_2) = \alpha \mathcal{K} g_1 + \beta \mathcal{K} g_2 \tag{17}$$

This property allows, in principle, the entire apparatus of spectral analysis, namely eigenvalues, eigenfunctions, and modes, to be applied to intrinsically nonlinear systems, at the cost of working in a infinite-dimensional space.

The eigenfunctions of the Koopman operator $\varphi_i(\mathbf{x})$, associated with the eigenvalues λ_i, satisfy the relation:

$$\mathcal{K}\varphi_i(\mathbf{x}) = \lambda_i \varphi_i(\mathbf{x}) \tag{18}$$

If a vector-valued observable $\mathbf{g}(\mathbf{x})$, for instance the state of the system itself, can be expanded in the basis of Koopman eigenfunctions as:

$$\mathbf{g}(\mathbf{x}_k) = \sum_{i=1}^{\infty} \varphi_i(\mathbf{x}_k)\, \mathbf{v}_i \tag{19}$$

where $\mathbf{v}_i$ are the so-called *Koopman modes*, then the temporal evolution of the system can be rewritten as a linear superposition of modes, each characterised by its own oscillation frequency and its own growth or decay rate, encoded in the complex eigenvalue

λ_i.

From a practical standpoint, the exact Koopman operator is generally inaccessible, since it requires knowledge of an infinite-dimensional space of observables. Dynamic Mode Decomposition, as will be shown in the following section, can be interpreted as a finite-rank approximation of the Koopman operator, obtained by choosing the components of the system state itself as observables, or, as in the case of the present thesis, the spectral coefficients of the velocity field [34]. Such an approximation, however limited with respect to the full theoretical formulation, has proven extremely effective in extracting the dominant coherent structures in complex fluid dynamical systems, justifying the widespread use of DMD as an analysis tool in this field [35].

3.1.2 The Exact DMD Algorithm

As anticipated, Dynamic Mode Decomposition can be interpreted as a finite-rank approximation of the Koopman operator, obtained from a finite sequence of measured or simulated data. In this section, the Exact DMD formulation is derived, introduced by Tu et al. [32], which constitutes the algorithmic basis of the code developed in this thesis.

Assume a sequence of $m+1$ snapshots of a vector field is available, sampled at regular time intervals Δt:

$$\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \tag{20}$$

where each vector $\mathbf{x}_k \in \mathbb{C}^n$ represents the state of the system, for instance the velocity field discretised on a spatial grid, at time $t_k = k\Delta t$. The fundamental assumption of DMD is that there exists a linear operator $A \in \mathbb{C}^{n\times n}$, approximately constant in time, describing the evolution of the system between two successive instants [11]:

$$\mathbf{x}_{k+1} \approx A\mathbf{x}_k \tag{21}$$

From the sequence of snapshots, two matrices are constructed, referred to as the data matrix X and the shifted data matrix X':

$$X = \begin{bmatrix} | & | & & | \\ \mathbf{x}_0 & \mathbf{x}_1 & \ldots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}, \quad X' = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_m \\ | & | & & | \end{bmatrix} \tag{22}$$

both of dimension $n \times m$, where X' is obtained from X by shifting each column forward by one time step. The operator A is then defined, in a least-squares sense, as the matrix that best satisfies the relation:

$$X' \approx AX \tag{23}$$

Since in the applications of interest the state dimension n, for instance the number of points of the spatial grid, is typically much larger than the number of available snapshots m, the direct computation of A via the pseudoinverse would be computationally prohibitive, as well as numerically ill-conditioned. DMD circumvents this issue by computing not the full operator A, but rather its projection onto a reduced-dimensional subspace.

The first step consists in computing the Singular Value Decomposition (SVD) of the matrix X:

$$X = U\Sigma V^* \tag{24}$$

where $U \in \mathbb{C}^{n\times n}$ and $V \in \mathbb{C}^{m\times m}$ are unitary matrices whose columns contain, respectively, the dominant spatial and temporal modes of the system, $\Sigma \in \mathbb{R}^{n\times m}$ is a diagonal matrix containing the singular values in decreasing order, and the asterisk denotes the conjugate transpose. A rank-r truncation is then applied, retaining only the first r columns of U and V and the first r singular values, yielding the truncated matrices $U_r \in \mathbb{C}^{n\times r}$, $\Sigma_r \in \mathbb{R}^{r\times r}$ and $V_r \in \mathbb{C}^{m\times r}$. The value of r can either be fixed a priori, or determined dynamically based on an energetic criterion, retaining only the modes whose singular value is not negligible with respect to the dominant singular value σ_1.

Exploiting the truncated decomposition, the reduced operator $\tilde{A} \in \mathbb{C}^{r\times r}$ can be constructed, obtained by projecting the operator A onto the rank-r subspace identified by the columns of U_r [11]:

$$\tilde{A} = U_r^* X' V_r \Sigma_r^{-1} \tag{25}$$

The matrix $\tilde{A}$ therefore represents the operator A restricted to the dominant subspace, and shares its non-zero eigenvalues. Diagonalising $\tilde{A}$ yields the eigenvalues μ_i and the corresponding eigenvectors $\mathbf{w}_i$, collected in the matrix W:

$$\tilde{A}W = W\Lambda \tag{26}$$

where $\Lambda = \text{diag}(\mu_1, \mu_2, \ldots, \mu_r)$. The eigenvalues μ_i, in general complex, constitute the discrete DMD spectrum and encode the dynamical behaviour of the system, as will be further discussed in the following section.

Finally, the exact DMD modes $\Phi \in \mathbb{C}^{n\times r}$, namely the spatial structures associated with each eigenvalue, are not obtained as a simple projection of the eigenvectors W onto physical space (an approach known as projected DMD), but rather through the following reconstruction, which guarantees that the modes belong to the actual data space rather than merely to the subspace identified by the SVD [32]:

$$\Phi = X' V_r \Sigma_r^{-1} W \tag{27}$$

This formulation, accordingly referred to as Exact DMD, has the advantage of providing modes that are exact eigenvectors of the operator A (or of a full-rank extension thereof), rather than mere approximations within the reduced subspace, and is the formulation adopted in the implementation described in Chapter 4.

Once the modes Φ are obtained, the temporal evolution of the system can be reconstructed as a linear superposition of the modes, weighted by a vector of initial amplitudes $\mathbf{b} \in \mathbb{C}^r$:

$$\mathbf{x}_k \approx \Phi \Lambda^k \mathbf{b} \tag{28}$$

The amplitude vector $\mathbf{b}$ is typically determined by solving the least-squares problem that best approximates the first snapshot of the sequence:

$$\mathbf{b} = \Phi^\dagger \mathbf{x}_0 \tag{29}$$

where $\Phi^\dagger$ denotes the Moore-Penrose pseudoinverse. This approach, although more computationally demanding than the direct computation of the pseudoinverse, generally offers greater numerical stability in cases where the mode matrix Φ is ill-conditioned.

Fig. 3.1 summarises the complete Exact DMD workflow described above, from the construction of the snapshot matrices to the reconstruction of the system dynamics.

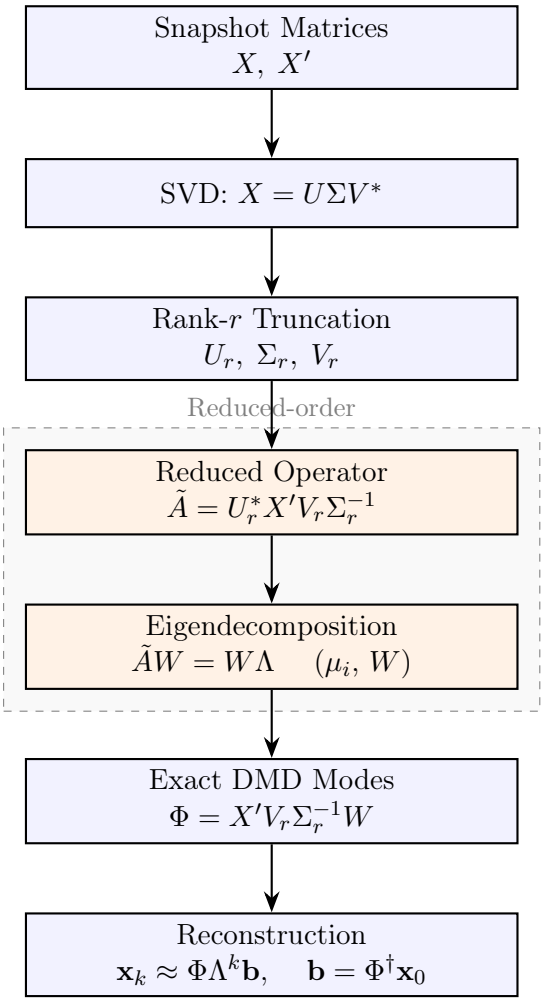


Figure 3.1: Flow diagram of the Exact DMD algorithm, from snapshot matrix construction to system reconstruction.

3.1.3 Physical Interpretation of DMD Spectra

Once the discrete eigenvalues μ_i and the spatial modes Φ_i have been obtained through the Exact DMD algorithm, it is essential to interpret their physical meaning, since it is precisely this interpretation that makes DMD a tool for physical analysis rather than a mere exercise in linear algebra [32, 36].

Discrete Eigenvalues and Stability

Each eigenvalue μ_i, in general complex, can be written in polar form as:

$$\mu_i = |\mu_i| \, e^{i\theta_i} \tag{30}$$

where the modulus $|\mu_i|$ and the phase θ_i encode, respectively, the information on growth or decay and on the oscillation frequency associated with mode Φ_i. In particular, the modulus of the discrete eigenvalue determines the stability of the corresponding mode: a mode with $|\mu_i| < 1$ decays in time, a mode with $|\mu_i| > 1$ grows, while a mode with $|\mu_i| = 1$ is neutral, i.e. it oscillates without amplification or damping. The unit circle in the complex plane therefore constitutes the natural reference against which the stability of the discrete spectrum is assessed: eigenvalues lying exactly on the unit circle, or in its immediate vicinity, typically correspond to coherent structures that persist in time, as in the classical case of periodic vortex shedding behind a bluff body [11].

Transition to the Continuous Spectrum

When the data are sampled at a known and constant time interval Δt, the discrete eigenvalues μ_i can be converted into continuous eigenvalues λ_i, through the relation:

$$\lambda_i = \frac{\ln(\mu_i)}{\Delta t} \tag{31}$$

The continuous eigenvalues $\lambda_i = \sigma_i + i\omega_i$ offer a more immediate physical interpretation: the real part σ_i represents the exponential growth or decay rate of the mode in physical time, while the imaginary part ω_i represents its angular frequency, directly related to the physical oscillation frequency $f_i = \omega_i/2\pi$. In this representation, the stability criterion translates into the position of the eigenvalue with respect to the imaginary axis of the complex plane: modes with $\sigma_i < 0$ are stable (decaying), modes with $\sigma_i > 0$ are unstable (growing), while modes with $\sigma_i = 0$ are neutrally stable [36].

It is important to note that this conversion presupposes knowledge of a well-defined physical time interval Δt between one snapshot and the next. Where the data sequence does not represent a temporal evolution but, as in the case of the implementation de-

scribed in Chapter 4, a succession of spatial positions along a direction of the domain, the eigenvalues μ_i retain an analogous meaning but referred to the spatial development of the structures, describing spatial amplification or attenuation rates rather than temporal ones.

Spatial Modes and Physical Structure

While the eigenvalues μ_i (or λ_i) describe the dynamical behaviour of the system, the modes Φ_i describe its spatial structure. Each mode Φ_i is, in general, a complex vector defined over the entire discretised spatial domain; its real and imaginary parts represent two snapshots of the same spatial structure, shifted by a quarter wavelength with respect to one another, consistent with the travelling-wave nature typical of oscillatory modes. The overall amplitude of the mode, together with the weight b_i determined during the reconstruction phase, quantifies its relative energetic contribution with respect to the other identified structures.

Criterion for Selecting Dominant Modes

Since the number of modes r extracted from the decomposition can be significant, especially in the presence of high-dimensional datasets such as those originating from DNS simulations, it is common practice to rank the modes according to their physical relevance, generally based on the amplitude $|b_i|$, the growth rate σ_i, or a combination of the two. Modes with higher energy and a growth rate close to zero typically correspond to the dominant, persistent coherent structures in the flow, whereas modes with negligible energy or strong damping are generally associated with numerical noise or dissipative scales of lesser physical interest, an aspect that justifies the introduction of spectral truncation criteria, as discussed in Chapter 4 in relation to the selection of the dominant wavenumbers.

3.2 Spectral Analysis Fundamentals

Frequency-domain analysis represents an essential tool for the study of oscillatory phenomena and coherent structures typical of turbulent or transitional flows. Within the algorithm developed in this thesis, the Fourier transform is employed along the spanwise direction of the domain, in order to decompose the velocity field into the wavenumber domain k_z, as a preliminary step prior to the application of Dynamic Mode Decomposition.

In this section, the theoretical foundations of the Discrete Fourier Transform (DFT) and of its optimised variant for real-valued signals, the Real FFT, are recalled (§3.2.1), together with the resulting Hermitian symmetry property and Parseval's Theorem,

which underlies the correct energetic normalisation of the spectrum (§3.2.2).

3.2.1 Discrete Fourier Transform and Real FFT

The Discrete Fourier Transform (DFT) represents the fundamental mathematical tool for transferring a discrete signal from the physical domain to the frequency domain, or, in the case relevant here, from the spatial domain to the wavenumber domain. Given a real sequence u_n, sampled at N_z equally spaced points along the spanwise direction z, its DFT is defined as:

$$\hat{u}_k = \sum_{n=0}^{N_z-1} u_n \, e^{-i\frac{2\pi}{N_z}kn}, \qquad k = 0, 1, \ldots, N_z - 1 \tag{32}$$

where $\hat{u}_k$ represents the complex spectral coefficient associated with the discrete wavenumber k, related to the physical wavenumber by $k_z = 2\pi k/L_z$, with L_z the domain extent along z. Direct computation of the DFT according to its definition requires a computational cost on the order of $\mathcal{O}(N_z^2)$ operations, which rapidly becomes prohibitive as spatial resolution increases. For this reason, computational practice almost universally relies on the Fast Fourier Transform (FFT), an algorithm that exploits the recursive structure of the transform to reduce the computational cost to $\mathcal{O}(N_z \log N_z)$ [37], making the transform efficiently computable even for large datasets, such as those typical of DNS simulations.

In the specific case where the input signal u_n is real, as is the case for the velocity components of a fluid dynamic field, the full DFT is redundant: half of the computed spectral coefficients are in fact deducible from the other half through the Hermitian symmetry property, discussed in the following section [38]. Exploiting this redundancy, an optimised variant of the algorithm can be employed, known as the Real FFT (rFFT), which computes exclusively the unique portion of the spectrum, corresponding to the non-negative wavenumbers $k = 0, 1, \ldots, \lfloor N_z/2 \rfloor$. For a sequence of length N_z, the resulting tensor dimension in the spectral domain is therefore:

$$N_{z,\mathrm{rfft}} = \left\lfloor \frac{N_z}{2} \right\rfloor + 1 \tag{33}$$

The use of the rFFT provides a twofold advantage within the algorithm developed in this thesis: on the one hand, it halves the volume of data to be processed and stored with respect to a full complex transform, significantly reducing memory occupation in a context, such as high-resolution DNS simulations, where this aspect is critical; on

the other hand, it avoids the computational redundancy associated with the explicit calculation of spectral coefficients that would otherwise be deducible by symmetry, resulting in a reduction of computation time.

3.2.2 Hermitian Symmetry and Parseval's Theorem

As anticipated in the previous section, the Fourier transform of a real sequence possesses a fundamental symmetry property, known as Hermitian symmetry, which constitutes the mathematical foundation underlying the use of the Real FFT in the algorithm developed in this thesis.

Hermitian Symmetry
Given a real sequence u_n of length N_z, the complex spectral coefficients $\hat{u}_k$ obtained from its DFT satisfy the relation [38]:

$$\hat{u}_k = \hat{u}^*_{N_z-k} \tag{34}$$

where the asterisk denotes the complex conjugate. This relation implies that the coefficient associated with wavenumber k is the complex conjugate of the coefficient associated with wavenumber $N_z - k$, that is, the corresponding negative wavenumber $-k$ in the centred representation of the spectrum. Several notable consequences follow from this property:

- The zero-wavenumber coefficient, $\hat{u}_0$, corresponding to the spatial mean of the signal, is always a real number.
- If the spatial resolution N_z is even, the coefficient at the Nyquist frequency, $\hat{u}_{N_z/2}$, is also real.
- All remaining coefficients, for $k > N_z/2$, are complex and are uniquely determined, by mirror symmetry, from the coefficients corresponding to $k < N_z/2$.

Fig. 3.2 illustrates this symmetry property: coefficients associated with negative wavenumbers (dashed, grey) are entirely determined by their positive counterparts (solid, blue), while the DC and Nyquist components (red) constitute the only unique, non-redundant elements of the spectrum.

It is precisely this structural redundancy that justifies the use of the rFFT discussed in the previous section: explicitly computing the coefficients for $k > N_z/2$ would be a superfluous operation, since they carry no additional information beyond the "unique" half of the spectrum.

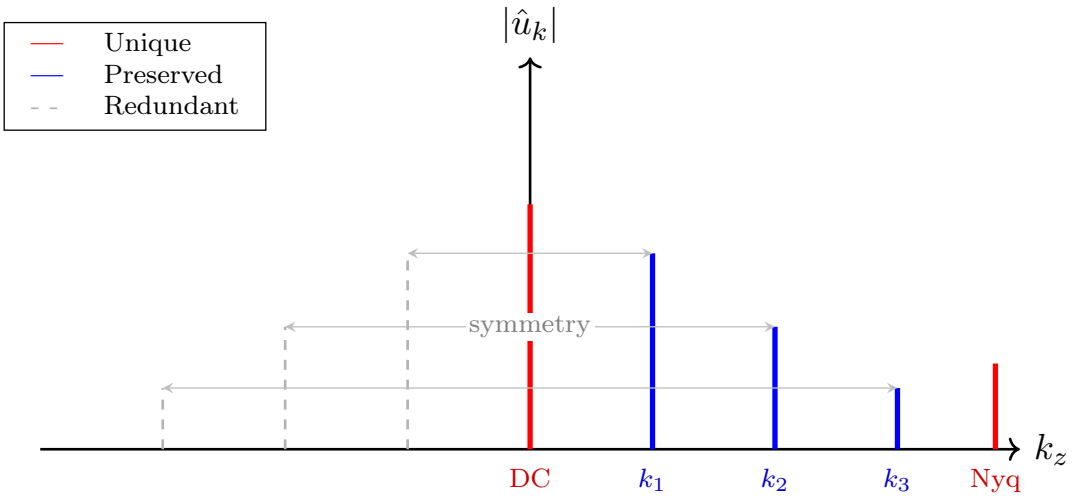


Figure 3.2: Hermitian symmetry of the spectrum.

Parseval's Theorem

The transition from the domain of complex DFT coefficients to that of physically measurable quantities, such as the amplitude and energy associated with each spectral mode, requires a normalisation procedure, whose theoretical foundation is provided by Parseval's Theorem [38]. Given a real sequence u_n of length N_z, the theorem establishes that the total energy of the signal must remain invariant under the transition between the physical and the spectral domain:

$$\sum_{n=0}^{N_z-1} |u_n|^2 = \frac{1}{N_z} \sum_{k=0}^{N_z-1} |\hat{u}_k|^2 \tag{35}$$

This identity guarantees that no energetic information is lost or artificially introduced in the transition to the spectral domain, and constitutes the foundation for the correct normalisation of modal amplitudes. However, since the rFFT computes exclusively the unique portion of the spectrum comprised between $0 \leq k \leq N_z/2$, exploiting Hermitian symmetry to omit redundant coefficients, particular care must be taken in reconstructing the physical amplitude of each oscillatory mode: in order not to underestimate its energy content, the contribution of complex coefficients possessing a conjugate counterpart in the negative frequency half-space, discarded for computational efficiency by the algorithm, must be doubled, while the original normalisation is retained for the zero-mean component and, when present, for the Nyquist-frequency component, which lack a distinct symmetric counterpart. This normalisation procedure, applied in the algorithm developed in this thesis during the computation of spectral amplitudes, is described in detail in Chapter 4.

4 Algorithm Architecture and Implementation

4.1 High-Performance Computing and Distributed Memory Parallelism

The analysis of high-resolution DNS simulation datasets, such as those employed in this thesis, involves data volumes that systematically exceed the RAM capacity of a single compute node, making it indispensable to resort to distributed parallel computing strategies spanning multiple nodes of an HPC (High-Performance Computing) cluster. Unlike shared-memory parallelism, in which multiple processors access a single common address space, distributed-memory parallelism requires each process, or *rank*, to manage an autonomous portion of memory and to explicitly communicate with the other processes through message passing, typically implemented through the MPI (Message Passing Interface) standard [12], employed in the algorithm developed in this thesis.

In this section, the fundamental concepts of the distributed-memory model and of MPI collective communications are recalled, with particular reference to the All-to-All operation extensively employed in 4.4 for data redistribution [13] (§4.1.1), as well as the theoretical principles governing the scalability of a parallel algorithm, summarised by Amdahl's Law and the distinction between strong and weak scaling, which provide the interpretive framework necessary for the discussion of the scalability results presented in Chapter 5 (§4.1.2).

4.1.1 Distributed Memory Model and MPI Collective Communications

In the distributed-memory parallel computing model, each process, or *rank*, has a private memory space, not directly accessible by the other processes. Unlike the shared-memory model, in which threads can directly read and write to a common memory region, in the distributed-memory model every exchange of information between processes must occur explicitly through message passing, according to the paradigm known as *message passing* [13]. This architectural feature, although it imposes an additional burden on the programmer in terms of explicit communication management, allows the model to scale effectively to an arbitrarily large number of compute nodes, since it does not require shared and synchronised access to a single memory area, unlike shared-memory systems, whose parallelism is intrinsically limited by the RAM capacity available on a single node.

The MPI (Message Passing Interface) standard represents the most widely adopted

implementation of the message-passing paradigm in high-performance scientific computing [12], and underlies the entire parallel architecture of the algorithm developed in this thesis. MPI provides two main categories of communication operations: point-to-point communications, in which a single sending process transmits data to a single receiving process, and collective communications, in which all processes belonging to the same communication group (*communicator*) simultaneously participate in the data exchange according to a predefined scheme [13].

Among collective operations, the All-to-All operation plays a central role in the algorithm developed in this thesis, since it constitutes the mechanism through which the transition from the temporal decomposition to the spatial decomposition of the domain is realised, as described in detail in section 4.4. Unlike simpler collective operations, such as *broadcast*, in which a single process distributes the same data to all others, or *gather*, in which the data from all processes are collected onto a single destination process, the All-to-All operation requires each process to send a distinct portion of its own data to every other process, while simultaneously receiving a distinct portion of data from each of them. Conceptually, this operation can be interpreted as a distributed transposition of a data matrix, in which the dimension along which the data are partitioned is exchanged with another dimension of the dataset, while keeping the information distributed among the various processes rather than centralising it on a single node.

In its most general form, employed in the algorithm developed in this thesis, the operation is referred to as All-to-Allv (where the suffix *v*, for *variable*, indicates its generalisation), and allows each process to send and receive different amounts of data to and from each other process of the communication group, rather than imposing a rigidly uniform subdivision. This flexibility is enabled by the specification of four control vectors per process: the *sendcounts* and *recvcounts* vectors, indicating respectively how many elements are sent to, and received from, each other process, and the *sdispls* and *rdispls* displacement vectors, indicating the position, within the respective send and receive buffers, at which the data destined for, or originating from, a given process must be placed. The explicit management of these vectors, although entailing greater implementation complexity compared to a uniform All-to-All operation, is indispensable in the context of this thesis, since the subdivision of the spatial domain among processes is not, in general, perfectly homogeneous, as discussed in section 4.4.1 in relation to computational load balancing.

A further relevant aspect of MPI collective communications concerns their synchronising nature: all processes belonging to the communication group must participate in

the operation for it to complete, implicitly introducing a synchronisation point among the processes. While this aspect, on the one hand, guarantees the consistency of the distributed system state upon completion of the communication, it can, on the other hand, constitute a source of inefficiency when the computational load is not balanced among processes, since processes that complete their portion of work more quickly must wait for the completion of the slower ones before being able to proceed.

4.1.2 Scalability Laws: Amdahl's Law and Strong/Weak Scaling

One of the central aspects in evaluating the performance of a parallel algorithm concerns its ability to effectively exploit an increasing number of processors, a property known as scalability. The best-known theoretical formalisation of this concept is Amdahl's Law, formulated by Gene Amdahl in 1967 [39], which establishes an upper bound on the speedup achievable by parallelising an algorithm, as a function of the fraction of code that is intrinsically non-parallelisable.

