Technical Document and User Guide

Version 1.0.0

Francis Lacombe, University of Waterloo

March 18, 2022

Contents

List of Code Listings

1 Introduction

`Krypton` is a python-based, open-source framework to solve the linear and non-linear Parabolized Stability Equations (PSE) on a curvilinear coordinate system as a predictive tool to estimate the laminar-to-turbulent transition at transonic conditions. It can solve the linear and nonlinear version of the PSE (LPSE and NPSE). The following system of equations are solved:

LST

$$\text{System solved :}(\mathbf{L} + \mathbf{N}D + \mathbf{P}D^2)\phi = 0$$

LPSE

$$\text{System solved :}(\mathbf{L} + \mathbf{N}D + \mathbf{P}D^2)\phi = -\mathbf{M}\frac{d\phi}{dx}$$

NPSE

$$\text{System solved :}(\mathbf{L} + \mathbf{N}D + \mathbf{P}D^2)\phi = -\mathbf{M}\frac{d\phi}{dx} + NLT$$

where NLT represents the non-linear modal interaction which acts as a forcing term to the equation set.

Written in Python and leveraging well-established libraries, the framework includes a laminar flow solver using a consistent numerical scheme as the modal stability calculations. The code is validated against published cases and can serve as the basis for future development of modal stability based problems in aerospace engineering, geophysical, and multiphase flows.

2 Installation

2.1 Dependencies

- Latest python version (3.6.4 or higher), I recommend installing the Anaconda Distribution (way faster). Link : `https://anaconda.org/anaconda/python`
- Python Packages: `Numpy, Scipy, Matplotlib, hickle, pypardiso, pyevtk, math`

To install hickle: Make sure to run `setup.py` with python3.7 or above (Anaconda3 distribution)

```
>>  git clone https://github.com/telegraphic/hickle.git
>>  cd hickle
>>  python setup.py -install
```

2.2 Installation

Once all the dependencies are met, the code can be cloned from the repository:

```
>> cd [INSTALLATION_FOLDER]
>> git clone https://git.uwaterloo.ca/flacombe/krypton-softwarex.git
```

3 Quick start guide

The file `main.py` is the first script you should check. This is where you will set the problem parameters (Ma, Re, Pr, wall temperature, pressure gradient, etc.), this is also where you set the discretization, BDF order. This is basically the only file you need to edit if you want to run a simulation. At some point,

```
>> python main.py
```

you might have to develop the code. The stability equations are located in the `PSE.py` script.

The code is object oriented and designed so it is easy to add new features and change the equations solved. If you have any suggestion to improve the efficiency or make the code clearer, I am open to suggestion.

4 Mathematical background

The stability equations are developed from the Compressible Navier-Stokes equations (CNSE) in their non-conservative form. The non-conservative formulation is selected as the focus is primarily on subsonic flows and, more importantly, because of the convenience handling the primitive thermodynamic variables, such as pressure, without the need for post-treatment; this is particularly helpful to study bypass transition induced by free-stream noise. The modified conservation of mass, momentum and energy, in non-conservative form, read:

$$-pT\frac{\partial u_i}{\partial x_i} = T\left[\frac{\partial p}{\partial t} + u_i\frac{\partial p}{\partial x_i}\right] - p\left[\frac{\partial T}{\partial t} + u_i\frac{\partial T}{\partial x_i}\right] \tag{1}$$

$$\begin{aligned}&\rho\left[\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right]\\ &= -\frac{\partial p}{\partial x_i} + \frac{1}{Re_\delta}\frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}\right]\end{aligned} \tag{2}$$

$$\begin{aligned}&\rho c_p\left[\frac{\partial T}{\partial t} + u_j\frac{\partial T}{\partial x_j}\right]\\ &= \frac{1}{Re_\delta Pr}\frac{\partial}{\partial x_i}\left(\lambda\frac{\partial T}{\partial x_i}\right)\\ &+ (\gamma-1)Ma^2\left(u_i\frac{\partial p}{\partial x_i}\right)\\ &+ \frac{(\gamma-1)Ma^2}{Re}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}\right]\frac{\partial u_i}{\partial x_j}\end{aligned} \tag{3}$$

4.1 Dimensionless variables

The use of dimensionless variables can greatly simplify any mathematical problem to a minimum number of physically meaningful key-variables. The equations (1)-(3) were nondimensionalized using a classical approach[1] with the following set of non-dimensional variables:

$$\begin{aligned} x_i &= \frac{x_i^*}{\delta_0} & u_i &= \frac{u_i^*}{u_0} & T &= \frac{T^*}{T_0} & p &= \frac{p^*}{p_s}\\ \mu &= \frac{\mu^*}{\mu_0} & \rho &= \frac{\rho^*}{\rho_0} & c_p &= \frac{c_p^*}{c_{p0}} & \lambda &= \frac{\lambda^*}{\lambda_0} \end{aligned} \tag{4}$$

Where $\delta_0 = \sqrt{\frac{x_0 \nu_0}{u_0}}$ is the length scale proportional to the inflow boundary-layer thickness and $p_s = \rho_0 u_0^2$ is the pressure scale. The dimensional quantities are defined with the asterisk (*). The subscript 0 stands for the reference quantities, typically the freestream conditions. The reference thermodynamic conditions must ensure that the equation of state is satisfied

$$p_0 = \rho_0 R_0 T_0 \tag{5}$$

The reference Reynolds and Mach numbers are defined as

$$Re_\delta = \sqrt{\frac{u_0 x_0}{\nu_0}} \qquad\qquad Ma = \frac{u_0}{\sqrt{\gamma R_0 T_0}} \tag{6}$$

In practice, the following reference quantities are set to

Ma = Value	Re_δ = Value	Pr = Value
ρ_0 = 1.	u_0 = 1.	T_0 = 1.