Amdahl's Law

Consider an algorithm whose total execution time T_1, on a single processor, can be decomposed into a parallelisable component, of fraction p, and an intrinsically serial component, of fraction $(1 - p)$, which must necessarily be executed by a single processor regardless of the degree of parallelism available. Running the algorithm on N processors, and assuming the parallelisable component is distributed perfectly evenly among them, the resulting execution time T_N is given by:

$$T_N = T_1 \left[(1 - p) + \frac{p}{N} \right] \tag{36}$$

The speedup S_N, defined as the ratio between the serial and parallel execution times, is therefore:

$$S_N = \frac{T_1}{T_N} = \frac{1}{(1 - p) + \frac{p}{N}} \tag{37}$$

The most significant consequence of this relation emerges in the limit of an infinite number of processors:

$$\lim_{N \to \infty} S_N = \frac{1}{1 - p} \tag{38}$$

This limit highlights how the maximum achievable speedup is intrinsically bounded

by the serial fraction $(1 - p)$ of the algorithm, regardless of how large the number of employed processors is: even an apparently negligible serial fraction can constitute a dominant bottleneck once the number of processors becomes sufficiently large. In the context of a distributed algorithm based on MPI communications, such as the one developed in this thesis, the serial component of Amdahl's Law can be interpreted, in a broad sense, as encompassing not only genuinely non-parallelisable portions of code, but also the time spent in synchronisation and collective communication operations, whose relative overhead tends to increase with the number of processors involved, as discussed in Chapter 5 in relation to the transition from a compute-bound to a communication-bound regime.

Fixed-Size and Scaled-Size Performance Analysis

In the practice of parallel algorithm performance analysis, it is customary to distinguish between two different modalities of scalability assessment, depending on whether the problem size is kept constant or is increased proportionally to the number of employed processors.

In the first case, the overall problem size remains fixed, while the number of processors employed to solve it is progressively increased. The goal of this analysis is to evaluate how effectively a fixed-size problem can be solved in a progressively shorter time as the degree of parallelism increases. The scenario described by Amdahl's Law corresponds precisely to this configuration: as the number of processors grows, the portion of work assigned to each of them decreases, while the serial component, and the communication overhead assimilable to it, remains substantially constant or even increases, inevitably leading to a progressive departure from the ideal speedup $S_N = N$, up to a possible saturation point beyond which adding further processors no longer yields any benefit, or even worsens overall performance.

In the second case, conversely, the problem size assigned to each processor is kept constant, while the overall problem size is increased proportionally to the number of employed processors. This analysis modality, formalised by the Gustafson-Barsis Law as a complement to Amdahl's Law [40], is particularly relevant in contexts where the primary goal is not the reduction of computation time for a fixed-size problem, but rather the ability to tackle progressively larger problems while maintaining an approximately constant execution time.

In the specific context of this thesis, the scalability analysis presented in Chapter 5 is conducted according to the first of these two logics: for a fixed size of the analysed DNS dataset, the number of MPI processes is varied, in order to identify the config-

uration that minimises the overall execution time, as well as the point beyond which the communication overhead, consistently with what is predicted by Amdahl's Law, begins to offset the benefits of increased parallelism.

4.2 Architecture of the Computational Pipeline and Parallelism Strategy

The design of the algorithm presented in this thesis stems from the need to process datasets whose size exceeds the physical memory (RAM) capacity of a single compute node.
In the specific case, the Direct Numerical Simulation (DNS) data analysed, on which the algorithm was structured and optimised, feature a spatial grid characterised by 730 points along the streamwise direction (x-axis), 90 points along the wall-normal direction (y-axis), and 303 points along the spanwise direction (z-axis).
Considering single-precision data (4 bytes per element) and the three velocity components, a single snapshot would occupy approximately 0.8 GB of memory; it is therefore evident how the total memory volume occupied by the entire dataset can reach very large values, making a classical serial approach impractical.

To address this issue, a pipeline has been developed that employs a distributed parallelism approach based on MPI (Message Passing Interface). In particular, the structure has been conceived to optimise memory allocation while simultaneously maximising the efficiency of inter-node communications and ensuring the numerical stability required by the Dynamic Mode Decomposition (DMD).

The overall architecture of the algorithm is schematised in Fig. 4.1. The diagram illustrates its constituent phases, providing an overview of the logical flow before proceeding with the detailed analysis of each individual step.

4.2.1 Hybrid Decomposition

The architecture of the implemented algorithm does not adopt a static decomposition; instead, two different distribution mechanisms are employed so that the various phases of the algorithm are individually optimised.
The first mechanism provides a temporal distribution of the dataset across the different processors, while the second provides a spatial distribution.

- Snapshot Configuration (Temporal Distribution): In this initial phase of the algorithm, the primary objective is to minimise Input/Output operation times while maximising their efficiency. The temporal decomposition allows each processor

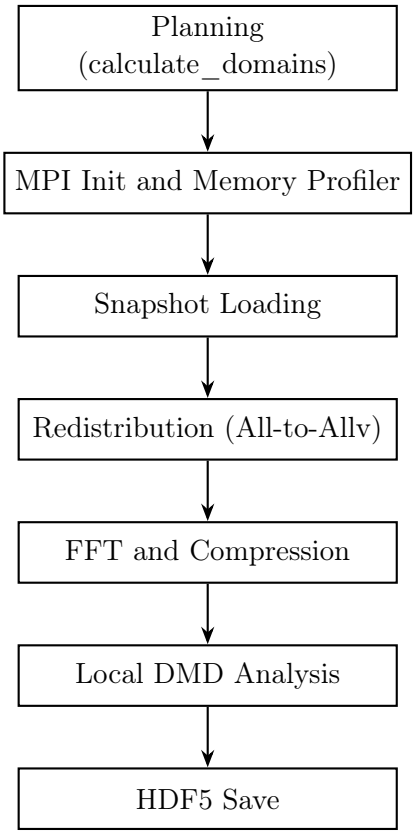


Figure 4.1: Procedural diagram of the algorithm: from the predictive planning phase (calculate_domains) through MPI initialisation, temporal snapshot loading, spatial redistribution via All-to-Allv, spectral compression, local DMD analysis, and distributed HDF5 storage.

(rank) to load the entire spatial domain (x,y,z), but only for a discrete subset of temporal snapshots (Δt). In this way, data reading is no longer a sequential process but becomes a parallel operation; this approach avoids bottlenecks related to the large dimensions of the managed datasets, drastically reducing waiting times and making the process easily scalable.

This configuration is also advantageous in the event that a subset of the spatial domain is to be selected, since all ranks hold information on the complete domain. It also facilitates the subsequent phase of the algorithm, which involves performing the Fast Fourier Transform (FFT) on the data along the z-direction. An important computational advantage is observed: the fact that each rank possesses the entire spatial domain facilitates the FFT operation, which can be performed in-place without the need for inter-node data exchanges to compute the Fourier coefficients, since the entire data vector along z resides in the local memory of the rank.

- Subdomain Configuration (Spatial Distribution): DMD requires, by definition,

knowledge of the complete temporal evolution of every spatial point. A parallel transposition of the data matrix has therefore been implemented. In particular, a global All-to-All communication has been adopted, through which the data are redistributed such that each rank receives the entire temporal history, but relative to a limited spatial domain. Specifically, the redistribution is carried out by partitioning the domain along the x-coordinate (parallel to the flow direction). This choice not only preserves the integrity of the z-vector, thereby simplifying the FFT operation as previously noted, but also maintains the integrity of the velocity profiles along the wall-normal direction (y-axis), facilitating the DMD analysis in capturing vertical gradients and vortical structures that generally develop in the xy-plane.

Fig. 4.2 illustrates the data distribution in the two configurations described above (temporal decomposition and spatial decomposition).

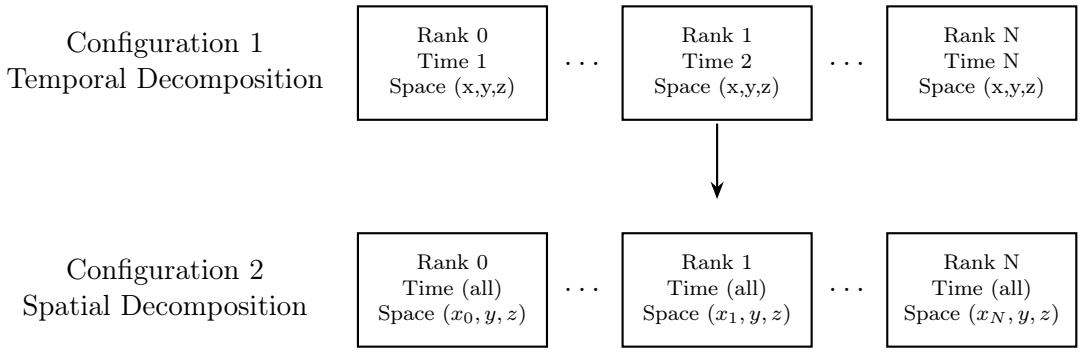


Figure 4.2: Representation of data redistribution among MPI ranks: transition from the temporal snapshot partition to the spatial subdomain partition.

4.2.2 Communication Optimisation and Data Reduction

The operational flow of the framework has been developed to reduce the computational load without losing physically significant information. This memory-preservation approach is applied throughout the algorithm through several solutions.

For instance, within the All-to-All communication a block-based communication system has been introduced, which segments the global message into smaller packets, ensuring that the global memory peak during the exchange remains within limits, preventing communication crashes.

Furthermore, following the FFT, a spatial compression has been implemented prior to redistribution. Performing the Fourier Transform along z allows working in the wavenumber domain k_z and, by exploiting the symmetry of real data, the data volume is reduced by 50%.

4.2.3 Memory Monitoring and Profiling

A Memory Profiling system has also been integrated into the algorithm; this allows real-time monitoring of RAM allocation during each phase of the pipeline (loading, All-to-All, FFT, DMD).
In detail, the memory profiler fulfils three main functions:

- Predictive Validation: it allows validation of the *calculate_rank_domains* function integrated in the algorithm. This function estimates the memory requirement for each rank based on the domain parameters (x,y,z,t) and the number of requested cores; the RAM estimate is of fundamental importance, as it forms the basis for drafting the SLURM scripts, which cannot be submitted without knowledge of these parameters. Prior knowledge of the parameters to be passed to SLURM avoids trial-and-error processes aimed at determining the minimum necessary memory allocation, preventing both job failures due to Out Of Memory (OOM) and excessive memory allocation and thus wastage of computational resources.
- Load Balancing: the real-time profiler allows verification that the workload distribution is homogeneous across the different ranks, avoiding overloads and potential pipeline slowdowns caused by communication phases.
- Resource Management: it monitors the effectiveness of Garbage Collection operations, appropriately inserted in the code, ensuring that data no longer needed are promptly removed from memory.

4.3 Input/Output Management and Temporal Partitioning

The loading and management of data represent one of the most complex phases of the pipeline. In this phase it is of fundamental importance to guarantee the integrity of the distributed data loading while simultaneously minimising the memory footprint of each MPI rank.

Unlike a global domain decomposition, the implemented approach is based on a sliding window along the x-axis. This methodological choice allows isolation and examination of the evolution of modes along the direction of transition development, requiring, however, an extremely rigorous I/O management. To handle these aspects, the implementation adopts a predictive planning strategy and a balanced data partitioning. In particular, the algorithm performs a preliminary analysis of the dataset topology with the aim of identifying the logical position of each time instant and each window within

the file structure. This allows each processor to know the portions of data within its competence, access the files in a targeted manner independently of other ranks, and allocate exclusively the resources necessary for its own fraction of computation, avoiding redundancies.
This preliminary organisation establishes the basis for the subsequent temporal data distribution.

4.3.1 Predictive Resource Analysis: calculate_rank_domains

The *calculate_rank_domains* function defines the logic of the entire system, enabling data subdivision and computing the spatial autonomy of each process.

In a first phase, the code determines the windows and computes the number of distinct positions each can assume within the considered domain. In particular, the parameter N_positions is defined as follows:

$$N\_positions = \frac{(Nx\_total - W\_size - 1)}{step + 1} \tag{39}$$

Note that the division is rounded down to the nearest integer. In Eq. 39, Nx_total represents the extent of the chosen domain to be subdivided among the different ranks, W_size is the set width of the sliding window, and step is the fixed advancement step of the window. It is important to highlight how this definition accounts for the fact that, for the correct application of DMD, each window must include the window itself shifted along x by one step in order to define the matrix X'; furthermore, the calculation of N_positions is such that the last window does not exceed the complete dataset, considering also the shift of one position for the matrix X'.

The second phase consists of managing active and inactive ranks. In the code, the parameter active_size evaluates the smaller value between the size, i.e. the number of processors requested by the user, and N_positions. In particular, if the number of requested processors exceeds the available windows, the code identifies the excess ranks and returns null values, ensuring that only the necessary cores are activated for loading.

The third phase handles the balanced distribution of the workload. In this phase, the code identifies the minimum number of windows to be assigned to each rank through the definition of the base parameter ($base = N\_positions//active\_size$), computes the number of remaining windows (remainder) and distributes them individually among the first ranks.
In detail, if the current rank index is less than the remainder, it will be assigned a

number of windows equal to base + 1; otherwise, it will be assigned only the base quota. This guarantees that the load difference between processors is at most a single window, thereby minimising waiting times during synchronisation.

The fourth phase concerns the computation of local pointers and the management of overlapping between the different domains assigned to the different processors. The function translates the number of windows into real grid indices, defining the start and end boundaries of the domain to be allocated to the processor; it is recalled that the terminal point of the domain must be such as to fully encompass the last positioned window, accounting for the shift of one position as previously explained.
The very fact that the domain must be cut so as to contain the entire last window plus one position means that the domains of adjacent ranks overlap. This overlap is essential as it guarantees that all windows at all expected positions are considered and that each rank holds all the spatial points characterising a given window, allowing individual ranks to process their own windows in complete autonomy, eliminating the need to exchange spatial data during computation.

As described in Sec. 4.3.1, once $N\_positions$ is computed via Eq. 39, the $calculate\_rank\_domains$ function distributes these positions among the active MPI ranks in a balanced manner, assigning each rank either $base$ or $base + 1$ windows. The resulting subdomains are not simply juxtaposed but partially overlap at their boundaries (*ghost overlap*): this redundancy is necessary because the last window of a rank also requires the points corresponding to the one-position shift used to build the X' matrix in DMD, thereby avoiding spatial data exchange between adjacent ranks.

Fig. 4.3 shows this partitioning for a simplified three-rank case, highlighting the overlap regions shared at the boundaries between contiguous domains.

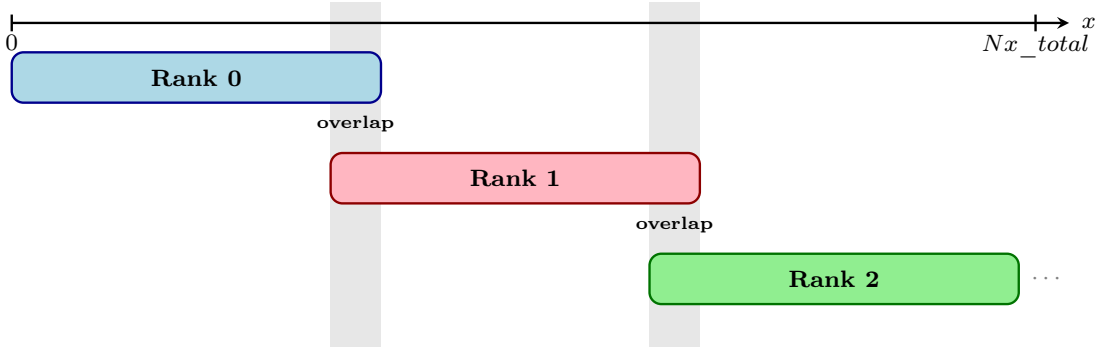


Figure 4.3: Balanced distribution of the sliding-window domain among MPI ranks.

4.3.2 Memory Peak Estimation Model and Diagnostics

The *calculate_rank_domains* function is not limited to geometry management and correct subdomain distribution to the different ranks, but also includes a node load simulator, which allows prior assessment of the memory to be allocated.

A first function interrogates the loaded file to extract the effective length of the time series; the contributions of different files are considered and summed so as to obtain the total number of time steps dynamically, decoupling the code from prior knowledge of the temporal dimension of the dataset.
Subsequently, a peak RAM estimate is computed for all the different phases of the algorithm (loading, All-to-All, FFT, DMD). The calculation is performed on the most loaded rank, as the most severe condition, accounting for the three velocity components u, v, w and the single-precision format of the data.
The evaluated quantities are summarised in a detailed final report, useful for job submission on the cluster. In particular, the overall output of the file includes for each rank: the domain boundaries in x, the number of windows contained therein, and the number of points within the domain. In addition, the maximum RAM peak reached by the heaviest process is reported, along with the maximum number of processors allocatable per node considering the total node memory and the maximum memory occupied by the most loaded rank. Finally, the minimum number of required nodes is computed.
These pieces of information constitute the directives for job submission, optimising resource allocation and preventing OOM failures.

4.3.3 Parallel I/O Architecture and Temporal Partitioning Logic

The management of the temporal dimension in the utils.py module does not consist of a simple reading procedure, but exploits a system that treats a sequence of physical files as a single global tensor. This approach is fundamental to guarantee the scalability of the code. The main functions managing this process are analysed below.

Topology Inspection and Automatic Parameter Detection
Before initiating the loading phase, the system must have a clear picture of the complete dataset structure. This phase is referred to as topology inspection and is fundamental for mapping variables distributed across multiple files and for adapting the algorithm to the DNS data structure.
The *Get_Var_Info_List* function interrogates the HDF5 file header using a read-only approach, scanning all keys and identifying all variables; for each of them, three parameters are then isolated: the tensor geometry (shape), the data type (dtype),

fundamental for predicting the memory footprint, and the total weight in megabytes (MB) (size_MB), obtained from the product of the dimensions and the data type size divided by 1024^2.
This operation returns the necessary information collected in VarInfo, allowing computation of the total number of snapshots by summing the contributions of all files in the list, an operation necessary for the subsequent temporal partitioning.

Temporal Detection Mechanism
In order to perform temporal partitioning, it is first necessary to clarify the arrangement of the different data dimensions, in particular the position of the temporal coordinate. To this end, the function *Time_slicing_idx* is defined.
This function receives as input the dimension of a velocity variable and the length of the temporal vector TimeList, and performs an iterative comparison between the dimensions of the velocity variable tensor and the time vector until a match is found. Once the matching dimension is identified, the coordinate of the tensor for which coincidence with the TimeList dimension occurs is determined, and the parameter time_idx is set. The latter instructs all partitioning functions as to which axis to partition on, eliminating the need for manual configuration and reducing the risk of errors in job submission.

Hybrid Loading Policy
With the metadata acquired, the system does not proceed with blind loading, but applies a procedural strategy to optimise resource usage; the software autonomously selects the most efficient reading mode based on data size.
The comparison between the variable size and a pre-set dimensional threshold is at the core of the decision-making process; this threshold represents the balance point beyond which reading a file serially would constitute a bottleneck, whereas parallel reading would be more advantageous. Two strategies are therefore distinguished:

- Serial Strategy: Variables falling below the set threshold are classified as global, and for these the code adopts a serial technique. Only rank 0 reads the file from disk, and once reading is complete, the data are broadcast to all other processors via an MPI broadcast operation. This prevents all processors from simultaneously attempting to read the same small file, an approach that would reduce the overall efficiency of the algorithm, using parallelisation under conditions where it would not only be unnecessary but detrimental.

- Parallel Strategy: Variables exceeding the threshold are defined as local, and for their reading the code activates the parallel logic. Each rank ignores data out-

side its competence and points directly to its specific temporal partition. Reading therefore occurs in a distributed fashion, with each processor operating simultaneously on a distinct portion of the dataset.

Fig. 4.4 presents a block diagram clarifying the hybrid loading policy.

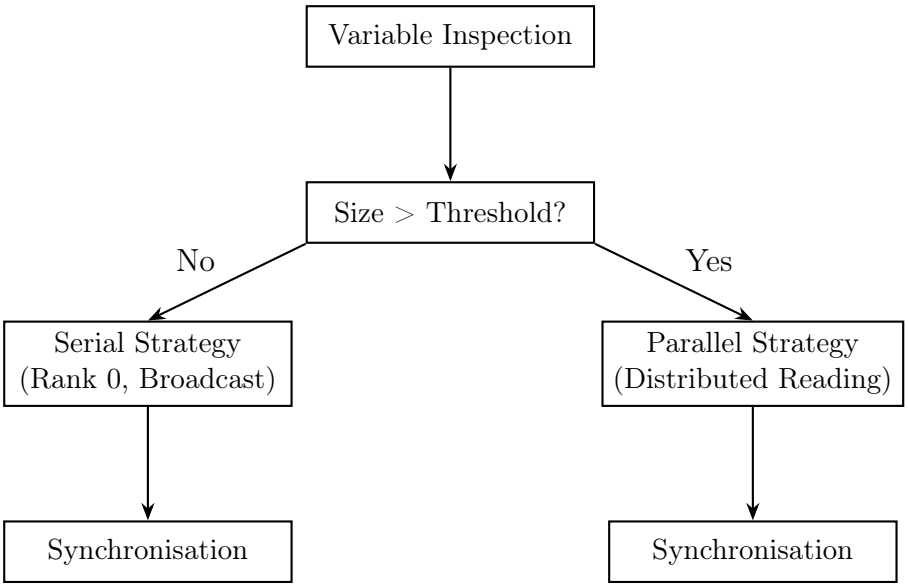


Figure 4.4: Hybrid loading policy: variables below the size threshold are read serially by rank 0 and broadcast (Serial Strategy), while larger variables are read directly and independently by each rank according to its temporal partition (Parallel Strategy).

Reading Optimisation

The *Read_ data_ modified_ V3* function has been implemented to handle the criticalities related to reading very large data without causing OOM system failures; in particular, the function is based on three strategies: contiguous pre-allocation, dynamic slicing, and direct data transfer.

Contiguous pre-allocation resolves the memory fragmentation problem; a simplistic approach would involve sequential reading of the files followed by concatenation, requiring progressively larger memory spaces as the operation proceeds. The data would in fact be copied, temporarily doubling the RAM usage.
The implemented function instead pre-computes the total temporal quota assigned to the rank, The comparison between the variable size and a pre-set dimensional threshold is at the core of the decision-making process; this threshold represents the balance point beyond which reading a file serially would constitute a bottleneck, whereas parallel reading would be more advantageous. In this way, every bit of data to be read

already has its allocated space, eliminating the risk of memory spikes and significantly accelerating data access.

The need to resort to dynamic slicing derives from the possibility of DNS data having different structures, requiring high flexibility from the function. Rather than using large blocks with conditional statements, slicing objects are constructed dynamically: the code defines two pointers for each read operation, the source pointer indicates the exact coordinates within the HDF5 file, while the destination pointer indicates where the selected snapshots are to be placed within the pre-allocated RAM block.

The direct transfer strategy finally avoids the standard Python reading in which data travel from disk to a temporary buffer, then to a Python variable, and finally to the destination.
As a replacement, the *read_ direct* function allows transferring data directly from the filesystem to the pre-allocated memory buffer exploiting a Zero-copy I/O process that reduces CPU overhead. In this way the processor does not need to move data between different memory areas, and the instantaneous RAM consumption is halved, eliminating the need for intermediate variables.
Fig. 4.5 clarifies the difference between standard reading and Zero-copy reading.

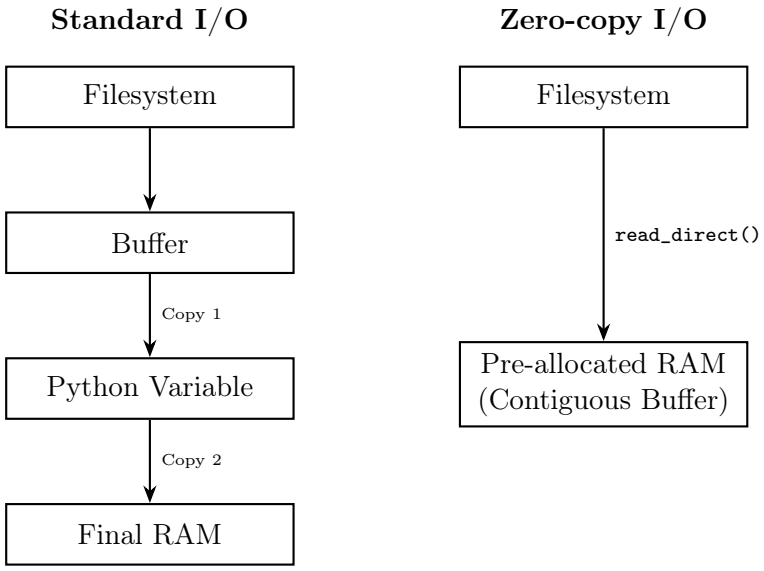


Figure 4.5: Comparison of data flows: on the left, the standard reading method with intermediate steps and copies; on the right, direct transfer via read_direct.

Performance Monitoring, Synchronisation and Validation

Once tensor loading is complete, a verification phase begins to ensure that data fragmentation has not compromised the temporal and spatial coherence of the dataset. In detail, rank 0 aggregates the time lists extracted from the various files in order to re-

construct the complete chronology and verify the absence of temporal gaps or missing snapshots; the spatial coordinate vectors are also reconstructed on rank 0 so that a control log can be generated to detect any errors.
The code also imposes an MPI barrier: in a parallel environment, processors do not always finish reading data at the same instant; it is therefore necessary to define a forced stop such that no process can begin the next phase until the last rank has completed its assigned operations.

4.4 Data Reorganisation: the MPI All-to-All Mechanism

As anticipated in the description of the hybrid decomposition, the transition between the temporal subdomain configuration and the spatial one represents the most critical phase of the pipeline from the standpoint of inter-processor and inter-node communication. In this phase, the distribution must be reconfigured from temporal to spatial: initially each rank holds the entire spatial domain for a temporal sub-interval, while at the end of the operation each rank must hold the entire temporal extent for a limited portion of the spatial domain (subdomain). This reconfiguration is an indispensable prerequisite for the execution of DMD.

4.4.1 Definition of Spatial Domains and Load Balancing

As already highlighted, the domain partition cannot follow a standard linear subdivision due to the specific requirements of DMD, which imposes the adoption of a sliding window. The calculate_rank_domains function handles this criticality, assigning to each active rank an integer number of window positions computed via Eq. 39.

A fundamental aspect of the code lies in the management of overlapping. As already noted, each window of size W_{size} must be translatable by one spatial point along the x-direction so that it is possible to construct both the matrix X, based on the physical window, and the matrix X' based on the shifted window; this implies that every domain assigned to the ranks must include ghost points, the point identified by x_{end} of each rank must be sufficiently extended to cover the full width of the last positioned window plus the shift required for the construction of matrix X'. Consequently, successive ranks share portions of a common domain; this overlap guarantees that each rank has all the spatial information necessary to process its own windows in complete autonomy, eliminating the need for spatial communications during the computation phase.

In order to clarify the spatial decomposition of the domain and the construction of the snapshot matrices, the sampling strategy is schematised in Fig. 4.6.

As shown in the diagram, the total longitudinal domain is analysed discretely through the sequential advancement of the sliding window of fixed width, whose translation is regulated by the STEP_SIZE parameter. For each position, the current window X_i is chromatically paired with the respective shifted matrix Y_i, translated forward by a single grid unit as previously explained. The overlap between consecutive positions guarantees the physical continuity of the extracted modes.
It is further noted that the advancement of the window is constrained by the domain geometry: neither the original window nor the shifted one can exceed the last available point. Consequently, the grid nodes following the width of the last step (highlighted in grey on the domain axis) do not have the spatial support necessary to generate a window pair and are therefore structurally excluded from the DMD.

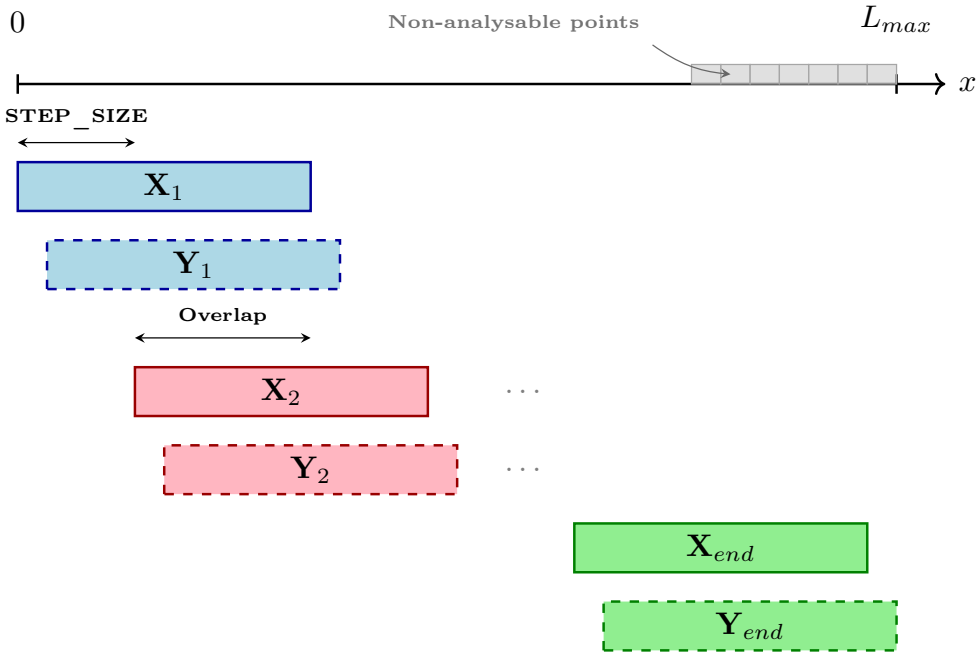


Figure 4.6: Diagram of domain subdivision into overlapping windows.

4.4.2 Concept of Parallel Transposition: the All-to-All Mechanism

To enable redistribution, the All-to-All logic is employed. This operation can be likened to a logistical transposition of a tensor: initially the data reside in an initial configuration characterised by a snapshot-based distribution, and for the purposes of DMD it is necessary to transition to a second configuration with spatial distribution.

To clarify the exchange logic, reference is made to a hypothetical dataset characterised by 100 total snapshots, 40 grid points $(x_0, x_1, ..., x_{39})$ and 4 MPI ranks among which to distribute the data. The initial phase is schematised below:

- rank 0: holds times 1–25 for all points $x_0...x_{39}$.
- rank 1: holds times 26–50 for all points $x_0...x_{39}$.
- rank 2: holds times 51–75 for all points $x_0...x_{39}$.
- rank 3: holds times 76–100 for all points $x_0...x_{39}$.

During the All-to-All operation, each rank consults the planning defined by *calculate_rank_domains* and subdivides its temporal domain into 4 spatial blocks. Assuming for simplicity a balanced subdivision of grid points, each rank at the end of the communication process should hold 10 grid points for 100 time instants. Taking as reference rank 0 in its initial configuration (complete spatial domain, temporal subdomain), the subdivision of the overall domain into the 4 subdomains to be appropriately transferred is reported below:

- Points $x_0...x_9$ (to be kept on rank 0)
- Points $x_{10}...x_{19}$ (to be sent to rank 1)
- Points $x_{20}...x_{29}$ (to be sent to rank 2)
- Points $x_{30}...x_{39}$ (to be sent to rank 3)

All other ranks simultaneously perform the same operation on their respective temporal intervals. During collective communication, data are redistributed into receive buffers that enable the redistribution.
With reference to rank 2, the following configuration is obtained:

- from rank 0 it receives domain $x_{20}...x_{29}$ relative to times 1–25
- from rank 1 it receives domain $x_{20}...x_{29}$ relative to times 26–50
- from rank 2 it receives domain $x_{20}...x_{29}$ relative to times 51–75
- from rank 3 it receives domain $x_{20}...x_{29}$ relative to times 76–100

Rank 2 in the final phase therefore holds a reduced spatial domain but the complete temporal history. This enables the correct execution of DMD since every temporal dynamic passing through the spatial points assigned to the rank is contained within the rank itself.

Fig. 4.7 reports a flow diagram clarifying the movement of the different blocks from the initial configuration towards the final configuration.

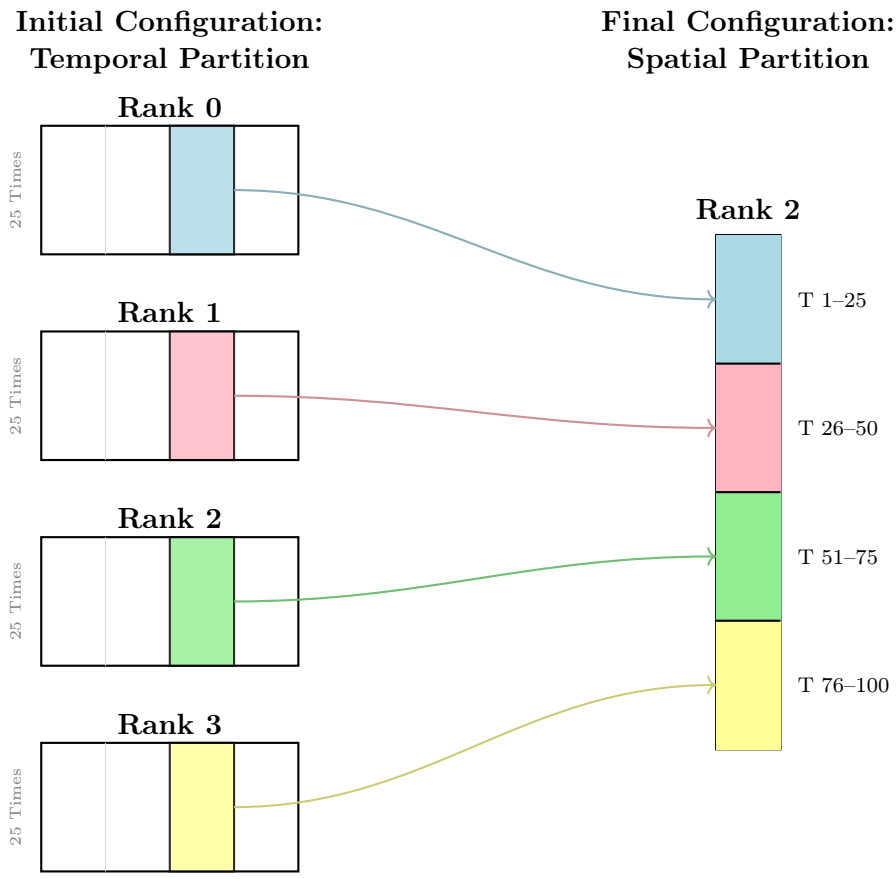


Figure 4.7: All-to-All diagram: transition from the initial configuration, in which each rank holds a temporal subset of the complete spatial domain, to the final configuration, in which each rank holds the complete temporal history of a restricted spatial subdomain.

4.4.3 Structure of the execute_chunked_transfer Function: Metadata and Synchronisation

The *execute_chunked_transfer* function coordinates the complex system of indices and pointers that enable the redistribution.

Pre-allocation

Before initiating the exchange, each rank computes the total size of the destination tensor U_full_time. This is determined by the sum of all Nt_local values (obtained via allgather), ensuring that the buffer can contain the entire dataset chronology, and by the spatial dimension limited to the specific subdomain of the rank (my_nx). This pre-allocation is crucial as it allows data to be written directly into their final position as they are exchanged, avoiding continuous data copies that would consume RAM and cause OOM errors.

Counts Management

In an All-to-Allv operation, unlike standard communication where all ranks exchange the same amounts of data, each rank can send and receive packets of different sizes. The arrays sendcounts and recvcounts are defined precisely to provide each processor with information on how much space must be reserved for each rank with which communication takes place.

Sendcounts in particular manages the planning of data sending. Each sender rank must declare how many elements it is about to send to each of the other ranks present in the cluster with which it is communicating.
The number of elements (grid points) is obtained from the product of the number of times (Nt_{chunk}) and the spatial volume destined for a given rank ($N_z, N_{x_{dest}}, N_y$).
Furthermore, if a given rank does not hold spatial data falling within the domain of the receiving rank, the corresponding sendcounts value is set to zero, avoiding the sending of unnecessary messages and optimising communication.
Finally, although the code computes the number of elements, for the MPI function these are converted into their effective weight in bytes (multiplied by 4 bytes, as the data are in single precision).

Fig. 4.8 illustrates how the local buffer of a sending rank is partitioned according to the sendcounts array: each contiguous slice corresponds to the portion of data destined for a specific recipient rank, including the self-directed slice that Rank 0 retains locally.

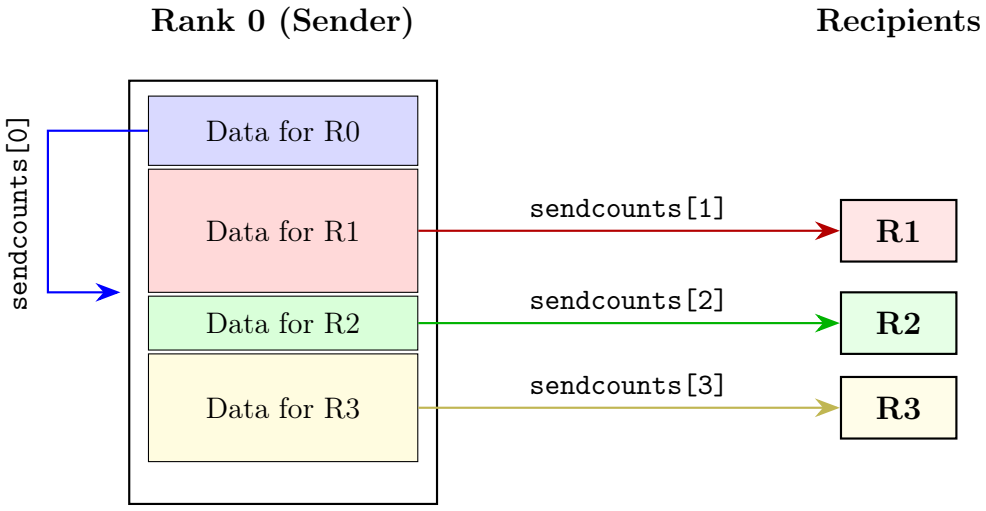


Figure 4.8: Composition of the sendcounts array in the All-to-Allv communication.

The recvcounts array, on the other hand, handles preparation for reception, symmetrically to what is done by sendcounts. Each rank must in fact know how many data it will receive from the other ranks with which it communicates, so as to accommodate

them correctly.
An initial synchronisation with comm.allgather(current_N_chunk) is provided to manage any differences in the number of snapshots processed by the different ranks and therefore the potential differences in process completion times.
Subsequently, the number of arriving elements is computed by multiplying the number of times Nt_{chunk} to be received from all ranks by the local spatial dimension (N_z, my_{nx}, N_y).

This management guarantees the robustness of the algorithm, as the pipeline is thereby rendered independent of differences in rank load and differences in domain distributions.

Displacement Logic
A further criticality in the All-to-All logic lies in the ability to map a complex multidimensional (4D) data structure into a linear (1D) memory represented by the buffer. Since MPI libraries operate on contiguous memory buffers, the code must instruct the system on how to interpret the information contained in the buffer and how to write to it; this is made possible through the use of Displacements (sdispls and rdispls).

In Python, the dataset is perceived as a logical tensor, but in RAM it is stored as an uninterrupted sequence of bits. When a rank extracts a portion of domain to send to another rank, it performs a linearisation (flattening) operation such that the 4D data cube is rewritten as a one-dimensional sequence of values. In this way, the dimensional hierarchy is lost, and each grid point is identified solely by its relative position with respect to the first value contained in the one-dimensional array.

The send Displacement (sdispls) manages the sending address; during the preparation phase, each rank constructs a single send buffer containing the packets for all other processors, and the sdispls vector acts as a position pointer within the buffer itself.
In particular, it indicates to MPI the exact point (offset) within the send buffer from which to begin extracting data for a given recipient; this is computed as the cumulative sum of the preceding sendcounts, if, for example, 100 elements have been reserved for rank 0, the sdispls for rank 1 will start at position 100.

The receive Displacement works analogously, managing the reconstruction of order. Symmetrically to sdispls, when data arrive at the destination rank, they are inserted into the receive buffer recv_buffer which combines the contributions of all senders.
The rdispls defines where to place the incoming packet from each rank to avoid overlaps and to guarantee temporal continuity; in this way the receiving rank is able to allocate the received data in the correct order, reconstructing the chronological sequence required for DMD.

It is noted that, also in this case, both for sdispls and rdispls it is necessary to translate the indices into a characteristic weight in bytes, multiplying the displacement by the data type size.

Fig. 4.9 illustrates how the linearised 1D send buffer is partitioned into contiguous segments destined for different recipient ranks (R0, R1, R2). The displacement values displs[0], displs[1], displs[2] represent the byte offset at which each recipient's data block begins within the buffer: displs[0]=0 indicates that the data for rank 0 starts at the buffer origin, while each subsequent value is computed as the cumulative sum of the preceding sendcounts, shifting the starting pointer forward accordingly. This mechanism allows MPI to unambiguously extract and route each data packet to its intended destination.

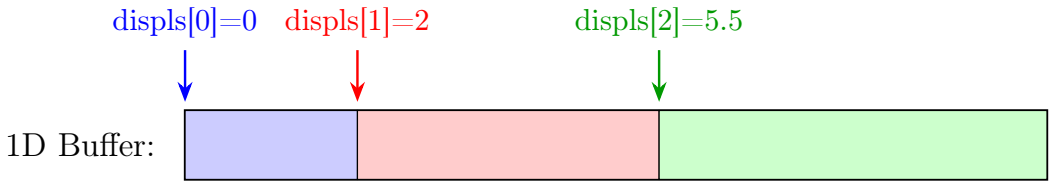


Figure 4.9: Relationship between buffer and displacement.

Unpacking and Reconstruction: from Buffer to 4D Dataset

After execution of the All-to-Allv function, the receiving rank interfaces with the recv_buffer containing the linear data from all other ranks in the cluster; the Unpacking phase allows extraction of the data relevant to the rank, restores their three-dimensional form, and arranges them chronologically.

The code iterates over the rdispls array and for each sender rank the corresponding data segment is extracted and a reshape operation is applied so that the data reassumes the form $(NT_{chunk}, N_z, nx, N_y)$; in this way the data are no longer merely a sequence of numbers but regain their physical meaning, once again becoming a spatial grid evolving over time.

Once communication is complete, the receiving rank must reorder the data arriving in the rcv_buffer, which arrive as temporal blocks from different senders. The code must therefore be able to reconstruct the correct sequence. This is made possible through the write_offsets, which indicate where the time series of each rank begins, and the local_write_counters, which keep track of how much has already been written; thus each packet is placed exactly in its chronological position within the final container U_full_time.

Fig. 4.10 illustrates how the final container U_full_time is progressively populated

during the unpacking phase. Each block (Chunk 1, Chunk 2, Chunk 3...) represents the temporal data received from a distinct sender rank. The write_offset pointers define the absolute position within U_full_time at which each chunk must be written, while the local_write_counter tracks how many elements from that sender have already been inserted, ensuring that successive packets are appended contiguously without overwriting previously received data.

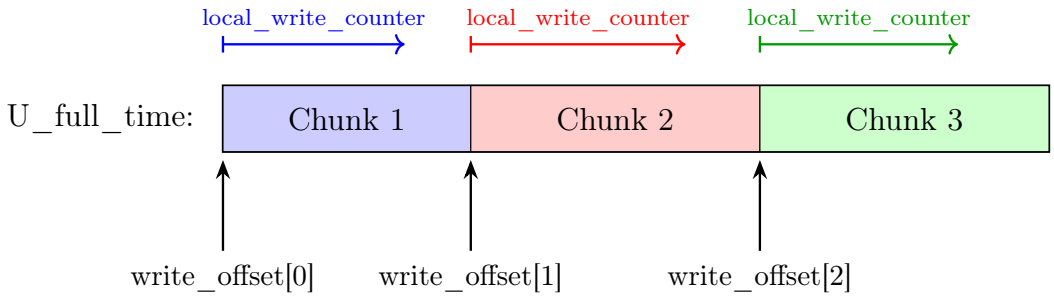


Figure 4.10: Chronological reconstruction logic.

OOM-Related Issues

A further particularly critical aspect in parallel computing is the management of RAM during global exchanges. Although MPI offers a standard All-to-Allv function, its direct use on massive datasets would inevitably lead to Out Of Memory errors. For this reason the algorithm has been structured to implement a fractioned version of the exchange for the following reasons:

- The Communication Buffer Limit: In a standard All-to-All, each rank must prepare a send buffer and a receive buffer containing the entire distributed dataset; in this way, if a rank contains a certain amount of data, the standard function would require twice the memory occupied by the rank, since this would be required simultaneously by both the send and receive buffers. The total memory occupied would therefore be triple, easily exceeding the physical capacity of the node and causing job crashes.
- The Safety Factor: In order to prevent system failure due to OOM, a control parameter named max_mem_per_chunk_GB has been introduced in the code, acting as an upper limit threshold. The software computes how many temporal snapshots can be exchanged without ever exceeding this threshold; the variable Nt_chunk (representing the maximum number of snapshots per single exchange) is determined dynamically based on the dataset size decreasing Nt_chunk increases the number of required exchanges but guarantees that the RAM limit is not exceeded.

- Residual Memory Management and Scalability: the chunked approach allows the system to progressively discard already-exchanged data. Between one exchange cycle and the next, the received data are written into the pre-allocated final container and the temporary buffers are emptied and reused. In this way, sudden allocation spikes are prevented, limiting the causes of failures in the cluster.

Fig. 4.11 presents a flow diagram summarising the main phases of the All-to-All redistribution.

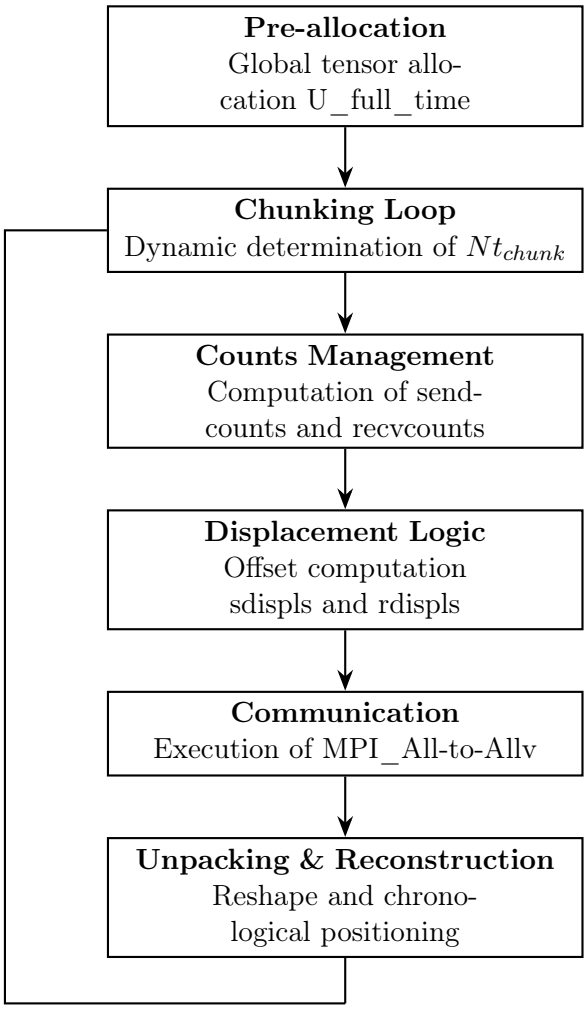


Figure 4.11: Flow diagram of execute_chunked_transfer.

4.5 Fourier Transform and Memory Optimisation

The effectiveness of the spectral decomposition analysed in this section is strictly dependent on the data redistribution operation described in the previous section. The All-to-Allv algorithm has in fact allowed overcoming the constraint of distributed spatial partitioning, so as to reconstruct for each rank the complete temporal history

relative to its own subdomain.

Data realignment is a crucial prerequisite: the FFT algorithm requires that the signal along the transformation direction (z-axis) be contiguous and entirely available in the local memory of the processor. Without the preceding global communication phase, the application of the Fourier transform would be limited to partial segments of the domain, preventing the correct resolution of the lowest wavenumbers and distorting the integrity of the energy spectrum. The tensor U_full_time, obtained from the unpacking process of the receive buffer, therefore constitutes the structured data foundation on which the domain change can be operated.

4.5.1 Spectral Decomposition and Hermitian Symmetry

The spectral transformation phase allows mapping the velocity fields from the three-dimensional physical domain to the wavenumber domain, applying the Real FFT algorithm along the spanwise (z) direction, exploiting the Hermitian symmetry property discussed in §3.2.1–3.2.2. The tensor *U_full_time*, obtained from the redistribution phase described in §4.4.3, is transformed in-place, yielding a spectral tensor whose dimension along z is reduced from N_z to $N_{z,\text{rfft}} = \lfloor N_z/2 \rfloor + 1$ (Eq. 33), halving the memory footprint with respect to a full complex transform.

4.5.2 Memory Allocation Optimisation Strategy

To prevent node failure due to memory spikes, a sequential pipeline with a sliding buffer has been implemented. Rather than transforming the entire velocity tensor **u**=(u,v,w) simultaneously, the software performs an in-place transformation and concatenation operation.

The system performs a preliminary check on the input tensor to identify its precision; if the physical data are in single precision (float32), the software forces allocation into a complex container of dtype complex64. The numpy.fft library would in fact tend to promote results to complex128, unnecessarily doubling memory usage. In this way, 50% of RAM is saved without any loss of physical significance for the data under analysis.

Furthermore, to avoid redundant memory allocation, the destination tensor UVW_hat is pre-allocated with the following dimensional layout:

$$\text{Shape} = (N_t, N_{z,rfft}, n_{x,local}, 3 \cdot N_y) \tag{40}$$

The concatenation along the N_y axis is not arbitrary: it ensures that the spectral components $\hat{u}, \hat{v}, \hat{w}$ for a given spatial coordinate (z,x,y) and temporal instant (t) are

stored in contiguous memory blocks.

The implementation follows an Extract-Transform-Delete scheme, designed to maintain constant memory consumption. The code operates as follows:

- Targeted Allocation: An empty tensor UVW_hat of defined size is created, correctly dimensioning the component payload.
- Sequential Loop: for each component (U,V,W), the function np.fft.rfft is called in isolation and the result is written via slicing into the designated sections of the tensor UVW_hat.
- Critical Memory Point: immediately after writing, the physical tensor is removed from the local reference and a call to the garbage collector is forced.

This sequence guarantees that the memory peak never exceeds the sum of the memory of the final complex tensor and the single physical component being processed.

Fig. 4.12 reports the flow diagram related to the management of the U,V,W matrices.

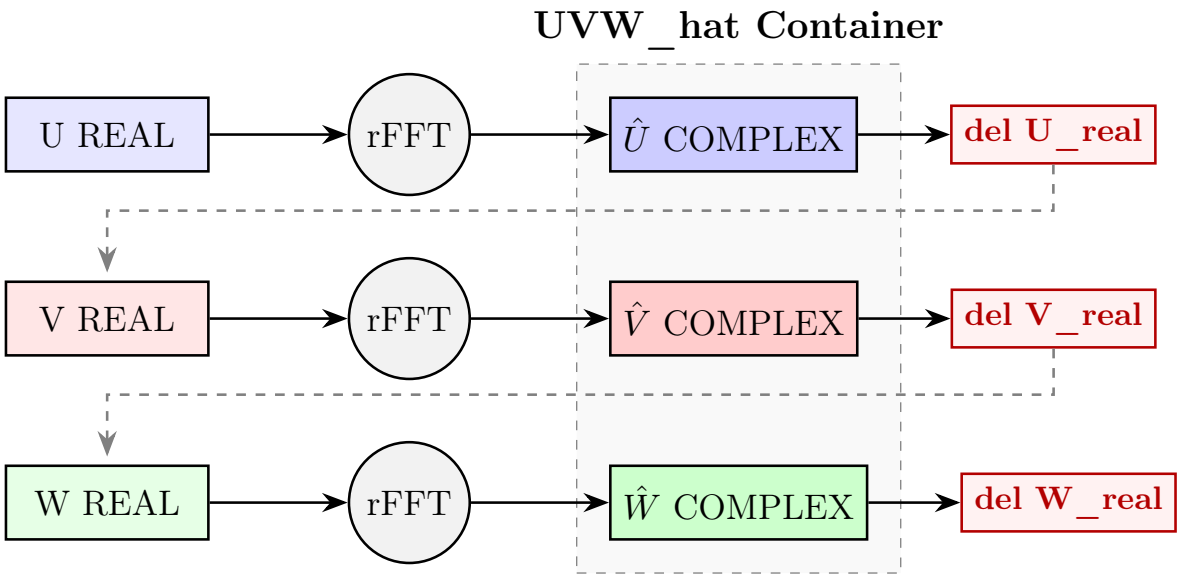


Figure 4.12: Flow diagram of the rFFT pipeline.

4.6 Energy Analysis and Optimised Spectral Mode Selection

The spectral decomposition via rFFT generates a wavenumber domain extending up to the Nyquist frequency. However, the entire spectrum is not useful for characterising turbulent flows via DMD.
In this phase, the software performs a spectral diagnostic to identify the dominant

wavenumbers k_z, enabling a model reduction that preserves the dynamic integrity of the system while simultaneously reducing RAM usage.

4.6.1 Spectral Normalisation Analysis: Theory and Algorithm

The transition from the domain of complex rFFT coefficients to that of physically measurable quantities requires a normalisation procedure, based on Parseval's Theorem discussed in §3.2.2 (Eq. 35). Since the rFFT algorithm used in the implementation computes only the one-sided portion of the spectrum ($0 \leq k \leq N_z/2$), to reconstruct the correct physical amplitude of each oscillatory mode without underestimating the energy content, it is necessary to double the contribution of the complex coefficients that possess a conjugate counterpart in the negative frequency half-space (discarded by the algorithm for efficiency). In order to compute amplitudes and energy contents, the function
compute_max_amplitude_and_energy has been implemented, in which the transformation of coefficients into physical amplitudes $\mathrm{A}(k_z)$ follows a conditional logic to distinguish symmetric spectral pairs from unique modes.
This distinction is managed by the code constructing a loop over the spectral vector file_hat_3D:

- DC Component ($k_z = 0$): Represents the spatial mean of the signal, has no negative counterpart and is therefore normalised simply by dividing by N_z.
- Nyquist Component ($k_z = N_z/2$): If present, is by definition real and does not possess a distinct conjugate mode; it therefore requires individual normalisation.
- Oscillatory Modes: For all other modes, a multiplicative factor of two is applied to recover the energy of the mirror image mode, ensuring that $\mathrm{A}(k_z)$ corresponds to the real amplitude of the sinusoid in the physical domain.

Division by N_z performs a regularisation function, rendering the mode amplitudes independent of the number of points in the physical grid, thereby potentially allowing comparison between DNS simulations performed with different spatial resolutions. This invariance is fundamental for the k_z selection phase: it ensures that spectral truncation occurs on absolute energy grounds and is not dependent on the numerical discretisation.

4.6.2 Spectral Energy Integration: Foundations and Implementation

Given the physical configuration of the problem, the flow field exhibits a marked inhomogeneity along the wall-normal direction (y). Velocity gradients vary significantly,

rendering the energy evaluation at a single spatial point insufficient to characterise the real weight associated with a given spectral mode.

In order to identify the dominant wavenumbers k_z, it is necessary to quantify the contribution of each scale to the global turbulent kinetic energy of the system. From a mathematical standpoint, this translates into the computation of the spectral energy integrated along the inhomogeneity coordinate. Defining A(k_z,x,y,t) as the normalised amplitude, the energy associated with a specific wavenumber for each longitudinal station and time instant is defined by the integral in Eq. 41:

$$E(k_z, x, t) = \int_{y_{min}}^{y_{max}} |A(k_z, x, y, t)|^2 \, dy \tag{41}$$

This integral represents the energy density associated with the spatial scale $\lambda_z = 2\pi/k_z$; the physical objective is to capture the energy content of coherent structures that possess a significant energy projection over the entire y-axis.

In the algorithm, the operation is implemented as reported in Eq. 42 by discretising the integration using the trapezoidal rule, which offers an optimal compromise between computational efficiency and accuracy:

$$E(k_z, x, t) \approx \sum_{i=0}^{N_y-1} \frac{|A_i|^2 + |A_{i+1}|^2}{2} (y_{i+1} - y_i) \tag{42}$$

To optimise the pipeline, the code adopts the following measures:

- Temporal Slicing: To reduce memory impact, the computation is performed on a snapshot subset via the time_step parameter.
- Vectorisation: Integration is handled through the use of np.trapezoid, operating simultaneously on all spectral modes to increase processor efficiency.
- Temporal Reduction: Instead of a temporal mean, which would tend to dampen intermittent but physically crucial events, the code extracts the maximum energy value along the N_t axis as highlighted in Eq. 43:

$$E_{\max}(k_z, x) = \max_{t \in [0, N_t]} \{E(k_z, x, t)\} \tag{43}$$

This maximum energy projection methodology guarantees that the selection of k_z follows the maximum achievable energy potential of a given scale during the entire observation interval. This aspect is particularly critical for DMD analysis: by identifying

modes characterised by the highest energy peaks, it is ensured that the subsequent decomposition focuses on the structures responsible for perturbation growth and momentum transport, while simultaneously filtering out low-energy stochastic fluctuations and purely local dissipative scales.

4.6.3 MPI Strategy and Global Diagnostics

Given the distributed nature of the data along the longitudinal x-axis, the computation of an energy spectrum representative of the entire physical domain cannot forgo a collective communication operation. Each MPI rank in fact holds only a portion of the spatial domain, and these subdomains may have different extents due to the not necessarily uniform load distribution.

The aggregation strategy is structured to minimise traffic, moving only the pre-processed 2D data instead of the original 4D tensors; the process is articulated in three main phases:

1. Dimensionality Reduction (Local Compression): Each rank locally executes the operations described in the preceding sections, transforming the four-dimensional tensor $\hat{u}(t, k_z, x, y)$ into two real two-dimensional maps: $E_{local}(k_z, x)$ and $A_{local}(k_z, x)$; in this way the communication payload is considerably reduced.

2. Synchronisation: Unlike a standard gather, which requires each process to send the same amount of data, the implementation uses gatherv, which allows management of the inhomogeneity of x-points assigned to the different ranks. From an implementation standpoint, the algorithm follows these steps:

   - Displacement Determination: rank 0 collects the number of points $n_{x,local}$ from each process to compute the counts and displs vectors.
   - Data Transfer: The local buffers are flattened and sent to rank 0, which recomposes them into a single global array using the previously computed pointers.

3. Reconstruction and Spatial De-aliasing: Once global data are available on rank 0, the algorithm addresses the problem of potential spatial overlap. As already explained, the domain distribution strategy includes overlapping zones between ranks to ensure continuity; rank 0 must therefore perform a Unique Mapping operation through the np.unique function. In this way the software identifies the unique x-coordinates and associates each with the corresponding energy value, ensuring that the final spectrum does not exhibit artefacts due to multiple count-

ing of interface zones between subdomains.

A further critical aspect is the management of data types: while spectral computation occurs in complex precision, aggregation for energy map diagnostics occurs in MPI.DOUBLE, so as to improve the precision of the final plots. Furthermore, the implementation includes a spatial sampling mask; this measure allows generation of readable spectra even for extremely long domains, avoiding excessive curve crowding in the diagnostic plots without, however, losing spectral resolution on the wavenumbers k_z. The result is a global diagnostic map enabling the subsequent spectral truncation of the DMD.

Fig. 4.13 summarises the complete workflow for spectral energy analysis and MPI aggregation: from the local normalisation and wall-normal integration performed independently by each rank, through the collective gatherv communication, to the global de-aliasing and spectral diagnostics executed on rank 0, which produce the energy map used for kz selection in the subsequent truncation step.

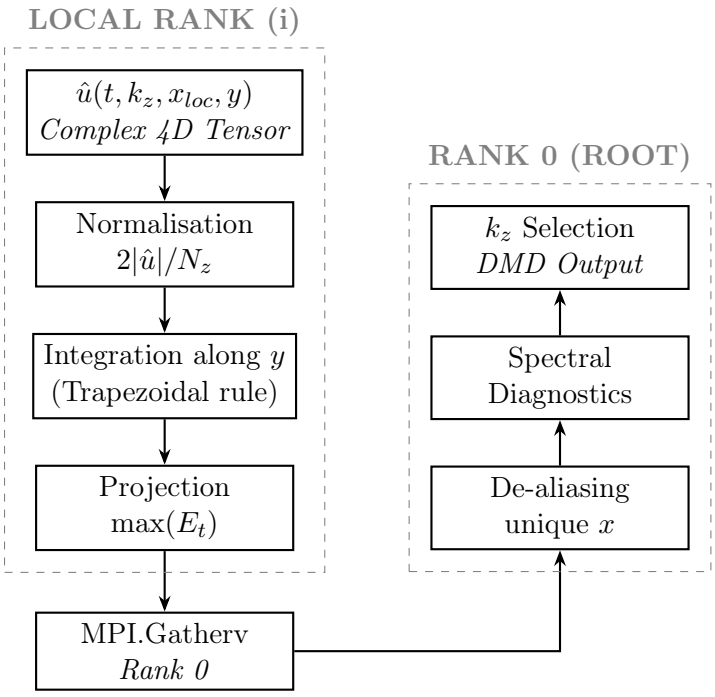


Figure 4.13: Compact workflow for data reduction and MPI aggregation.

4.6.4 Truncation Criterion for Modal Decomposition (DMD)

Including all wavenumbers resolvable by the numerical grid within the DMD input tensor is not a computationally feasible practice, nor is it of physical interest. It is in fact fundamental to select a subset of k_z, not only to reduce the size of the data to be

handled, but also to guarantee the identifiability of the dynamic modes.
Spectral truncation responds to two requirements:

- Physical Requirement: Referring to the energy distribution in the flow, turbulence is composed of structures of various sizes; the most important for system dynamics are the integral scales (low wavenumbers), which contain nearly all the kinetic energy and drive transport and transition processes.
  As k_z increases, the energy decays following Kolmogorov's law until it reaches the dissipative scales where energy is converted into heat by viscosity. These scales are extremely small, chaotic and influenced by numerical truncation errors; including them would mean analysing the system's noise. It is therefore necessary to eliminate them, focusing solely on coherent structures (such as streaks and vortices) that have a real impact on fluid evolution.

- Numerical Requirement: DMD is based on Singular Value Decomposition (SVD). If data from wavenumbers with near-zero energy are inserted, the snapshot matrix would be contaminated by stochastic components, and the DMD eigenvalues would be unstable and difficult to interpret.
  From a mathematical standpoint, a matrix containing too much irrelevant information is often ill-conditioned. If, instead, the wavenumbers k_z are reduced, the numerical stability of the Moore-Penrose pseudoinverse computation, which lies at the basis of DMD analysis, is dramatically improved.
  Finally, working with a restricted set of modes allows management of smaller matrices, speeding up computation and substantially reducing RAM usage, potentially allowing analysis of longer temporal windows or larger spatial domains with the same available resources.

4.7 Algorithmic Implementation of DMD

In order to extract the coherent structures of the flow, an architecture structured on two levels has been implemented. The first level, represented by the dmd_analysis function, constitutes the main core of the DMD analysis, while the second level, implemented in the *serial_window_dmd_on_fft* function, manages the systematic application of the first level across the entire physical domain.

4.7.1 The dmd_analysis Function

The dmd_analysis function constitutes the computational core of the methodology, implementing the Exact DMD algorithm derived in §3.1.2 (Eqs. 24–27) directly on

the complex coefficients derived from the rFFT, with the objective of transforming a temporal sequence of snapshots into a reduced linear dynamical system, optimising numerical stability.

A dynamic threshold filter has also been incorporated at this stage: the rank of the matrix is computed automatically and only modes whose singular values (s) do not exceed the energy of the principal mode s[0] by more than 1% are retained. This choice guarantees that subsequent computation occurs exclusively in the space of energetically dominant modes. The primary objective of this truncation is to avoid the emergence of non-physical growth rates; without this filter, high-frequency components or those linked to numerical noise would force the dynamic operator $\tilde{A}$ to include extremely high amplification rates that correspond to no real flow dynamics, these would simply be mathematical artefacts necessary to account for the noise.
By excluding such spurious modes, the extracted dynamical system is guaranteed to be intrinsically stable and the growth or decay of DMD modes faithfully reflects the flow evolution, preventing artificial divergences.

Finally, the amplitude of each DMD mode is computed by solving a least squares problem; this approach is preferred over the direct pseudoinverse for better stability management. In fact, should the mode matrix Φ be ill-conditioned, the least squares solution uses SVD-based decomposition internally to provide the most reliable solution, ensuring that the reconstruction of the initial field is the optimal projection onto the space of extracted modes, avoiding numerical divergences in the result synthesis phase.

4.7.2 Sliding Window Strategy and Vectorial Management

The *serial_window_dmd_on_fft* function logically manages the entire decomposition.

The implemented logic is based on a sliding window strategy: the analysis does not consider the longitudinal x-axis as a single block, since coherent structures evolve and change along the path. The domain is therefore subdivided.

Through a for loop, the code generates a list of windows local_windows; each window is characterised by a width WINDOW_WIDTH and consecutive windows are spaced by a STEP_SIZE parameter, specifically, STEP_SIZE will have a value smaller than the window width so that overlap between windows is guaranteed, ensuring continuity between modes extracted from one window and the next.

The core of the implementation lies in the transformation of the extracted data into DMD-compatible data; in particular, for each wavenumber k_z, a chunk of complex data is isolated and subjected to a reshaping and flattening operation:

- Transpose: the first data rearrangement phase consists of reordering the tensor axes; specifically, the x-coordinate representing the window width is brought to the first position and isolated as the evolution variable.
- Reshape: the second phase concatenates the temporal dimension (N_t) and the wall-normal dimension (N_y) into a single axis occupying the Feature axis (coinciding with the rows of the generated matrix), while the window width x becomes the snapshot axis.

For each spatial window of the sliding-window procedure, the locally extracted tensor has dimensions (N_t, W_{size}, N_y) for each of the three velocity components u, v, w: that is, for every streamwise position x_i within the window, all N_y wall-normal points are available for each of the N_t sampled time instants. Before DMD can be applied, this tensor must be transposed and reshaped into a two-dimensional matrix, in which columns correspond to the streamwise positions x_i (acting as the snapshot index), while rows stack, for each column, all the y and t values of the three velocity components.

Fig. 4.14 illustrates this step, restricted for clarity to the u-component only (v and w are stacked below following the exact same scheme). The column corresponding to position x_1 is shown "expanded" into its colored sub-blocks, one for each time instant t: each block collects all N_y points associated with that instant. The same color coding is reported on column x_1 of the resulting matrix, to highlight how the row ordering reflects the structure of the original tensor (at fixed t, all values of y vary first, before moving to the next time instant). The entire matrix construction procedure is repeated independently for each wavenumber k_z obtained from the FFT along the spanwise direction. In this configuration, DMD does not analyse the temporal evolution of coherent structures but their spatial evolution. The columns of the data_matrix represent successive portions of the longitudinal domain along x. Consequently, the extracted eigenvalues do not describe temporal frequencies, but spatial growth rates and longitudinal wavenumbers; in this way it is possible to identify how different structures amplify or decay as they propagate along the longitudinal axis.

The final implementation phase concerns the management of the physical dimensionality of the problem and the preservation of results in a form that allows targeted interrogation.

One of the main aspects concerns the management of the UVW vector field. The function is called on the tensor *UVW_hat*, which aggregates the u (streamwise), v (wall-normal) and w (spanwise) components. From the code perspective, this aggregation triples the row dimension (features) of the data_matrix. This choice has important

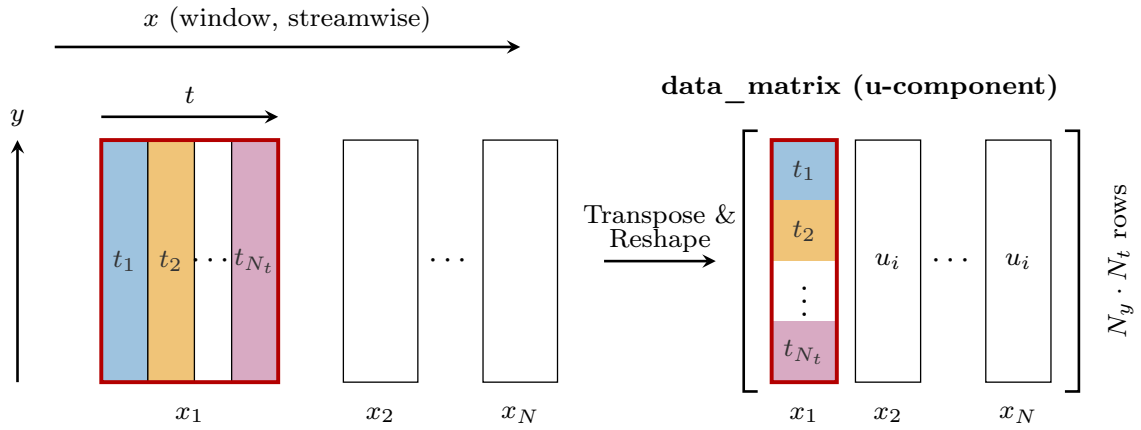


Figure 4.14: Construction of the DMD data matrix: reshaping and flattening of the complex spectral tensor into the snapshot matrix X, with the window width x mapped onto the snapshot axis and the combined (N_t, N_y) dimensions mapped onto the feature axis.

physical implications:

- Cross-Correlation: By performing DMD simultaneously on u, v and w, the algorithm is able to identify whether an instability in one component is intrinsically linked to another (understanding this link is fundamental, for example, for the lift-up mechanism which connects vertical fluctuations to the generation of streamwise streaks).
- Eigenvalue Uniqueness: For each identified coherent structure, the code extracts a single complex eigenvalue; this means that the three velocity components are forced to follow the same frequency and growth rate, capturing the unitary nature of vortices.

The function output is not a simple matrix, but a hierarchical object (list of dictionaries) designed to reflect the architecture of the physical problem. Each element of the results list represents a spatial window and contains the following information:

- Spatial Metadata: local_window and global_window indices, essential for identifying results on the geometry.
- DMD Modes per k_z: A sublist dmd_modes_kz which for each wavenumber archives:
  - eigenvalues: the local stability spectrum,
  - amplitudes: the energy weight of each mode,
  - modes: the complex vectors defining the spatial shape of the structure.

This data organisation is fundamental because it enables the post-processing phase and renders it flexible.

Fig. 4.15 reports a flow diagram clarifying the operation of the two functions analysed above and their interaction.

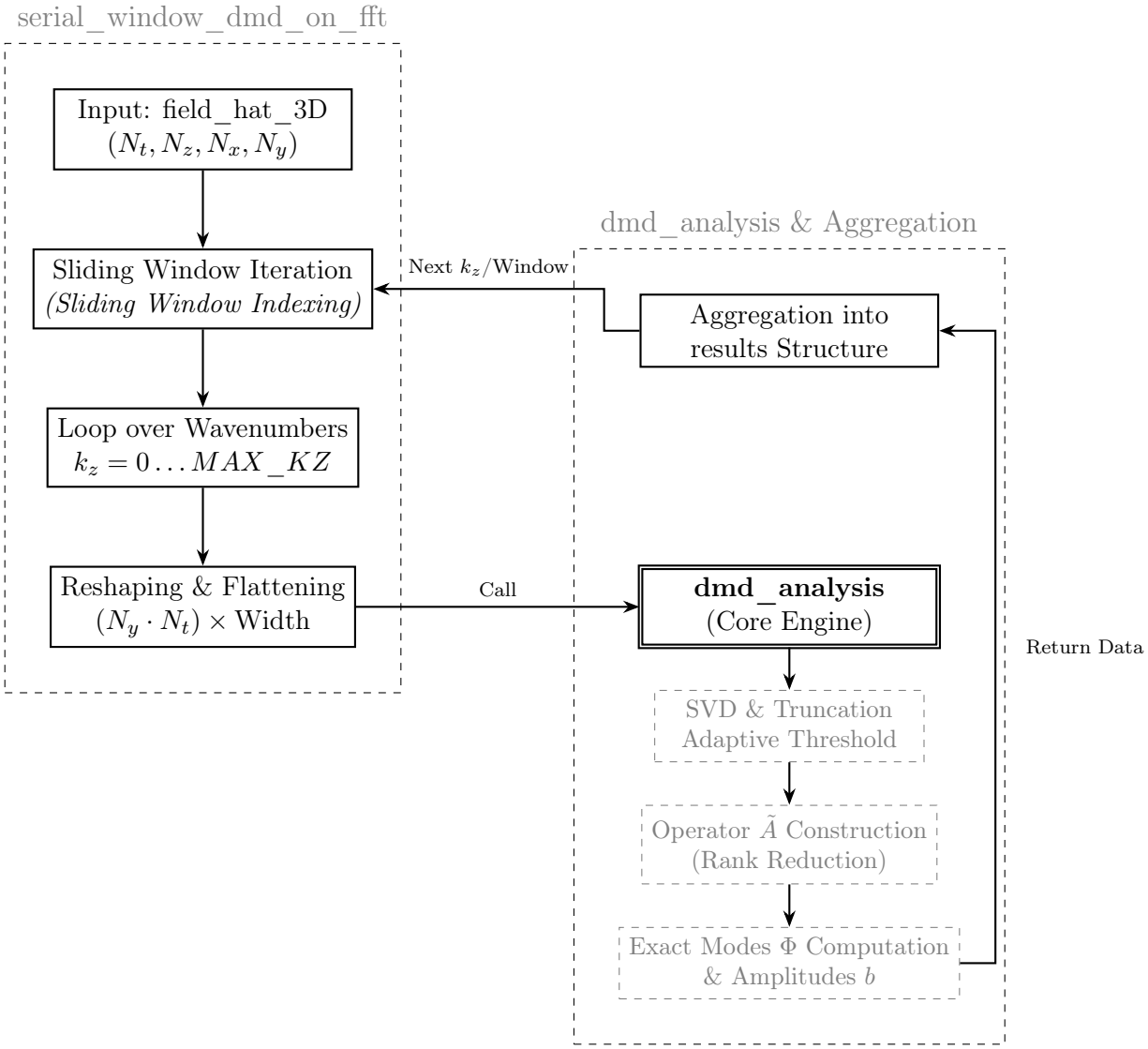


Figure 4.15: Software architecture: integration between spatial management (sliding window) and the exact DMD analysis engine.

4.8 Results Management and Storage: Distributed HDF5

At the end of the DMD, the produced data (complex modes, eigenvalues and amplitudes for each spatial window and for each wavenumber) require efficient storage. To this end, the *save_dmd_to_h5_distributed* function has been implemented, adopting the HDF5 (Hierarchical Data Format) format. This choice is motivated by the format's

capacity to manage multidimensional datasets and complex metadata within a single hierarchical structure.

4.8.1 Distributed I/O Strategy and External Links

In HPC simulation management, data saving often represents a critical bottleneck. An approach in which all processors write to a single shared file would lead to file locking and bandwidth saturation, drastically degrading performance. To avoid this problem, the algorithm implements an asynchronous distributed I/O strategy based on a hierarchical structure.

In the code, each rank works autonomously during the writing phase based on the following implementation:

- Pointer Isolation: each process opens its own file dmd_rank_X_TIMESTAMP.h5. This guarantees that each rank has its own pointer to the file system, eliminating waiting time for access to shared resources.
- Directory Organisation: to maintain order in the workspace, rank 0 preventively creates a dedicated folder (os.makedirs); synchronisation is guaranteed by a comm.barrier that prevents worker ranks from attempting to write before the directory is effectively available on disk.

The crucial aspect of the implementation lies in the creation of a Master file, a lightweight virtual interface generated exclusively by rank 0 upon completion of the global write operations; this is done using the *h5py.ExternalLink* function to create a dataset abstraction that allows:

- Maintaining Modularity: Heavy data remain confined in the data folder, rendering the Master file extremely small and easy to transfer.
- Unified Access Logic: During post-processing, loading the Master file is equivalent to loading the entire simulation. The software resolves links transparently: if the user accesses master['rank_4/data/w0'], the library automatically opens the rank 4 file in the background and retrieves the requested data.

To make saving clear, the implementation exploits HDF5 attributes. The Master file stores global parameters (Nt_total, Y_keep, global_domain), while local files retain metadata relative to the processed domain portion (domain_start, domain_end).
This structure ensures that, even by isolating a single rank file, it is possible to trace its physical position in the domain, guaranteeing analysis integrity even in case of partial dataset corruption.

4.8.2 Performance Optimisation and Dataset Structure

As seen, the external links strategy resolves the bandwidth saturation problem; however, it is important to highlight other internal aspects of the code that allow maximisation of write speed and minimisation of HDF5 system overhead.

In an HDF5 file, each created dataset entails a metadata write in the file header. Considering the presence of a large number of spatial windows and wavenumbers, creating an excessively deep hierarchy would drastically slow down the process; the code therefore adopts a flatter, less branched structure.
Rather than creating a subgroup for each k_z, datasets are saved directly within the window group (w_group) using identifying prefixes. This technical choice reduces the number of nodes in the HDF5 tree, speeding up both file closure by the rank and subsequent indexing by the master file.

Although the h5py library supports compression filters such as gzip, the final implementation favours speed (Raw Speed). Removing compression during the saving phase is a strategic choice for the following reasons:

- CPU Load Elimination: computing compression algorithms on large matrices would engage processors at a moment when the sole objective is to free RAM.
- Disk Saturation: Operating in an HPC environment, it is preferable to generate larger files but with minimal write times, leaving any compression to subsequent long-term archiving phases.

Of fundamental importance is also understanding how complex data are handled. HDF5 does not possess a universally compatible complex data type; the capability of h5py to automatically map complex numpy arrays into a structured data type composed of two floats (real and imaginary) is therefore exploited. This guarantees that the memory layout remains contiguous, allowing the system to perform direct I/O operations from memory to disk without the need for intermediate transformations.

4.8.3 Read Efficiency (Post-Processing)

The implementation choices described produce benefits not only during writing, but are also decisive during post-processing.

The use of External Links and the flat structure enables a selective data access mode; in this way, during analysis or visualisation, it is not necessary to load the entire database into RAM. The user can open the Master file and selectively access the single window w_i or single wavenumber k_z of interest.

Furthermore, this approach allows considerable reduction of the memory footprint: the HDF5 library, thanks to the links, opens only the file of the specific rank containing the requested data, keeping memory consumption extremely low even in the case of datasets exceeding the available RAM capacity.

This system transforms the dataset from a static archive into an interactive interface, enabling complex analyses to be conducted efficiently and at global scale.

Fig. 4.16 reports the hierarchical schema with which data are saved.

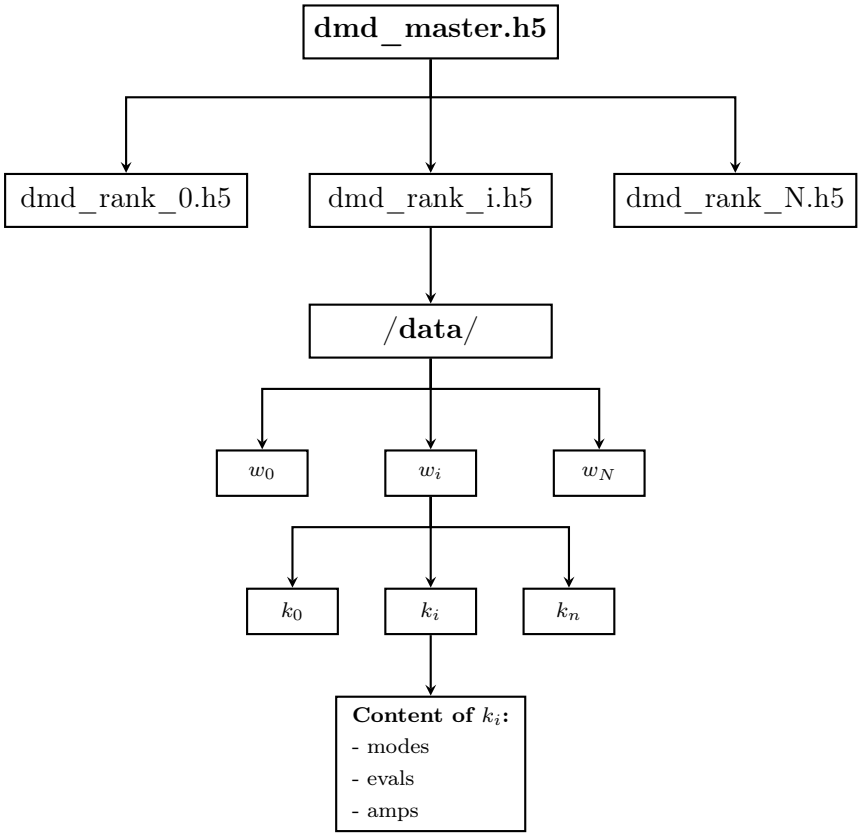


Figure 4.16: Hierarchical structure of the HDF5 database: logical organisation by rank, spatial windows (w) and spectral wavenumbers (k).

5 Validation and Performance Analysis

5.1 Validation of Data Redistribution (All-to-All)

The integrity of the data during the transition between temporal partitioning (input) and spatial partitioning represents the cornerstone of the parallel computing algorithm. In a distributed context this phase is highly complex since each MPI rank must manage the decomposition of its own local snapshots and the subsequent re-routing towards the destination ranks via All-to-All collective communication.
Given the criticality of this phase for the correctness of the final result, it is indispensable to verify its correct implementation in order to assess its reliability.

The validation presented here aims to exclude critical errors such as memory pointer misalignment, temporal chronological inversion, or loss of numerical precision during the packing and unpacking operations of the buffers.

5.1.1 Global Transposition Mechanism and Index Mapping

The distribution process described in the previous chapter acts as a distributed tensor transposition. Considering the global dataset as a tensor

$$\boldsymbol{X} \in \mathbb{R}^{n_x \times n_y \times n_z \times n_t},$$

the initial configuration sees each rank holding a complete temporal submatrix for a portion of the space. The All-to-All operation must guarantee that every element $\boldsymbol{X}(i, j, k, t)$ is correctly delivered to the rank that manages the spatial window containing the considered point (i, j, k).
The schema in Fig. 5.1 reports the logic of the validation of the parallel framework.

In order to optimise the exchange, as previously described, the data are linearised into contiguous buffers. The objective of the validation is to confirm that the logic used to fill the buffers is consistent with the expected ordering, preventing data corruption phenomena arising from accesses to incorrect memory locations.

5.1.2 Analysis of the Spectral Invariance of the Linear Operator A

Dynamic Mode Decomposition is based on the construction of a linear operator A that approximates the system dynamics. If the data redistribution were to introduce even a minimal discontinuity between snapshots or a systematic error in the sampling, the spatial correlation structure would be altered, leading to an incorrect estimate of the

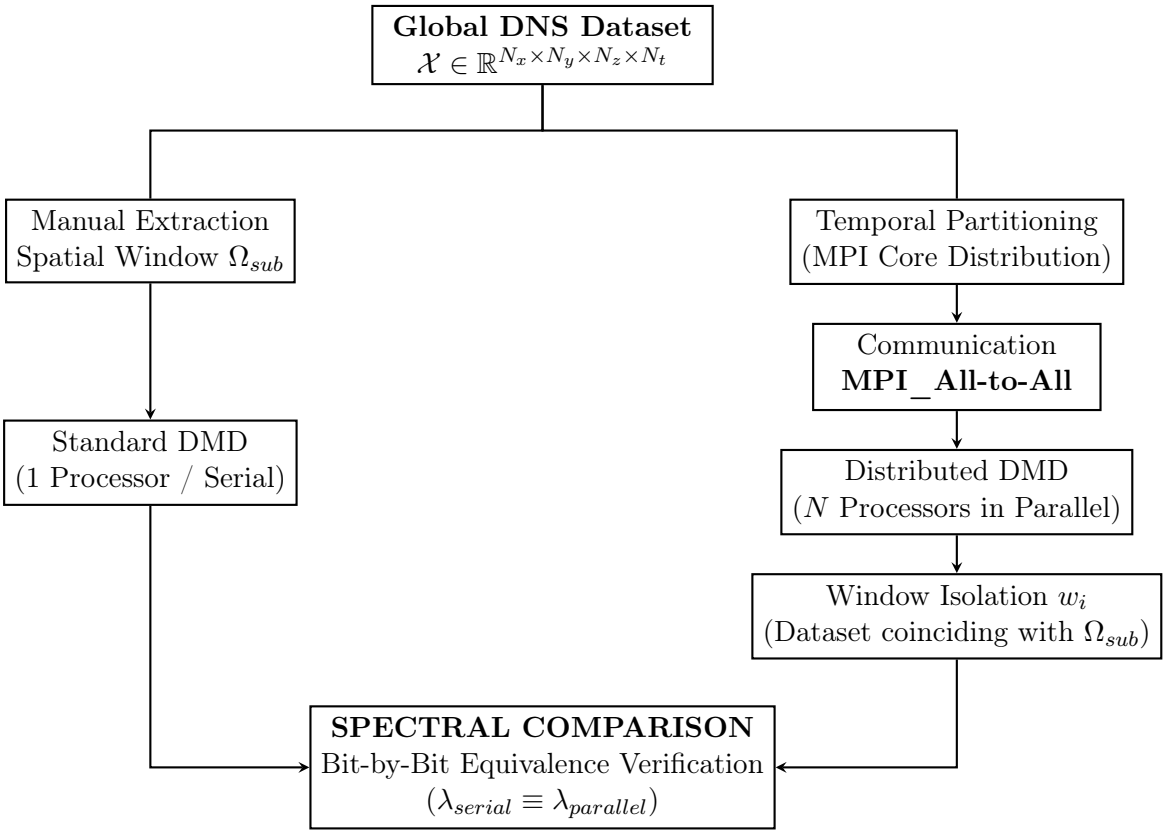


Figure 5.1: Logical diagram of the parallel framework validation procedure: direct spectral comparison between the serial approach on the native subdomain and the output of the local window after MPI redistribution.

discrete eigenvalues μ.

To prevent this, a verification protocol has been developed to check the correct data distribution. In detail, a local spatial window w_0 has been isolated, restricted to a reduced set of 12 snapshots to keep the serial benchmark computationally tractable, and the eigenvalues relative to the selected window have been computed following two distinct paths:

- **Serial Approach (Benchmark):** Execution of DMD on a single core using the entire original DNS dataset, appropriately cropped so as to coincide with the domain of the selected window w_0.
- **Parallel Path:** Execution of the distributed DMD (algorithm under validation) in which the snapshot matrix for the same window w_0 is obtained via All-to-All redistribution among N independent processes.

Fig. 5.2 shows the projection of the discrete eigenvalues μ onto the complex plane for the two considered cases, together with the unit circle, which represents the reference

stability boundary for the discrete DMD operator (a mode is stable if $|\mu| \leq 1$).

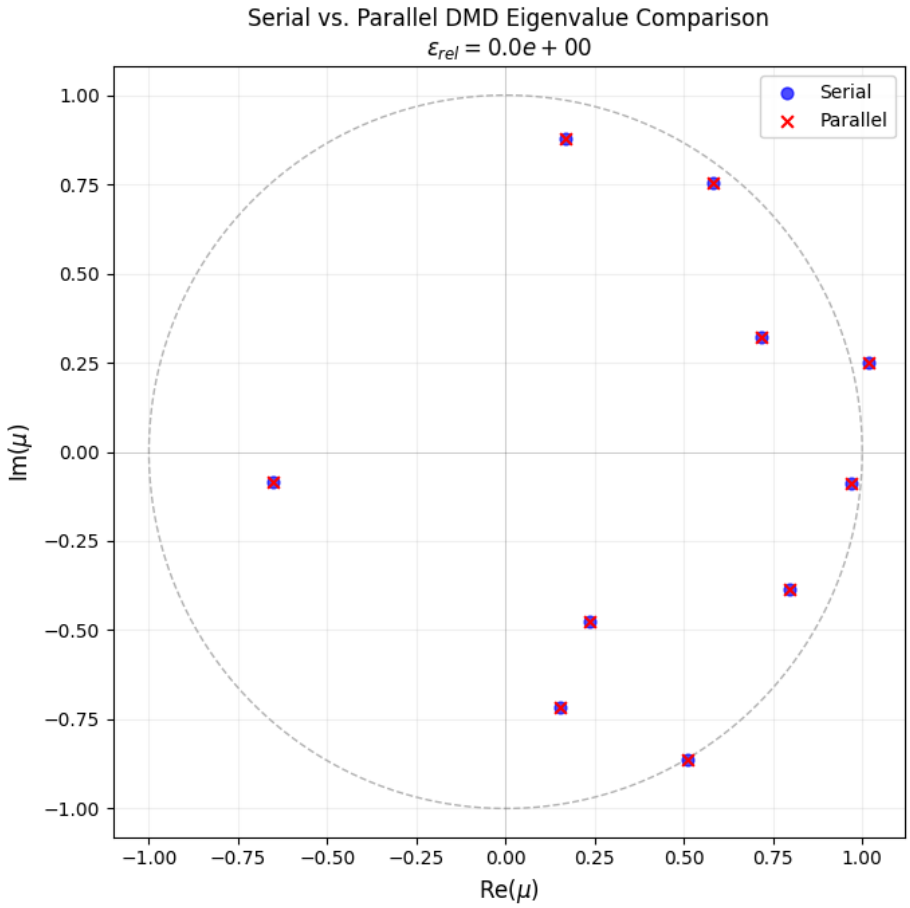


Figure 5.2: Comparison between the discrete eigenvalues μ obtained from the serial benchmark and from the parallel (redistributed) execution, for spatial window w_0 and 12 snapshots. The dashed line represents the unit circle, used as stability reference for the discrete DMD spectrum.

Since the diagonalisation routine does not guarantee a consistent ordering of the eigenvalues between two independent executions, the two sets were first sorted by modulus prior to comparison, in order to avoid a spurious mismatch caused purely by index misalignment rather than by an actual numerical discrepancy. The relative error was then computed as:

$$\varepsilon_{rel} = \frac{\|\mu_{serial} - \mu_{parallel}\|_2}{\|\mu_{serial}\|_2} \tag{44}$$

The comparison yields $\varepsilon_{rel} = 0$, i.e. the two eigenvalue sets are bit-by-bit identical after sorting. This result is even stronger than a simple visual superposition: it confirms that the matrix underlying the SVD (Singular Value Decomposition) has not been

altered in any measurable way by the redistribution process, and that the All-to-All communication introduces no numerical perturbation whatsoever into the spectral content of the window under analysis.

5.1.3 Verification of Temporal Continuity and Local History Integrity

Beyond spectral consistency, it is furthermore fundamental to verify that the redistribution maintains the temporal resolution of the original signal, excluding the possibility that snapshots are delivered in the wrong order.
Fig. 5.3 reports a diagram clarifying the validation logic adopted in order to guarantee the preservation of the temporal chronology of the data.

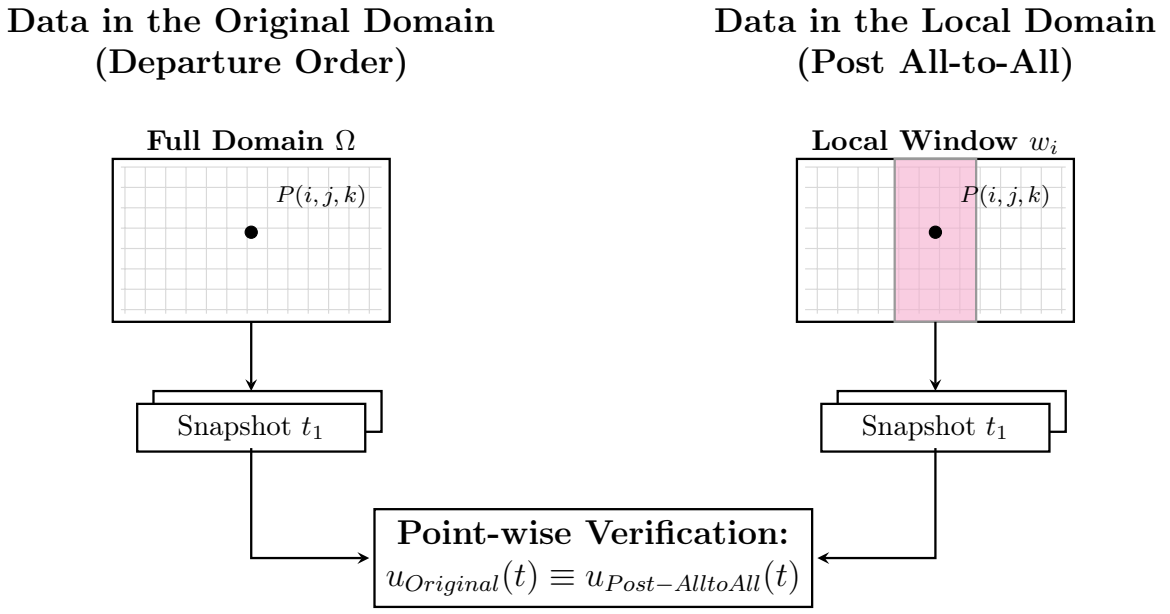


Figure 5.3: Logical diagram of point-wise validation: parallel and symmetric extraction of the time series for the same geometric point $P(i, j, k)$ from the original domain and from the redistributed local window.

To verify what has been described above, a point $P(i, j, k) = (405, 5, 70)$ within the physical domain was selected, and the temporal history of the velocity component $u(t)$ at this point was analysed over a set of 12 snapshots. In particular, the temporal evolution was extracted following two approaches also in this case: the first extraction comes from the departure domain prior to communication (serial reference), while the second is performed at the end of communication by the local destination rank of the selected spatial point (rank 0, in this instance).

Fig. 5.4 reports the temporal history of the velocity component u for the selected spatial point for both extraction modalities performed.

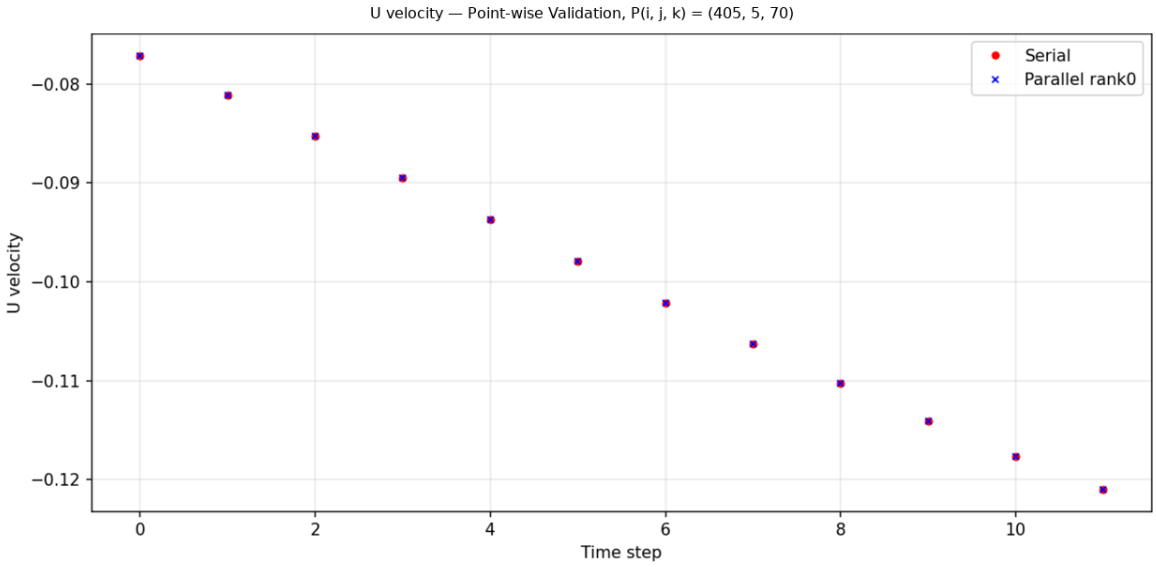


Figure 5.4: Temporal history of the streamwise velocity component u at point $P(i, j, k) = (405, 5, 70)$, extracted from the original domain (Serial) prior to redistribution and from the local destination rank (Parallel, rank 0) after the All-to-All communication, over 12 snapshots.

As shown in the figure, the two series are indistinguishable at every time step, confirming that the reassembly of MPI packets in the destination rank follows the chronological sequence rigorously. In this way, the absence of errors in the buffer computation is confirmed, which would have generated visible discontinuities in the signal.

It is also observed that data precision is maintained unaltered throughout the process, guaranteeing that no information loss occurs.

The success of these validations demonstrates that the decomposition of the domain into independent spatial windows, while fragmenting the physical memory, does not compromise its unity. The implemented algorithm therefore allows operation on local temporal histories, which remain faithful to the original DNS data, guaranteeing that spectral analyses are based on numerically correct data.

5.2 Validation of the DMD Algorithm via Reference Benchmark

Once the spatial decomposition and redistribution procedure via the All-to-All communication mechanism has been completed and verified, the Dynamic Mode Decomposition is computed.
Within the developed framework, as already highlighted, this step is performed in a

serially independent manner on each window w_i into which the domain has been subdivided, and for each wavenumber k_z.
It is therefore fundamental to validate the DMD function implemented in the framework, to exclude the introduction of numerical artefacts. To this end, a cross-validation protocol has been structured. As a case study, the two-dimensional laminar flow around a cylinder in the vortex shedding regime has been selected.

5.2.1 Benchmark Configuration and Workflow

The validation involved a direct comparison, with identical input datasets, between the Python code developed in the framework and the reference MATLAB implementation by Steven L. Brunton.

The test dataset consists of a temporal series of two-dimensional snapshots describing the transient vorticity field in the wake of a cylinder at a Reynolds number $Re = 100$, in which the nonlinear dynamics regenerate a periodic limit cycle. The dataset, obtained from the reference material accompanying the DMD book by Kutz et al., comprises 151 snapshots of the vorticity field, sampled on a spatial grid of 199×449 points, at a fixed sampling interval $\Delta t = 0.2$ (non-dimensional time units). The vorticity field was preferred over the raw velocity components since it automatically satisfies the incompressibility constraint and provides a cleaner visualisation of the coherent wake structures.

The input data matrices X and X' (shifted from each other by a temporal interval Δt) were provided in parallel to both solvers and defined as follows:

$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \dots & x_{m-1} \\ | & | & & | \end{bmatrix}, \quad X' = \begin{bmatrix} | & | & & | \\ x_2 & x_3 & \dots & x_m \\ | & | & & | \end{bmatrix} \tag{45}$$

where the vector x_k is defined as follows:

$$x_k = \begin{bmatrix} x_k(t_1) \\ x_k(t_2) \\ \vdots \\ x_k(t_n) \end{bmatrix}, \qquad k = 1, \dots, m \tag{46}$$

Both codes perform the decomposition addressing the same algebraic steps:

- Singular value decomposition of the initial matrix: $X \approx U_r \Sigma_r V_r^*$

- Projection and construction of the reduced transition matrix: $\tilde{A} = U_r^* X' V_r \Sigma_r^{-1}$
- Diagonalisation of $\tilde{A}$ to derive the complex eigenvalues μ_i and the corresponding eigenvectors w_i
- Reconstruction of the global spatial DMD modes via projection: $\Phi = X' V_r \Sigma_r^{-1} W$

For this benchmark, the SVD truncation rank was fixed at $r = 21$, a value consistent with the standard configuration adopted in the reference literature for this test case, chosen so as to retain the dominant energetic content of the flow while excluding the numerically unstable, low-energy tail of the spectrum.

5.2.2 Quantitative Analysis of Spectral Invariance

The most sensitive indicator of mathematical fidelity of the algorithm is the discrete spectrum of eigenvalues μ_i obtained directly from the diagonalisation of the reduced operator $\tilde{A}$, whose modulus establishes the amplification or damping rate of the mode ($|\mu| \leq 1$ for stable modes, i.e. within the unit circle), while the argument maps the associated phase rotation between successive snapshots.

Fig. 5.5 reports the projection of the discrete eigenvalues μ obtained with the two tested DMD codes (MATLAB and Python) onto the complex plane, together with the unit circle as stability reference.

A bit-by-bit superposition is shown between the two spectra. From a strictly numerical standpoint, the deviation between the MATLAB eigenvalues μ_{MATLAB} and those of the implemented Python code μ_{Python} falls below the double precision tolerance:

$$\varepsilon_{rel} = \frac{\|\mu_{MATLAB} - \mu_{Python}\|_2}{\|\mu_{MATLAB}\|_2} < 10^{-15} \tag{47}$$

The perfect stochastic and energetic coincidence excludes any anomaly in the singular value matrix inversion algorithm or in the management of the rank $r = 21$ truncation. Physically, this attests that the developed DMD code is capable of isolating with precision the frequency associated with vortex shedding and all its higher-order nonlinear harmonics, on the reference cylinder wake dataset described in §5.2.1.

5.2.3 Conservation of Spatial Modes

To further corroborate the correctness of the implemented code, beyond the eigenvalue check, a comparison of the spatial modes has also been carried out; indeed, operations intrinsic to the SVD or to diagonalisation in different environments can introduce

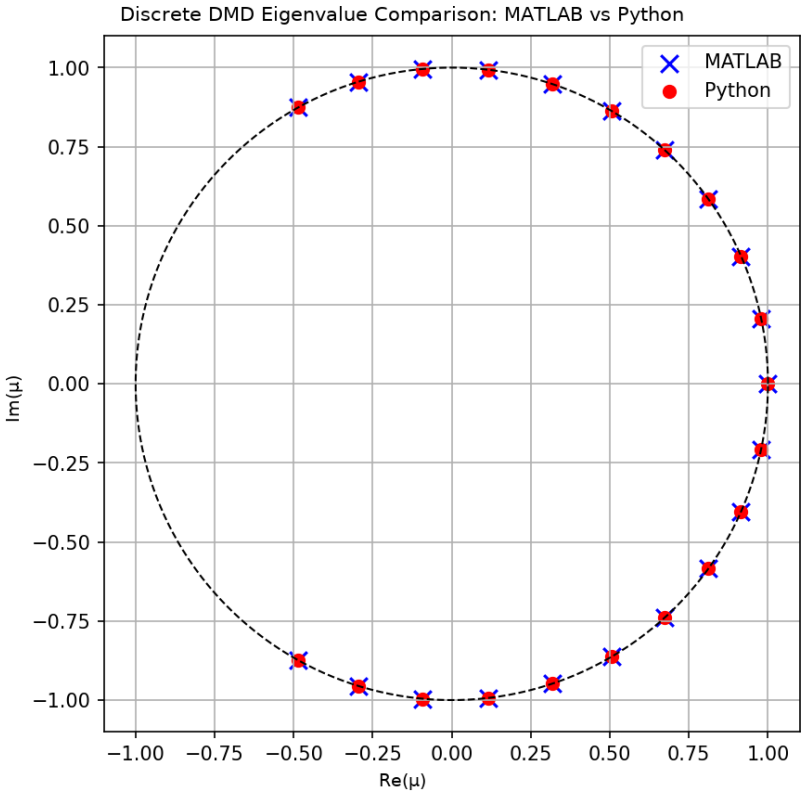


Figure 5.5: Comparison between the discrete eigenvalues μ obtained from the reference MATLAB implementation and the Python implementation developed in this work, on the cylinder wake benchmark dataset ($Re = 100$, $r = 21$). The dashed line represents the unit circle, used as stability reference for the discrete DMD spectrum.

arbitrary phase rotations or sign inversions in the complex vectors, compromising the physical interpretation of coherent turbulent structures.

The second validation phase involved two-dimensional mapping and geometric comparison of the spatial fields associated with the complex modes Φ_i. Since the DMD modes are only defined up to an arbitrary complex phase factor, a phase-alignment procedure was applied prior to comparison: for each pair of corresponding modes (matched by descending eigenvalue modulus), the optimal phase offset was computed via the complex inner product $\langle \Phi_{MATLAB}, \Phi_{Python} \rangle$, and a normalised similarity metric was evaluated as:

$$\text{similarity} = \frac{|\langle \Phi_{MATLAB}, \Phi_{Python} \rangle|}{\|\Phi_{MATLAB}\|_2 \, \|\Phi_{Python}\|_2} \tag{48}$$

Three representative modes were selected for visual inspection: modes 1, 5 and 10, ranked by decreasing eigenvalue modulus, so as to cover a broad portion of the retained spectrum rather than restricting the comparison to the most energetic, and therefore

most numerically robust, leading modes. Fig. 5.6 reports the spatial contour plots of the real part of these three modes, comparing the MATLAB reference (top row) against the Python implementation after phase alignment (bottom row), together with the cylinder geometry for physical reference.

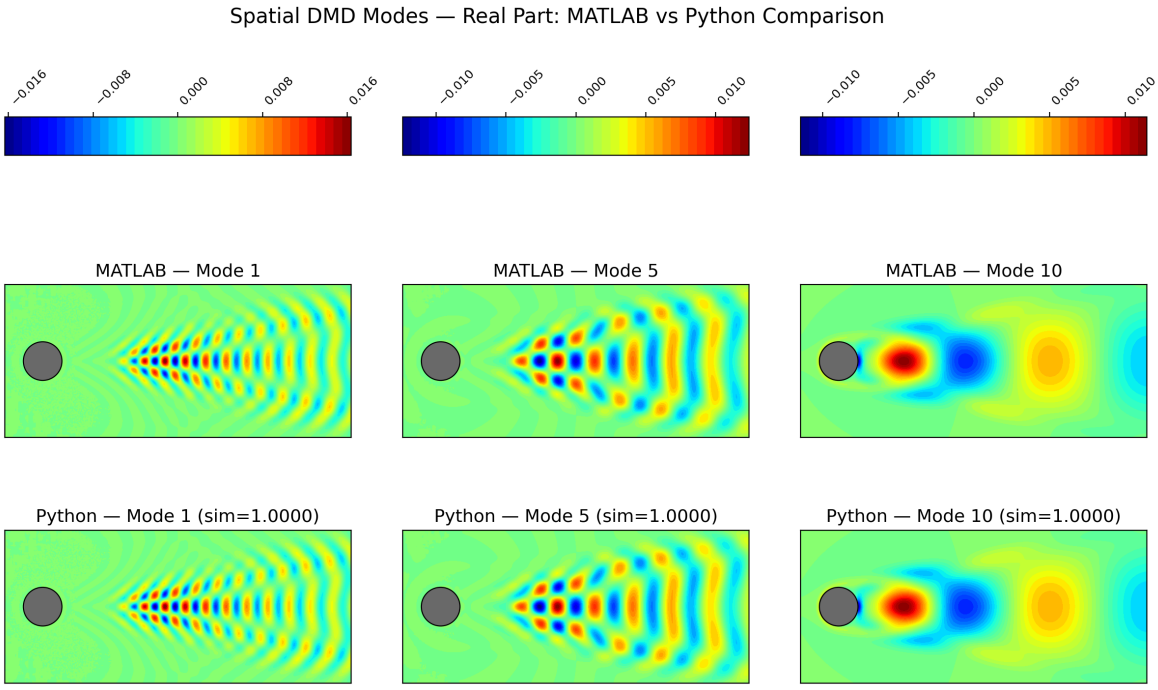


Figure 5.6: Spatial contour plots of the real part of DMD modes 1, 5 and 10 (ranked by decreasing eigenvalue modulus), comparing the MATLAB reference implementation (top row) against the Python implementation after phase alignment (bottom row), on the cylinder wake benchmark ($Re = 100$, $r = 21$). The grey disk represents the cylinder geometry.

The alternating positive and negative macrostructures, typical of the symmetric Von Kármán wake, preserve the exact spacing, the geometric extent of the vorticity cores, and the correct phase orientation across all three modes. Mode 10, in particular, exhibits a qualitatively different spatial structure with respect to the leading modes, dominated by a pair of steady, large-scale recirculation regions rather than the alternating wave pattern of modes 1 and 5, and is nonetheless reproduced with the same fidelity. The similarity metric evaluated over the full set of $r = 21$ retained modes yields a minimum value of 1.0000 across the entire spectrum, with no measurable degradation for higher-order, less energetic modes.

Since DMD modes are complex-valued, encoding both spatial amplitude and phase, the comparison was extended to the imaginary component. Fig. 5.7 reports the real and imaginary parts of the leading mode (Mode 1) for both implementations.

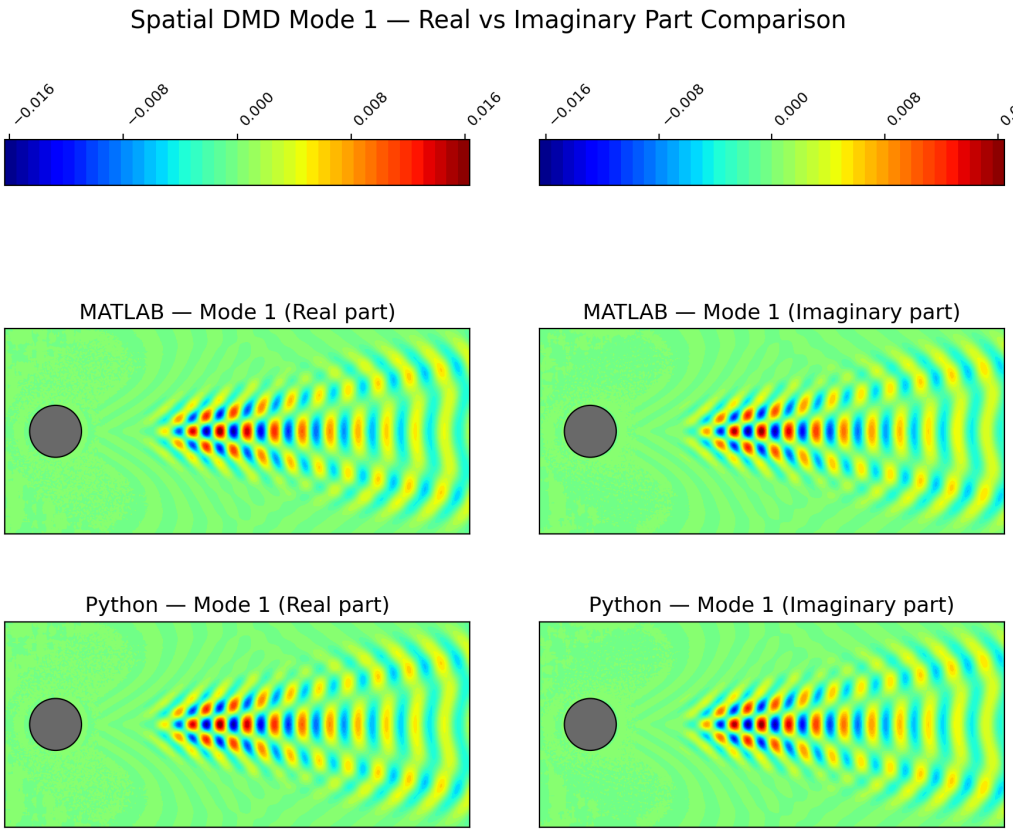


Figure 5.7: Real and imaginary parts of the leading DMD mode (Mode 1), comparing the MATLAB reference implementation (top row) against the Python implementation after phase alignment (bottom row).

The two components exhibit the expected quarter-wavelength spatial shift, consistent with the travelling-wave nature of the oscillatory DMD mode, and are reproduced identically between the two codes.

No distortion phenomena at the domain boundaries or attenuations due to incorrect computation of the initial mode amplitudes b_i are observed. This level of reproducibility, together with the spectral invariance demonstrated in §5.2.2, ensures that the Python solver preserves the physical information content of the reference dataset in its entirety.

5.3 Scalability Analysis

Before initiating parallel computation it is fundamental to estimate the memory requirements to avoid failures due to Out Of Memory phenomena or, conversely, unnecessary wastage of resources.

To this end, as already explained in the previous chapter, a module has been introduced

that allows prior estimation of the memory required to manage the selected dataset. In particular, the code receives the following input parameters:

- Geometric and temporal dimensions of the global DNS domain
- Specifications of the cluster nodes used (e.g. gigabytes of RAM available per compute node)

The code analyses the data weight in single precision and the temporary buffers required by the decomposition functions and MPI primitives. The predictive memory estimate is made at the end of each phase at the moment of maximum memory usage, and the most severe condition multiplied by a defined safety factor is then taken as reference.
On the basis of these constraints, the code returns the minimum number of HPC nodes necessary to have the memory required for problem management. Once the number of nodes has been established, the number of active processors (MPI ranks) within them can be varied to analyse performance. Fig. 5.8 illustrates two alternative resource allocation strategies within homogeneous HPC nodes: a low-parallelisation configuration with fewer, memory-rich processes per node (left), versus a high-parallelisation configuration with more, memory-constrained processes per node (right). This flexibility allows the predictive module to explore the trade-off between per-rank memory availability and the degree of parallelism achievable within the fixed RAM budget of each node.

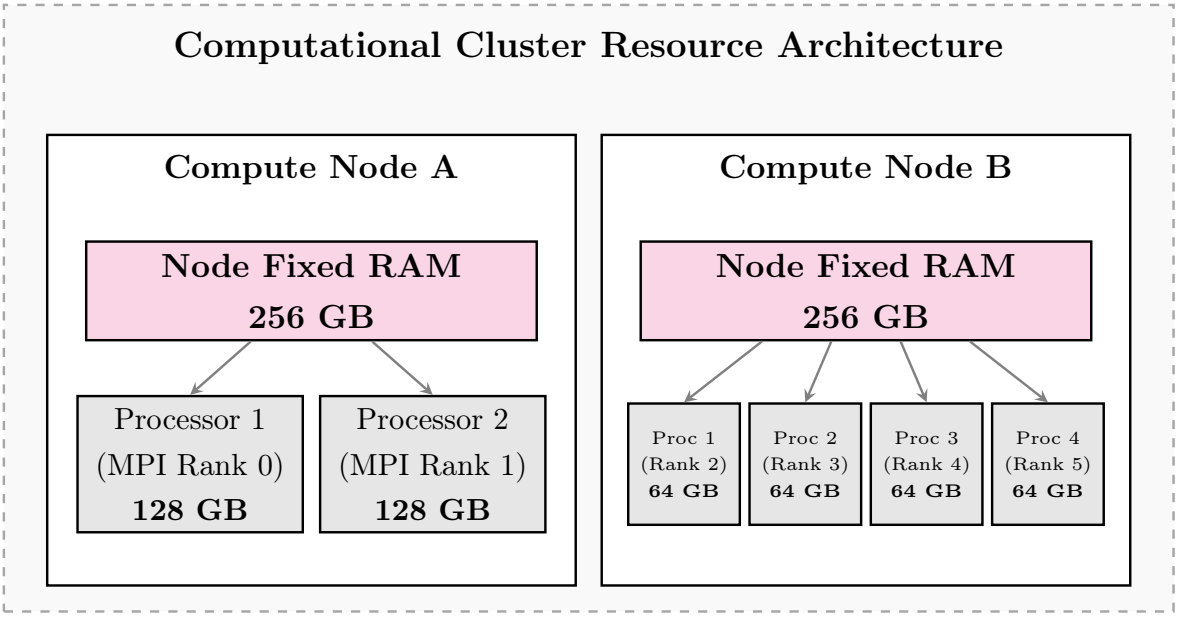


Figure 5.8: Schematic representation of flexible RAM resource allocation within homogeneous nodes of an HPC cluster.

Fig. 5.9 summarises the logical workflow of the predictive memory dimensioning module: starting from the physical dimensions of the DNS domain and the hardware specifications of the target cluster, the module computes the raw tensor weight, estimates the temporary buffer overhead introduced by the decomposition and MPI communication primitives, and finally evaluates the RAM constraint to determine the minimum number of compute nodes required.

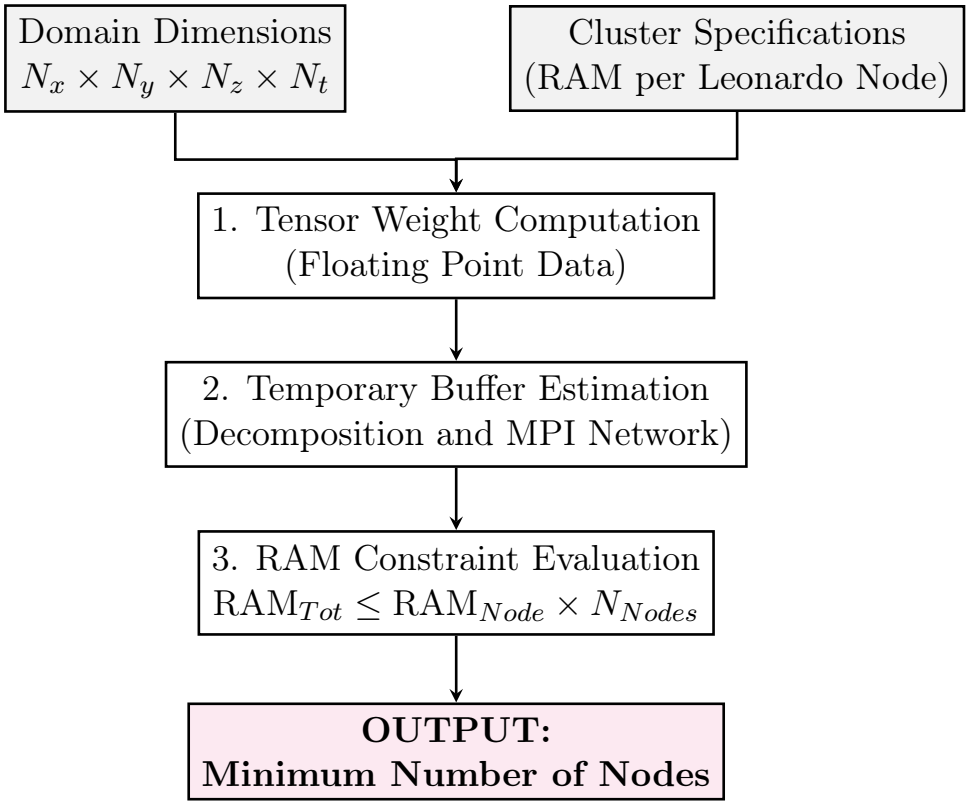


Figure 5.9: Logical flow diagram of the predictive module for memory dimensioning: integration of the physical constraints of the DNS database and the hardware specifications of the Leonardo supercomputer for determining the minimum number of compute nodes.

5.3.1 Stability Analysis and Saturation Point

With the overall domain size and minimum number of nodes fixed as per the theoretical estimate, the algorithm was executed allocating a progressively increasing number of processors (MPI ranks), keeping the number of physical compute nodes fixed at four. The test case comprises $N_t = 1500$ temporal snapshots, with $Y_{POINTS\_TO\_KEEP} = 50$ wall-normal points retained per velocity component.

Table 5.1 reports the total execution time and the corresponding speedup, relative to the 12-rank baseline, for the tested configurations.

Ranks	Total time (s)	Speedup
12	1178.42	1.00
14	997.48	1.18
16	972.57	1.21
18	930.27	1.27
20	781.45	**1.51**
24	793.78	1.48
28	820.94	1.44
32	814.37	1.45
36	822.80	1.43

Table 5.1: Total execution time and speedup relative to the 12-rank baseline, for a fixed number of 4 compute nodes.

Fig. 5.10 reports the same data graphically, showing the total execution time and the corresponding speedup as a function of the number of processors.

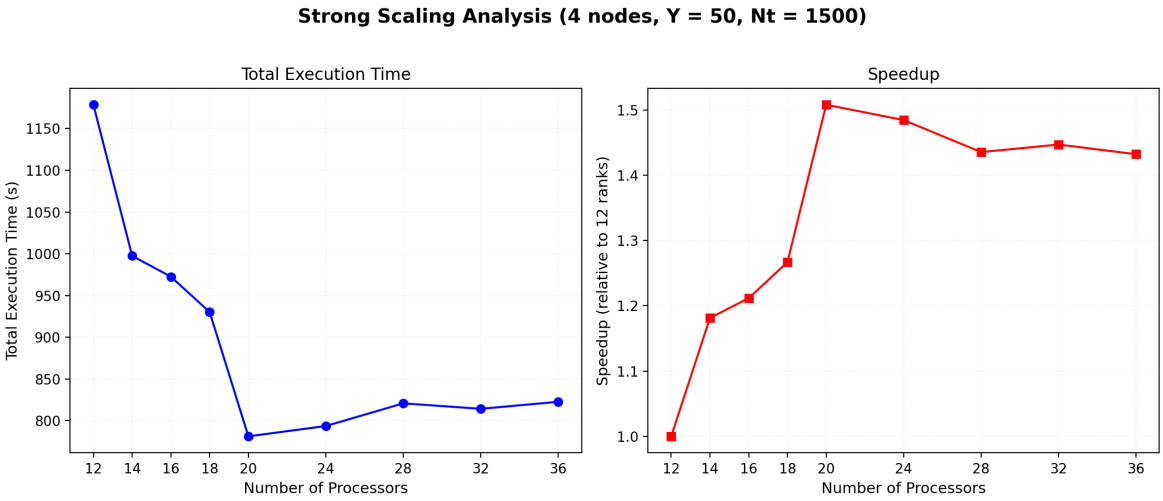


Figure 5.10: Strong scaling analysis: total execution time (left) and speedup relative to the 12-rank baseline (right), for a fixed number of 4 compute nodes, $N_t = 1500$, $Y = 50$.

Under ideal conditions of perfect scalability, doubling the number of processors should halve the computation time. However, in HPC systems the experimental trend of execution times exhibits a behaviour with a minimum, rather than a monotonic decrease.

It is observed that for a limited number of processors (12 to 20), the total execution time decreases as the number of cores increases, reaching an absolute minimum at 20 ranks (781.45 s, speedup 1.51×). In this phase, the benefit of domain decomposition across multiple processors clearly outweighs the costs of parallelism management.

Beyond this point, increasing the number of processors does not yield further improvement: from 20 to 36 ranks, the execution time plateaus in a narrow band between approximately 780 and 823 seconds, with the speedup slightly decreasing from 1.51× to 1.43×. The domain subdivision becomes increasingly fragmented: the amount of data exchanged between individual processors decreases, while the frequency and relative overhead of MPI communications increases, progressively shifting the algorithm from a compute-bound to a communication-bound regime.

5.3.2 Temporal Profiling and Decomposition of Algorithmic Phases

To understand the nature of the inversion point and isolate the bottlenecks, a second diagnostic analysis was performed that does not merely compute the global time, but evaluates the temporal duration of each individual macro-phase of the framework: data reading, MPI redistribution, FFT analysis, DMD analysis, and result saving.

The results of this profiling are visualised in Fig. 5.11, a cumulative bar chart in which the total height of each bar represents the total execution time for a given number of processors, subdivided into coloured segments associated with each individual phase.

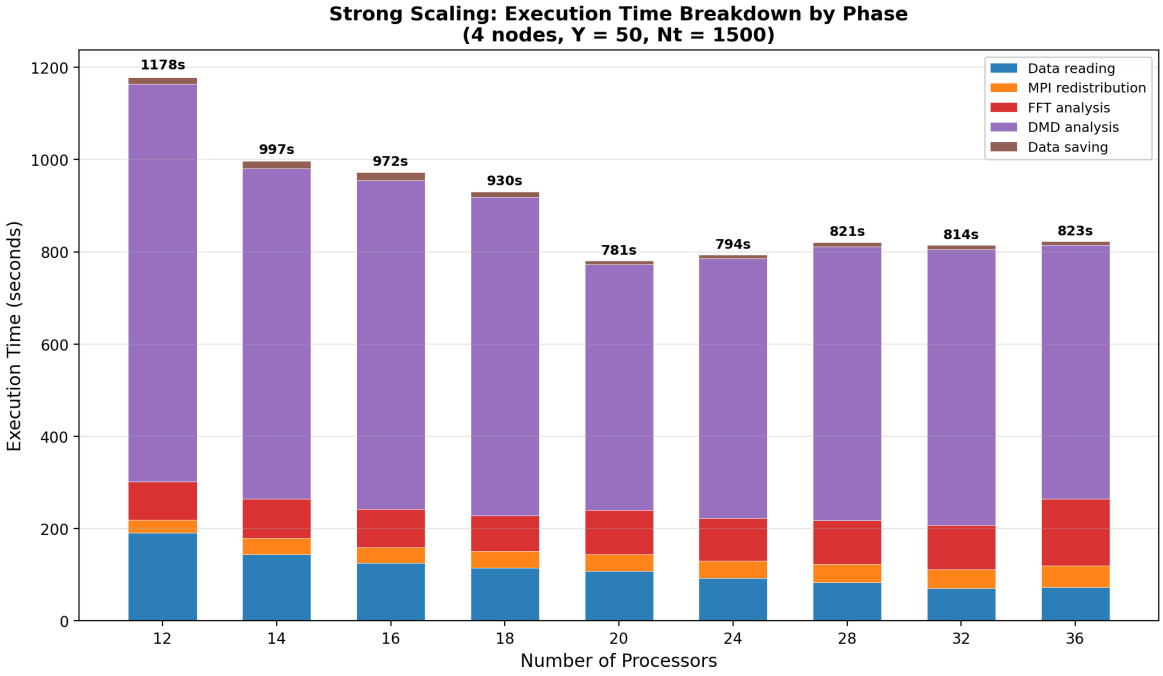


Figure 5.11: Execution time breakdown by algorithmic phase, as a function of the number of processors, for a fixed number of 4 compute nodes, $N_t = 1500$, $Y = 50$.

Across all tested configurations, the DMD analysis phase is by far the dominant contributor to the total runtime, accounting for approximately 65% to 73% of the total execution time. This confirms that, for the domain and window parameters adopted in

this test case, the algorithm remains firmly compute-bound with respect to the DMD computation itself, rather than being limited by inter-process communication.

The data reading phase shows a clear and monotonic decreasing trend with the number of processors, from 190.45 s at 12 ranks down to approximately 71–73 s at 32–36 ranks: this behaviour is consistent with the temporal decomposition strategy described in §4.2.1, since increasing the number of ranks reduces the number of snapshots each process must read from disk in parallel.

Conversely, the MPI redistribution phase exhibits a mild but consistent increasing trend, growing from 28.37 s at 12 ranks to 45.91 s at 36 ranks. This is the expected signature of the compute-bound to communication-bound transition anticipated in §5.3.1: as the number of ranks increases, the spatial subdomain assigned to each process shrinks, and the relative overhead of the All-to-All communication grows with respect to the useful payload exchanged, in accordance with the chunked communication strategy discussed in §4.4.3.

The saving phase remains a minor contributor throughout (below 18 s in all configurations), confirming the effectiveness of the distributed HDF5 I/O strategy described in §4.8.1 in avoiding write bottlenecks even as the number of participating ranks increases.

Overall, the combination of the decreasing reading time and the mildly increasing communication time explains the shape of the total execution time curve reported in Fig. 5.10: the net gain from parallel reading dominates up to approximately 20 ranks, after which the growing relative cost of communication offsets further gains from increased parallelism, producing the observed plateau in the total execution time.

Taken together, the strong scaling analysis and the phase-resolved profiling presented in this section provide a coherent picture of the parallel performance of the developed framework: for the tested configuration ($N_t = 1500$, $Y = 50$, 4 compute nodes), the algorithm scales effectively up to approximately 20 MPI ranks, beyond which the diminishing size of the per-rank workload is increasingly offset by the relative overhead of the All-to-Allv communication, in accordance with the compute-bound to communication-bound transition anticipated by Amdahl's Law in §4.1.2. These results indicate that, for datasets of comparable size, allocating substantially more than 20–24 ranks over the same number of nodes yields no additional benefit, and provide practical guidance for the efficient allocation of computational resources in future applications of the framework to larger DNS datasets.

6 Application to a DNS Separation Dataset

The dataset analysed in this chapter describes the direct numerical simulation of a flat-plate boundary layer subject to an imposed adverse pressure gradient, leading to separation. The streamwise domain considered spans from $x = 200$ to $x = 700$ (in grid units), subdivided into overlapping windows of width 50 points with a step of 4 points between consecutive windows, while a total of 50 wall-normal points were retained per velocity component, and $N_t = 1500$ temporal snapshots were used for each window. For each streamwise window, the Exact DMD algorithm was applied jointly to the three velocity components (U, V, W).

It should be noted that, owing to storage constraints on the computing cluster, the domain was extracted and processed in two separate, non-overlapping campaigns ($x =$ 200-500 and $x =$ 500-700), subsequently combined to obtain the complete reconstruction presented in this section.

6.1 Objective of the Analysis

In the previous chapters, the distributed framework developed in this thesis was validated numerically and characterised in terms of performance, using a reference benchmark case (cylinder wake) of limited size. In this section, the framework is instead applied to a large-scale DNS dataset describing a boundary layer flow with separation, with the aim of assessing whether Dynamic Mode Decomposition, applied incrementally along the streamwise direction following the spatial-DMD formulation discussed in Sec. 2.6, is able to isolate a spectral signature consistent with the transition mechanisms of the separation bubble described in Sec. 2, in particular with the Kelvin-Helmholtz instability.

6.2 Energy Content Along the Streamwise Direction

As a first step, the energy spectrum of the velocity field was computed, obtained via wall-normal integration of the spanwise spectral energy, as a function of the streamwise position x and the spanwise wavenumber k_z. Figure 6.1 shows the result for the three velocity components.

The energy content is found to be negligible for $x \lesssim 450$, growing substantially from $x \approx$ 450-500 onward and reaching its highest values in the range $x \approx$ 500-650. This trend is consistent with the physical expectation of a boundary layer that is still laminar, or only just separated, in the initial part of the analysed domain, followed by a

downstream region in which the coherent structures associated with separation have attained significant energy content.

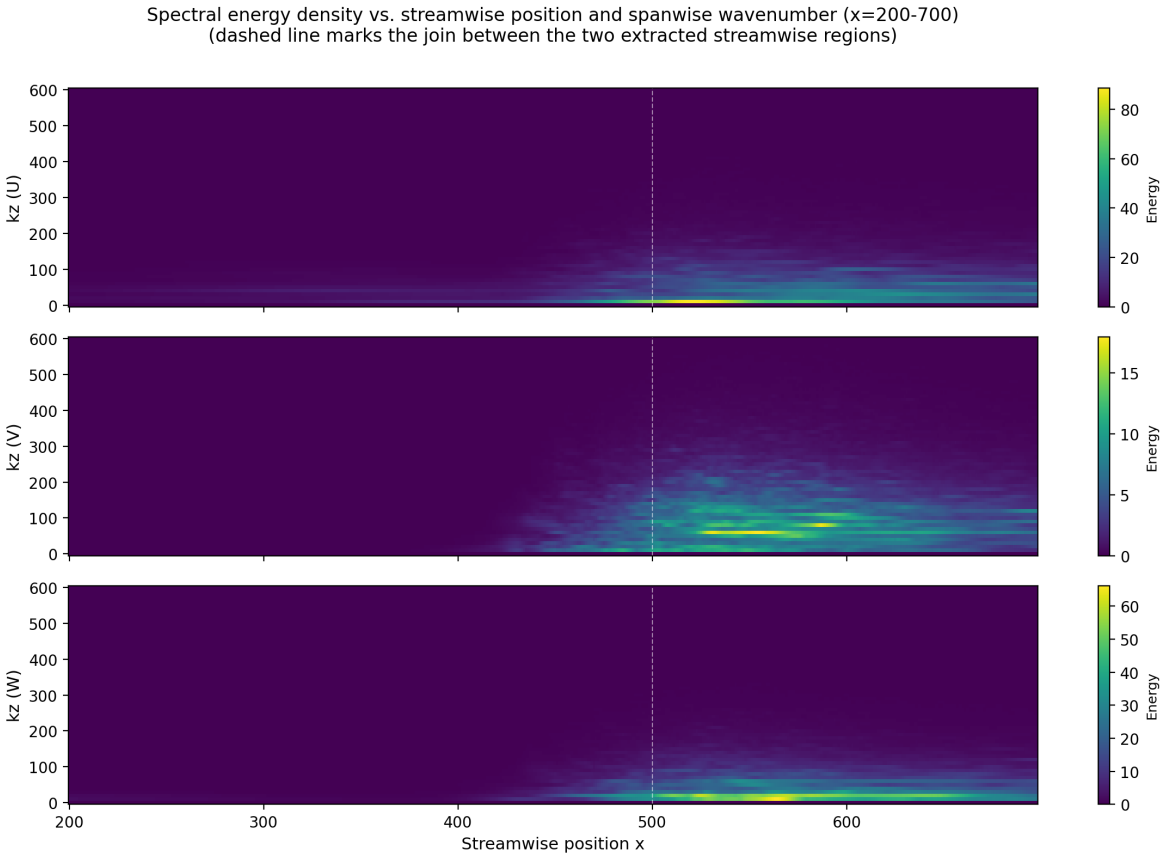


Figure 6.1: Spectral energy density, computed via wall-normal trapezoidal integration, as a function of streamwise position x (x = 200-700) and spanwise wavenumber k_z, for the U, V and W velocity components. The DC ($k_z = 0$) component has been excluded from the colour scale for clarity. Data combine two independently extracted, non-overlapping streamwise regions (x = 200-500 and x = 500-700; the dashed line marks the join).

6.3 Spatial Evolution of the DMD Spectrum

The central result of this section concerns the evolution, along x, of the modulus of the DMD eigenvalues $|\mu|$ associated with the energetically dominant spanwise wavenumber in each window. For each window, both the mean value of $|\mu|$ over the available eigenvalues and its maximum value were computed, the latter being more directly comparable to the approach adopted in the literature for identifying the most unstable Ritz eigenvalue at each domain extent (Sec. 2.6).

Figure 6.2 shows that, for all three velocity components, $|\mu|$ exhibits a well-defined bell-shaped trend: starting from values close to unity (neutral condition) near $x = 225$, it

grows to a maximum of approximately 1.03 (mean value) and 1.07 (maximum value) around $x \approx$ 400-410, then decreases through the neutral threshold $|\mu| = 1$ near $x \approx$ 470-500 and continues to decay to values of approximately 0.97 in the downstream part of the analysed domain ($x \approx$ 650-700).

This trend qualitatively reproduces the growth-saturation-decay pattern predicted by the theory for the Kelvin-Helmholtz instability within the separation bubble (Sec. 2.4): a linear amplification stage of disturbances within the separated shear layer, followed by saturation near the position of maximum bubble displacement, and a subsequent decay associated with the turbulent reattachment of the boundary layer. An aspect that reinforces the physical consistency of this result is the temporal ordering between the two figures: the growth of $|\mu|$ precedes the accumulation of energy observed in Fig. 6.1, an order consistent with the idea that the amplification of disturbances (captured by the eigenvalue) precedes the formation of energetically saturated structures.

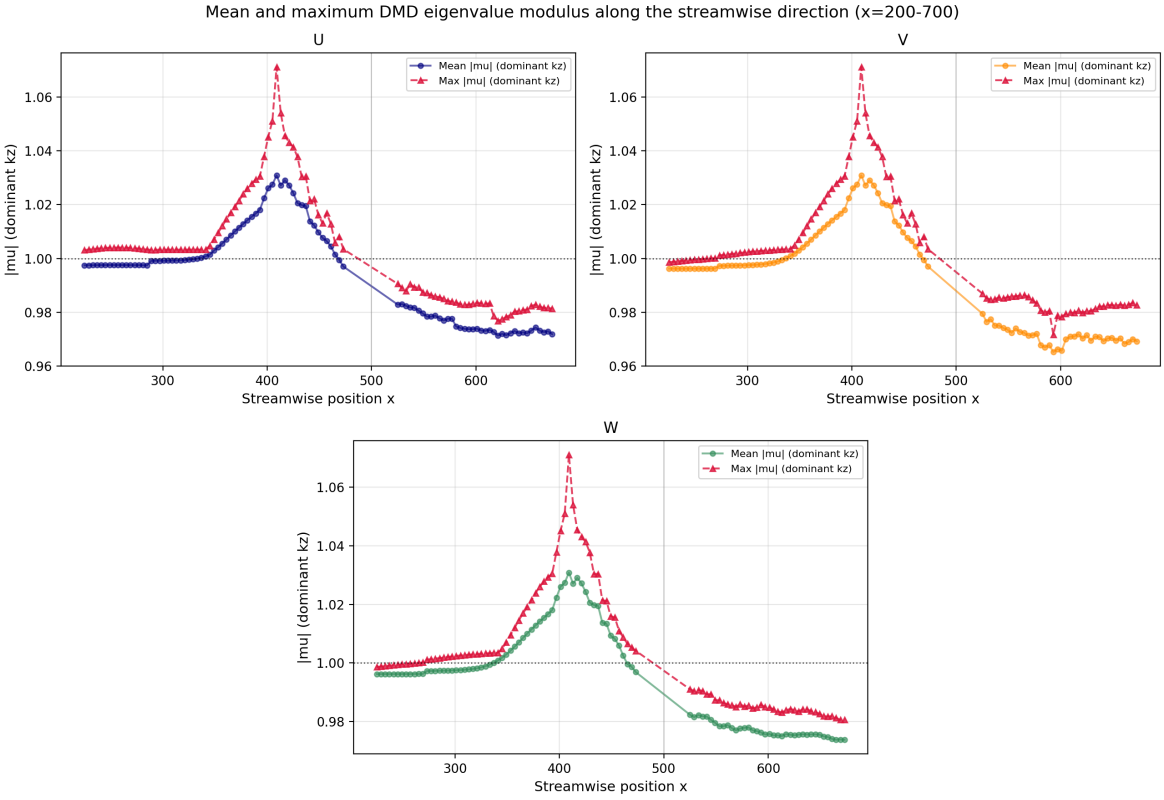


Figure 6.2: Mean and maximum modulus of the DMD eigenvalue spectrum, $|\mu|$, computed for the energetically dominant spanwise wavenumber in each streamwise window, for the U, V and W velocity components, over the full range $x =$ 200-700. The dotted horizontal line marks the neutral-stability threshold $|\mu| = 1$.

It is worth noting that this growth phase ($|\mu| > 1$) did not emerge from an initial analysis restricted to the region $x =$ 500-700 alone: as is evident from Fig. 6.2, this

region lies entirely downstream of the instability peak, already in the decay phase. Extending the analysis upstream ($x = 200$-500) was therefore necessary to capture the growth phase, confirming the importance of a sufficiently wide streamwise coverage when applying incremental DMD to separated flows.

Figure 6.3 shows the discrete DMD eigenvalues on the complex plane for three representative windows, chosen upstream of the growth region ($x \approx 230$), at the instability peak ($x \approx 410$), and well downstream in the decay region ($x \approx 650$). It should be noted that the number of eigenvalues available differs substantially between these windows, since a different SVD truncation rank was used at each streamwise position; the downstream window therefore covers a visibly wider arc of the unit circle simply because more eigenvalues are available there, not because the underlying dynamics are inherently richer. Given the modest amplitude of the modulus variation identified in Fig. 6.2 (at most a few percent around the neutral value), the three point clouds remain visually close to the unit circle at this scale; the figure is therefore best read as evidence of the differing frequency (phase) content captured at each streamwise position, complementing rather than replacing the quantitative modulus comparison of Fig. 6.2.

6.4 Evolution of the Dominant Spanwise Wavenumber

Figure 6.4 shows the evolution of the energetically dominant spanwise wavenumber along x, for the three velocity components. In the growth region ($x \lesssim 450$), the dominant wavenumber is low and essentially constant, indicating a large-scale, nearly two-dimensional structure, consistent with the still two-dimensional nature of the Kelvin-Helmholtz vortices in the early roll-up stage, at low free-stream turbulence levels (Sec. 2.4).

Downstream of the instability peak, the dominant wavenumber increases and becomes more variable, indicating a transition toward smaller-scale structures, consistent with the three-dimensional breakdown mechanism of Kelvin-Helmholtz vortices via secondary elliptic instability, described in the same section.

6.5 Component-Wise Energy Growth and the Onset of Three-Dimensionality

A further piece of physical insight can be obtained by examining the total spanwise-integrated fluctuation energy (summed over all $k_z > 0$) separately for each velocity component, rather than the energy spectrum as a function of k_z shown in Fig. 6.1. Figure 6.5 shows this quantity, on a logarithmic scale, for U, V and W over the full

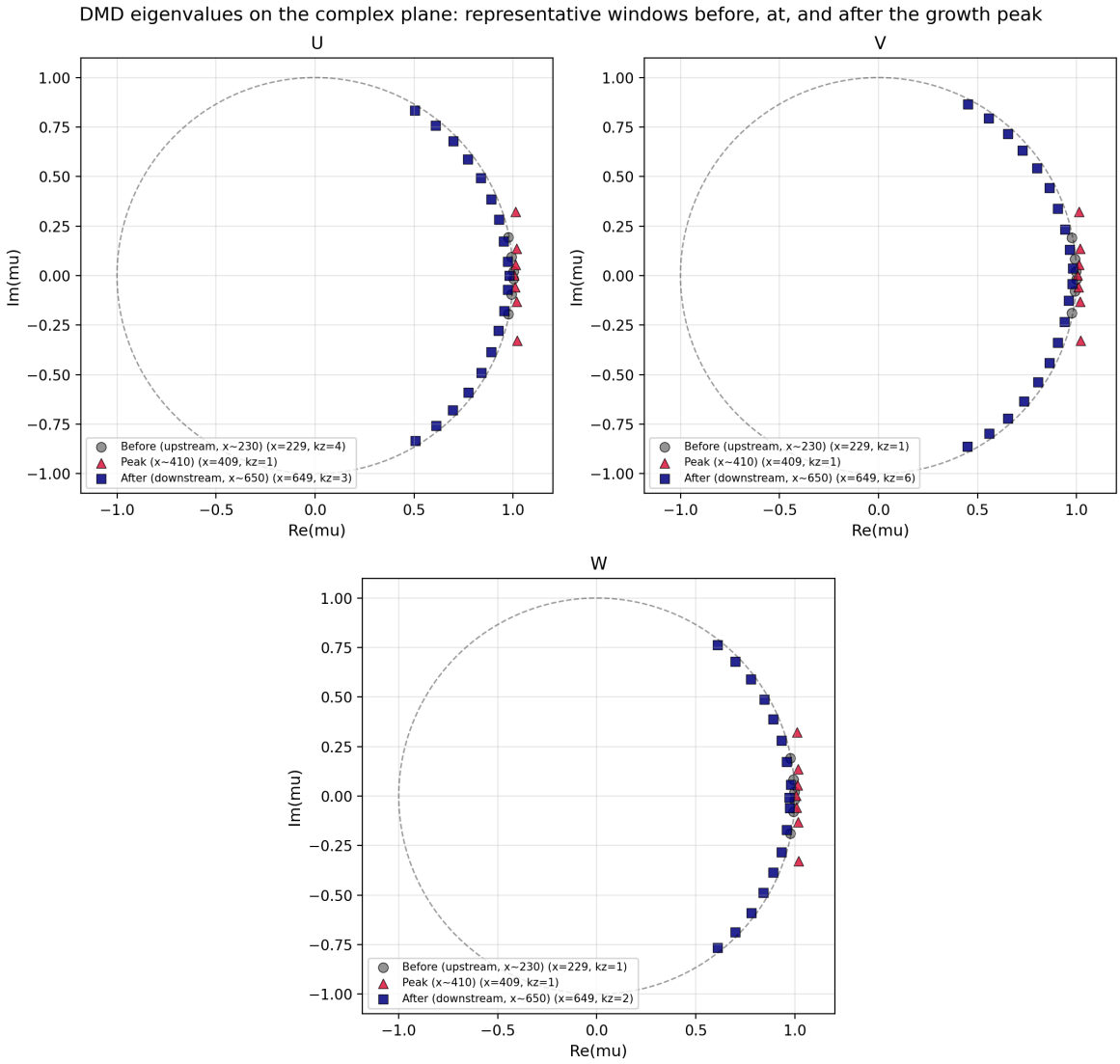


Figure 6.3: Discrete DMD eigenvalues on the complex plane, for the energetically dominant spanwise wavenumber, compared across three representative streamwise windows: upstream of the growth region ($x \approx 230$), at the instability peak ($x \approx 410$), and downstream in the decay region ($x \approx 650$). The dashed line marks the unit circle.

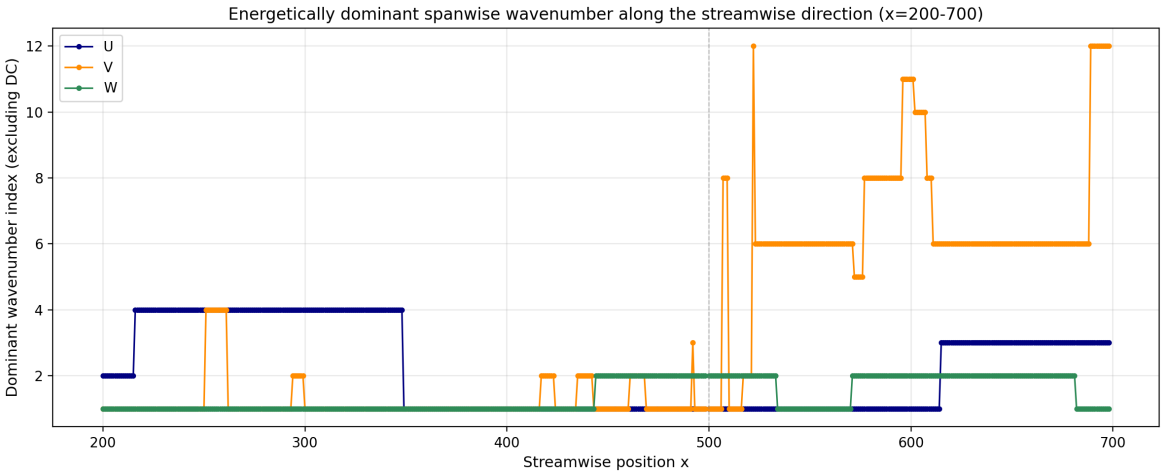


Figure 6.4: Energetically dominant spanwise wavenumber index, identified via the true energy criterion (wall-normal integral of the spectral amplitude), as a function of streamwise position $x = 200$-700, for the U, V and W velocity components. Data combine two independently extracted streamwise regions (no overlap; dashed line marks the join). The V component exhibits a localised excursion to higher wavenumbers around $x = 580$-605, superimposed on a much lower, near-constant baseline wavenumber throughout the upstream growth region.

analysed range.

The three components do not behave identically. The streamwise component U grows monotonically from the very beginning of the analysed domain. The wall-normal component V and the spanwise component W, by contrast, both initially decay, reaching a minimum around $x \approx 315$ and $x \approx 345$ respectively, before growing sharply and eventually overtaking, in relative terms, the growth rate of U. This behaviour is physically consistent with the picture emerging from Sec. 6.4: in the early part of the domain the flow is dominated by an essentially two-dimensional, streamwise-elongated disturbance, for which the wall-normal and spanwise motions are comparatively weak and initially decaying. It is only as the structure approaches the instability peak and begins to acquire a genuinely three-dimensional character that V and, shortly after, W pick up significant energy. The delay between the two, V turning upward before W, is itself consistent with the wall-normal component being more directly coupled to the primary shear-layer roll-up, while the spanwise component only becomes energetic once true spanwise (out-of-plane) distortion of the structure develops.

6.6 Robustness of the Growth-Decay Signature

Since the result presented in Sec. 6.3 relies on selecting, in each window, a single energetically dominant spanwise wavenumber, it is important to verify that the observed

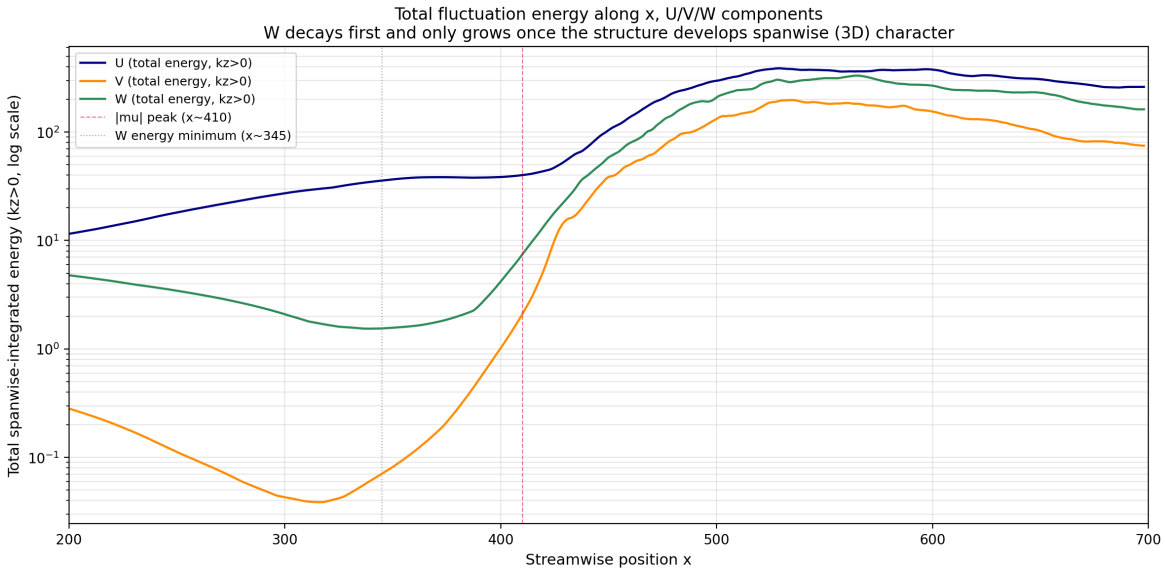


Figure 6.5: Total spanwise-integrated fluctuation energy (summed over $k_z > 0$), on a logarithmic scale, as a function of streamwise position x = 200-700, for the U, V and W velocity components. The dashed vertical line marks the location of the $|\mu|$ growth peak identified in Fig. 6.2; the dotted line marks the energy minimum of the W component.

growth-decay pattern is not an artefact of this particular selection criterion. Figure 6.6 shows the mean eigenvalue modulus $|\mu|$ along x computed independently for several fixed spanwise wavenumbers ($k_z = 1, 2, 3, 4, 6, 8$), for the U component.

The curves obtained for essentially all of these wavenumbers collapse onto the same bell-shaped trend in the growth region ($x \approx$ 225-450), confirming that the growth-saturation signature is a broadband phenomenon involving nearly all spanwise scales simultaneously, rather than being specific to the single dominant wavenumber tracked in Fig. 6.2. The curves only begin to separate in the downstream decay tail ($x \gtrsim 500$), where higher wavenumbers are found to decay faster than lower ones, a trend consistent with stronger viscous dissipation acting on smaller spanwise scales.

6.7 A Secondary Local Feature: Spanwise Wavenumber Excursion in V

Besides the broad growth-decay trend discussed above, a more localised feature is worth reporting. As already noted in Sec. 6.4 and visible in Fig. 6.4, the dominant spanwise wavenumber of the V component undergoes a brief excursion from $k_z = 6$ to $k_z = 8$ and back to $k_z = 6$ within the narrow window $x \approx$ 580-605, well inside the downstream decay region. Figure 6.7 compares the discrete DMD eigenvalues, on the complex plane,

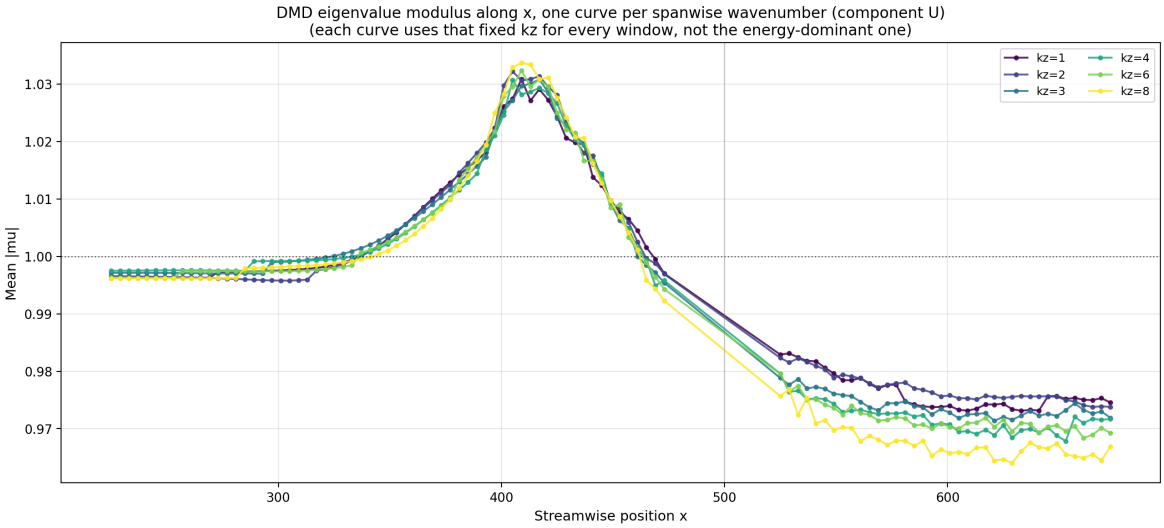


Figure 6.6: Mean modulus of the DMD eigenvalue spectrum along x, computed independently for several fixed spanwise wavenumbers ($k_z = 1, 2, 3, 4, 6, 8$), for the U velocity component. Unlike Fig. 6.2, each curve here uses the same fixed k_z throughout, rather than the energy-dominant one for each window.

for three windows respectively upstream of, within, and downstream of this excursion. The eigenvalues associated with the $k_z = 8$ window lie systematically closer to the interior of the unit circle than those of the neighbouring $k_z = 6$ windows, indicating a locally more strongly damped structure. This localised feature is presented here only as a secondary observation superimposed on the main decay trend of Sec. 6.3, rather than as an independent transition event; no attempt has been made in this preliminary analysis to establish its physical origin with certainty.

6.8 Limitations of the Analysis

Some limitations of the analysis presented in this section should be explicitly stated. First, the exact separation location of the bubble and the associated momentum thickness θ were not determined in this work; it was therefore not possible to normalise the observed characteristic wavelengths according to the criterion adopted in the literature (Sec. 2.5), nor to quantitatively verify the possible collapse of the normalised values onto the known benchmark values ($\lambda_z/\theta \approx 22$, $\lambda_x/\theta \approx 35$). The comparison with the literature presented in this section therefore remains qualitative.

Second, the criterion used to select the dominant wavenumber relies on the energy spectrum computed independently of the DMD. It was verified that the raw DMD amplitude does not constitute a reliable proxy for the physical energy content, particularly at high wavenumbers, where the eigenvector matrix becomes ill-conditioned. The

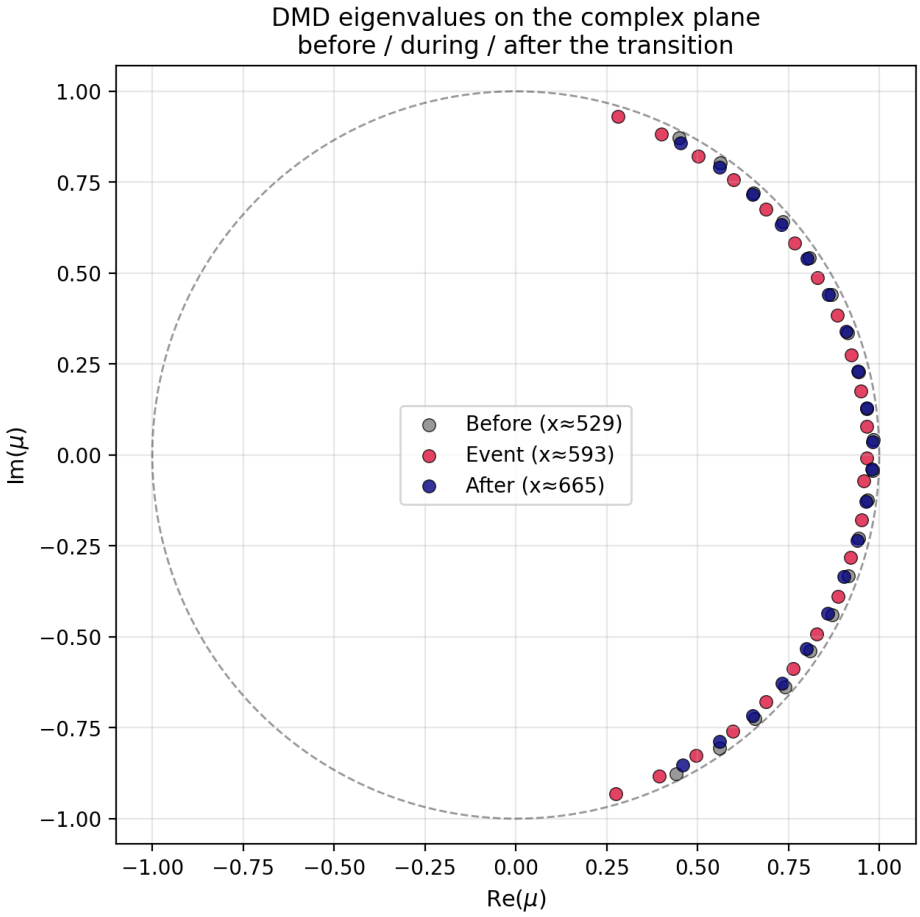


Figure 6.7: Discrete DMD eigenvalues on the complex plane for the dominant spanwise wavenumber, compared across three streamwise windows bracketing the localised excursion of the V component: before ($x \approx 529$), during ($x \approx 593$, $k_z = 8$), and after ($x \approx 665$). Eigenvalues associated with the intermediate window lie systematically closer to the interior of the unit circle.

true energy spectrum was therefore chosen for identifying the dominant scale in each window, a choice that should be explicitly stated as a methodological decision rather than an inherent property of the DMD output.

6.9 Summary of Findings

Taken together, the results presented in this section show that the incremental, spatial-DMD framework developed in this thesis is able to isolate, from a large-scale DNS dataset, a spectral signature consistent with the classical growth-saturation-decay picture of the Kelvin-Helmholtz instability within a separation bubble. This signature consists of a broadband amplification of the DMD eigenvalue modulus across nearly all spanwise wavenumbers, peaking around $x \approx$ 400-410. A decay associated with the downstream reattachment region follows. This signature is corroborated by an independent quantity, the total fluctuation energy, which grows with a compatible streamwise ordering. This quantity further reveals that the streamwise component energises first, while the wall-normal and spanwise components only become significant once the structure develops a genuinely three-dimensional character close to the instability peak. The dominant spanwise wavenumber itself evolves from a low, nearly two-dimensional value upstream to a higher, more variable one downstream, consistent with the expected three-dimensional breakdown of the Kelvin-Helmholtz rollers. Taken together, these independent diagnostics, eigenvalue modulus, integrated energy, and dominant wavenumber, offer mutually consistent evidence for the same underlying physical transition.

7 Conclusions

As discussed in Chapter 2, the coexistence of streaky structures and the Kelvin-Helmholtz instability within the separated shear layer plays a central role in the transition process, and Dynamic Mode Decomposition has proven, in prior work, to be an effective tool for isolating these structures from experimental and numerical data alike. However, the scale of DNS datasets, comprising thousands of three-dimensional snapshots, exceeds the memory capacity of a single compute node, making a distributed computational approach indispensable.

To this end, a parallel framework based on the Message Passing Interface (MPI) was developed. The framework implements a hybrid decomposition strategy: an initial temporal partitioning of the dataset, enabling efficient parallel I/O and in-place spectral decomposition along the spanwise direction, followed by an All-to-Allv redistribution onto a spatial partitioning, which allows the Exact DMD algorithm to be applied independently by each MPI rank to its assigned streamwise subdomain and spectral wavenumber, as detailed in Chapter 4.

The correctness and reliability of the implementation were established through a comprehensive validation campaign, presented in Chapter 5. The All-to-Allv data redistribution was shown to introduce no measurable numerical perturbation, with the discrete DMD spectra obtained from the serial and parallel execution paths matching within numerical precision. The Exact DMD algorithm itself was independently validated against a reference MATLAB implementation on the classical cylinder wake benchmark, yielding agreement in both eigenvalues and spatial modes to within double-precision tolerance. Finally, a strong scaling analysis conducted on the Leonardo supercomputer at CINECA showed that the framework scales effectively up to approximately 20 MPI ranks, beyond which the relative overhead of inter-process communication progressively offsets the benefits of increased parallelism, consistently with the compute-bound to communication-bound transition predicted by Amdahl's Law.

Beyond this methodological validation, Chapter 6 presented a first application of the framework to a genuine large-scale DNS dataset describing a separated boundary layer, extending over a streamwise domain of nontrivial extent (x = 200-700, in grid units). This preliminary analysis showed that the modulus of the dominant DMD eigenvalue follows a well-defined growth-saturation-decay trend along the streamwise direction, qualitatively consistent with the classical picture of Kelvin-Helmholtz amplification, saturation, and subsequent decay associated with turbulent reattachment. This trend was found to be broadband in nature, involving nearly all analysed spanwise wavenum-

bers simultaneously rather than being an artefact of the specific wavenumber-selection criterion adopted, and was independently corroborated by the streamwise evolution of the integrated fluctuation energy, which revealed a physically consistent ordering between velocity components. The streamwise component energises first, while the wall-normal and spanwise components initially decay before growing sharply as the structure acquires a genuinely three-dimensional character near the instability peak. The evolution of the dominant spanwise wavenumber, from a low and nearly two-dimensional value upstream to a higher and more variable one downstream, further supports this interpretation. A secondary, spatially localised feature, a brief excursion of the dominant wavenumber of the wall-normal component, was also identified within the decay region and reported as a subsidiary observation, without attempting to establish its physical origin with the same degree of confidence afforded to the main trend. As explicitly discussed in Sec. 6.8, this application remains preliminary in scope: the comparison with literature benchmarks is qualitative rather than quantitative, owing to the lack of a normalisation based on the local momentum thickness, and the wavenumber-selection criterion itself was based on the true spectral energy rather than on the raw DMD amplitude, a methodological choice motivated by the observed ill-conditioning of the eigenvector matrix at high wavenumbers.

Taken together, these results demonstrate that the developed framework constitutes not only a validated and scalable computational tool, but also one capable of yielding physically interpretable results when applied to a real, large-scale separated flow dataset. The combination of rigorous numerical validation, characterised parallel performance, and a first physically consistent application to DNS data addresses the full scope originally set out for this thesis: overcoming the memory limitations that would otherwise make such an analysis intractable with a serial approach, while demonstrating that the resulting distributed pipeline produces results consistent with established physical expectations.

Limitations and Future Work

Some aspects of the present work remain open to further investigation. An initial run of the scalability analysis presented in §5.3.1 had revealed an anomalous execution time in the FFT phase at 18 and 36 MPI ranks, nearly twice that of neighbouring configurations. The 18-rank configuration was subsequently re-executed under otherwise identical conditions: the FFT phase returned to a value fully consistent with the neighbouring configurations (76.80 s, against the anomalous 168.86 s originally recorded), while all other phases remained within the expected range, confirming that the origi-

nal anomaly was attributable to transient contention on the shared HPC nodes rather than to an internal load imbalance, and that the corrected value has been adopted throughout this thesis.

Regarding the preliminary physical application presented in Chapter 6, several directions remain open. A quantitative, rather than qualitative, comparison with the literature would require determining the exact separation location and the associated local momentum thickness, so as to normalise the observed characteristic wavelengths and directly compare them against established benchmark values. The robustness of the identified growth-saturation-decay signature was assessed only against the wavenumber-selection criterion and within a single realisation of the dataset; repeating the analysis on an independently generated dataset, or on a different portion of the same simulation, would provide a stronger test of its generality. A direct visualisation of the spatial structure of the dominant mode, complementing the eigenvalue- and energy-based diagnostics presented in this thesis, would also help to connect the observed spectral signature more concretely to the classical streak and Kelvin-Helmholtz roll-up structures discussed in the literature review, and constitutes a natural next step building directly on the framework and the dataset already available. More broadly, extending the analysed domain further upstream and downstream of the present range, and applying the same diagnostics to the remaining wavenumbers not examined in detail here, would allow a more complete characterisation of the transition process across the whole separation bubble.

Finally, on the computational side, further extensions could include a weak scaling analysis, to assess the framework's behaviour as both the problem size and the number of compute nodes are increased proportionally. Further optimisation of the communication pattern underlying the All-to-Allv redistribution could also be pursued for configurations beyond the saturation point identified in this work.

A Superseded Scalability Result

For completeness and transparency, this appendix reports the original scalability figures obtained before the 18-rank configuration was re-executed (see the discussion in Sec. 5.3.1 and in the Limitations and Future Work section of Chapter 7). The anomalous FFT timing at 18 ranks, visible in both Fig. A.1 and Fig. A.2, was subsequently found to be non-reproducible and attributable to transient contention on the shared HPC nodes rather than to an intrinsic property of the algorithm; the corrected values are used throughout the main text.

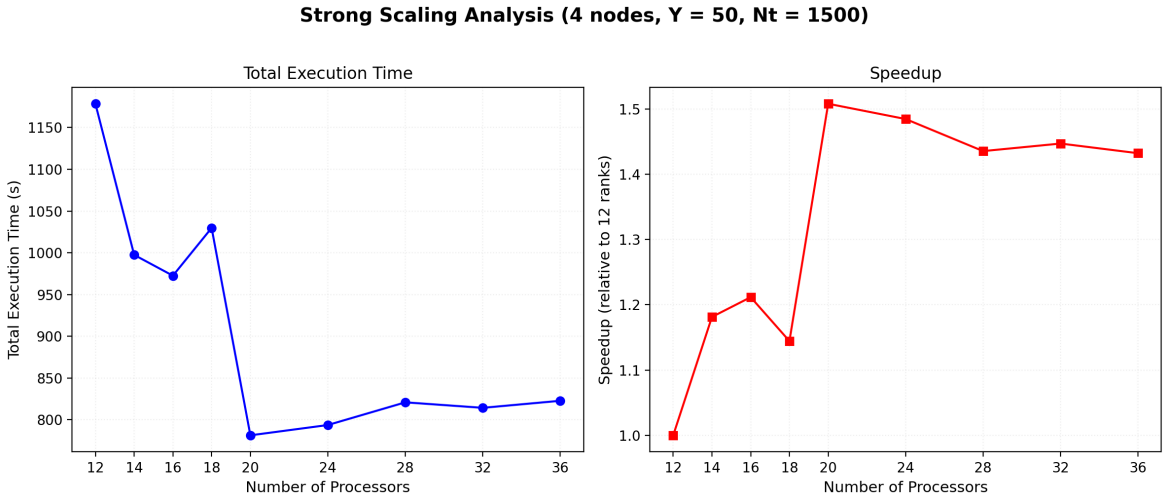


Figure A.1: Original (superseded) total execution time and speedup, including the anomalous data point at 18 ranks later found to be non-reproducible. Retained here for transparency; superseded by the corrected figure in Sec. 5.3.1.

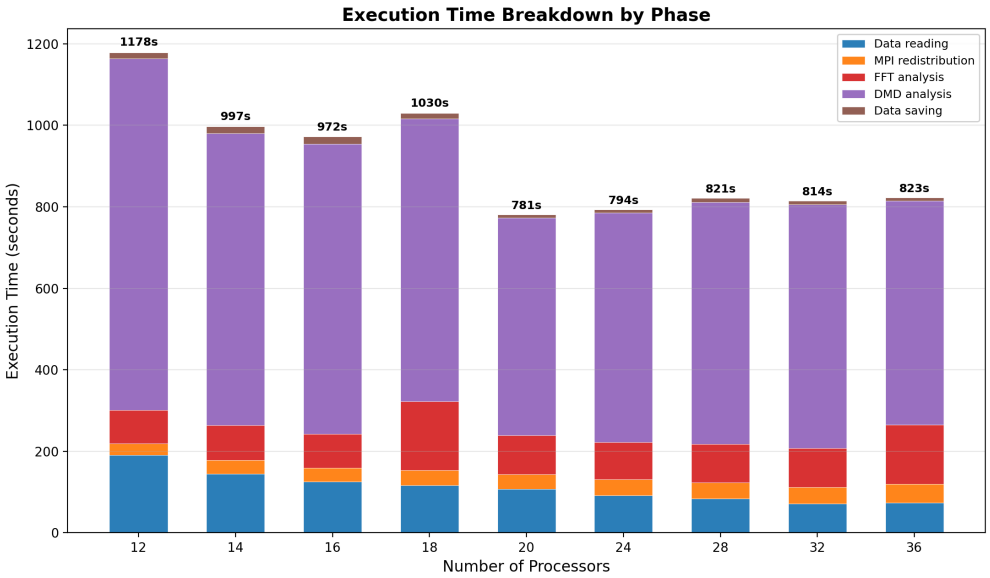


Figure A.2: Original (superseded) execution time breakdown by phase, including the anomalous FFT timing at 18 ranks later found to be non-reproducible. Retained here for transparency; superseded by the corrected figure in Sec. 5.3.1.

References

[1] D. C. Wisler. *The Technical and Economic Relevance of Understanding Boundary Layer Transition in Gas Turbine Engines*. Tech. rep. NASA/CP-1998-206958. Minnowbrook II, 1997 Workshop on Boundary Layer Transition in Turbomachines, J. E. LaGraff and D. E. Ashpis, eds., pp. 53–64. NASA, 1998.

[2] E. Dick and S. Kubacki. "Transition Models for Turbomachinery Boundary Layer Flows: A Review". In: *International Journal of Turbomachinery, Propulsion and Power* 2.2 (2017), p. 4. DOI: 10.3390/ijtpp2020004.

[3] J. D. Coull. "Endwall Loss in Turbine Cascades". In: *Journal of Turbomachinery* 139.8 (2017), p. 081004. DOI: 10.1115/1.4035851.

[4] J. Hourmouziadis. *Aerodynamic Design of Low-Pressure Turbines*. AGARD Lecture Series 167. AGARD, 1989.

[5] E. M. Curtis et al. "Development of Blade Profiles for Low-Pressure Turbine Applications". In: *Journal of Turbomachinery* 119.3 (1997), pp. 531–538. DOI: 10.1115/1.2841154.

[6] R. J. Howell et al. "High Lift and Aft-Loaded Profiles for Low-Pressure Turbines". In: *Journal of Turbomachinery* 123.2 (2001), pp. 181–188. DOI: 10.1115/1.1350409.

[7] R. D. Sandberg and V. Michelassi. "The Current State of High-Fidelity Simulations for Main Gas Path Turbomachinery Components and Their Industrial Impact". In: *Flow, Turbulence and Combustion* 102 (2019), pp. 797–848. DOI: 10.1007/s10494-019-00013-3.

[8] R. Pichler et al. "LES and RANS Analysis of the End-Wall Flow in a Linear LPT Cascade: Part I — Flow and Secondary Vorticity Fields Under Varying Inlet Condition". In: *Proceedings of ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition*. GT2018-76233. 2018. DOI: 10.1115/GT2018-76233.

[9] J. L. Lumley. *Stochastic Tools in Turbulence*. Vol. 12. Applied Mathematics and Mechanics. New York: Academic Press, 1970.

[10] L. Sirovich. "Turbulence and the Dynamics of Coherent Structures. Part I–III". In: *Quarterly of Applied Mathematics* 45 (1987), pp. 561–590.

[11] Peter J. Schmid. "Dynamic mode decomposition of numerical and experimental data". In: *Journal of Fluid Mechanics* 656 (2010), pp. 5–28. DOI: 10.1017/S0022112010001217.

[12] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard, Version 3.1*. Tech. rep. University of Tennessee, 2015.

[13] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. 3rd. MIT Press, 2014.

[14] L. Prandtl. "Verhandlungen des dritten internationalen Mathematiker-Kongresses". In: Heidelberg, Leipzig, 1904, pp. 484–491.

[15] H. Schlichting. *Boundary Layer Theory*. New York: McGraw Hill, 1979.

[16] R. E. Mayle. "The 1991 IGTI Scholar Lecture: The Role of Laminar-Turbulent Transition in Gas Turbine Engines". In: *Journal of Turbomachinery* 113.4 (1991), pp. 509–536.

[17] P. S. Klebanoff, K. D. Tidstrom, and L. M. Sargent. "The Three-Dimensional Nature of Boundary-Layer Instability". In: *Journal of Fluid Mechanics* 12.1 (1962), pp. 1–34.

[18] H. W. Emmons. "The Laminar-Turbulent Transition in a Boundary Layer – Part I". In: *Journal of Aeronautical Sciences* 18 (1951), pp. 490–498.

[19] A. Hatman and T. Wang. "Separated-Flow Transition Part 1 – Experimental Methodology and Mode Classification". In: *ASME Turbo Expo*. Stockholm, Sweden, 1998.

[20] R. G. Jacobs and P. A. Durbin. "Simulations of Bypass Transition". In: *Journal of Fluid Mechanics* 428 (2001), pp. 185–212.

[21] M. Gaster. *The Structure and Behaviour of Laminar Separation Bubbles*. Tech. rep. Technical Report No. 3595. Aeronautical Research Council, 1967.

[22] S. Hosseinverdi and H. F. Fasel. "Numerical Investigation of Laminar-Turbulent Transition in Laminar Separation Bubbles: The Effect of Free-Stream Turbulence". In: *Journal of Fluid Mechanics* 858 (2019), pp. 714–759.

[23] S. S. Diwan and O. N. Ramesh. "On the Origin of the Inflectional Instability of a Laminar Separation Bubble". In: *Journal of Fluid Mechanics* 629 (2009), pp. 263–298.

[24] O. Marxen and D. S. Henningson. "The Effect of Small-Amplitude Convective Disturbances on the Size and Bursting of a Laminar Separation Bubble". In: *Journal of Fluid Mechanics* 671 (2011), pp. 1–33.

[25] D. Simoni et al. "Inspection of the Dynamic Properties of Laminar Separation Bubbles: Free-Stream Turbulence Intensity Effects for Different Reynolds Numbers". In: *Experiments in Fluids* 58 (2017), p. 66.

[26] A. Dotto et al. "Dynamic Mode Decomposition and Koopman Spectral Analysis of Boundary Layer Separation-Induced Transition". In: *Physics of Fluids* 33 (2021), p. 104104.

[27] J. Verdoya et al. "Inspection of Structures Interaction in Laminar Separation Bubbles with Extended Proper Orthogonal Decomposition Applied to Multi-Plane Particle Image Velocimetry Data". In: *Physics of Fluids* 33 (2021), p. 043607.

[28] M. S. Istvan and S. Yarusevych. "Effects of Free-Stream Turbulence Intensity on Transition in a Laminar Separation Bubble Formed over an Airfoil". In: *Experiments in Fluids* 59 (2018), p. 52.

[29] B. Abu-Ghannam and R. Shaw. "Natural Transition of Boundary Layers – The Effects of Turbulence, Pressure Gradient, and Flow History". In: *Journal of Mechanical Engineering Science* 22.5 (1980), pp. 213–228.

[30] B. Thwaites. "Approximate Calculation of the Laminar Boundary Layer". In: *The Aeronautical Quarterly* 1.3 (1949), pp. 245–280.

[31] A. Alessandri et al. "Dynamic Mode Decomposition for the Inspection of Three-Regime Separated Transitional Boundary Layers Using a Least Squares Method". In: *Physics of Fluids* 31 (2019), p. 044103.

[32] Jonathan H. Tu et al. "On dynamic mode decomposition: Theory and applications". In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421. DOI: 10.3934/jcd.2014.1.391.

[33] Bernard O. Koopman. "Hamiltonian systems and transformation in Hilbert space". In: *Proceedings of the National Academy of Sciences* 17.5 (1931), pp. 315–318. DOI: 10.1073/pnas.17.5.315.

[34] Clarence W. Rowley et al. "Spectral analysis of nonlinear flows". In: *Journal of Fluid Mechanics* 641 (2009), pp. 115–127. DOI: 10.1017/S0022112009992059.

[35] Igor Mezić. "Spectral properties of dynamical systems, model reduction and decompositions". In: *Nonlinear Dynamics* 41.1–3 (2005), pp. 309–325. DOI: 10.1007/s11071-005-2824-x.

[36] J. Nathan Kutz et al. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016. DOI: 10.1137/1.9781611974508.

[37] James W. Cooley and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of Computation* 19.90 (1965), pp. 297–301. DOI: 10.1090/S0025-5718-1965-0178586-1.

[38] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. 3rd. Prentice Hall, 2010.

[39] Gene M. Amdahl. "Validity of the single processor approach to achieving large scale computing capabilities". In: *Proceedings of the April 18–20, 1967, Spring Joint Computer Conference (AFIPS '67)*. ACM, 1967, pp. 483–485. DOI: 10.1145/1465482.1465560.

[40] John L. Gustafson. "Reevaluating Amdahl's law". In: *Communications of the ACM* 31.5 (1988), pp. 532–533. DOI: 10.1145/42411.42415.