The other reference quantities are deduced from

$$\nu_0 = \frac{u_0 x_0}{Re_\delta^2} \qquad\qquad p_s = \rho_0 u_0^2 \tag{7}$$

4.2 Local stability analysis

The modal stability theory is a branch of fluid mechanics that has been developed to study the evolution of small amplitude perturbations within a variety of flows. It relies on the decomposition of the flow quantities into a steady part $\overline{q}$ and an unsteady part q' [2].

$$q(\vec{x}, t) = \overline{q}(\vec{x}) + q'(\vec{x}, t) \tag{8}$$

Where $\vec{x}$ is the spatial coordinate vector and t is time. Substituting equation (8) into the compressible Navier-Stokes equations and neglecting the cross-product of fluctuating parts (non-linear terms) leads to the *Linearized Compressible Navier-Stokes Equations*:

$$\frac{\partial u_i'}{\partial x_i} + \left(\frac{T'}{\overline{T}} + \frac{p'}{\overline{p}}\right)\frac{\partial \overline{u}_i}{\partial x_i} = \frac{p'}{\overline{p}\,\overline{T}}\frac{D\overline{T}}{Dt} + \frac{1}{\overline{T}}\left[\frac{DT'}{Dt} - u_i'\frac{\partial \overline{T}}{\partial x_i}\right] - \frac{T'}{\overline{p}\,\overline{T}}\frac{D\overline{p}}{Dt} - \frac{1}{\overline{p}}\left[\frac{Dp'}{Dt} + u_i'\frac{\partial \overline{p}}{\partial x_i}\right] \tag{9}$$

$$\left(\frac{\overline{\rho}}{\overline{p}}p' - \frac{\overline{\rho}}{\overline{T}}T'\right)\frac{D\overline{u}_i}{Dt} + \overline{\rho}\left(\frac{Du_i'}{Dt} + u_j'\frac{\partial \overline{u}_i}{\partial x_j}\right) = -\frac{\partial p'}{\partial x_i} + \frac{1}{Re}\frac{\partial \sigma_{ij}'}{\partial x_j} \tag{10}$$

$$\left(\frac{\overline{\rho}c_p}{\overline{p}}p' - \frac{\overline{\rho}c_p}{\overline{T}}T'\right)\frac{D\overline{T}}{Dt} + \overline{\rho}c_p\left(\frac{DT'}{Dt} + u_j'\frac{\partial\overline{T}}{\partial x_j}\right) = \frac{1}{RePr}\frac{\partial}{\partial x_i}\left(\overline{\lambda}\frac{\partial T'}{\partial x_i} + \lambda'\frac{\partial\overline{T}}{\partial x_i}\right) + (\gamma-1)Ma^2\left(\overline{u}_i\frac{\partial p'}{\partial x_i} + u_i'\frac{\partial\overline{p}}{\partial x_i}\right) + \frac{(\gamma-1)Ma^2}{Re}\phi_{ij}' \tag{11}$$

In the modal theory, the stability is mode-dependant and must therefore be computed for each mode independently, hence the name *modal*. However, the stability of the first few modes is usually a good indicator of the overall stability of the flow. In this regard, the perturbation vector in equation (8) is usually assumed of the following form:

$$q'(\vec{x},t) = \hat{q}(\vec{x})\chi(\vec{x},t) = \hat{q}(\vec{x})e^{i\theta} \tag{12}$$

Where $\hat{q}$ and χ are the amplitude and phase functions, respectively. This constitutes the starting point of all modal stability theories. From there, different simplifications can be made on the amplitude and/or phase functions which lead to different branches of modal stability theory (*local*, *nonlocal*, *global*). The next section will outline the main steps of the *local* stability theory.

The local stability theory relies on the ansatz that the perturbation vector q' grows (or decays) following a quasi-static process. In such case, the stability analysis is carried at a specific *local* position and does not account for the evolution of the perturbations along their path [2]. Therefore, the amplitude function varies in the normal direction only. The spatial and temporal wavenumbers can then be taken as constants, which leads to the following form for the perturbation vector:

$$q' = \hat{q}(y)\exp(i(\alpha x + \beta z - \omega t)) \tag{13}$$

Introducing the equation (13) into the equations (9)-(11) leads to the following system of equations.

$$\mathbf{L}\vec{\phi} + \mathbf{S}\frac{d\vec{\phi}}{dy} + \mathbf{T}\frac{d^2\vec{\phi}}{dy^2} = \vec{0} \tag{14}$$

Where $\vec{\phi} = [\hat{u},\hat{v},\hat{w},\hat{p},\hat{T}]^T$ and the matrices **L**, **S** and **T** depend on the mean flow quantities and wavenumbers (α,β,ω) only. With the following boundary conditions

$$\hat{u}(0) = \hat{v}(0) = \hat{w}(0) = \hat{T}(0) = \hat{p}(0) - 1 = 0 \tag{15}$$

$$\hat{u}_y(\infty) = \hat{v}_y(\infty) = \hat{w}_y(\infty) = \hat{p}_y(\infty) = \hat{T}_y(\infty) = 0 \tag{16}$$

The boundary condition on $\hat{p}$ has two purposes: (1) avoiding the trivial solution; and (2) giving a scaling to the solution. Whereas linear-stability-based models (all models derived from equations (9)-(11)) are unable to predict the

amplitude of fluctuating waves [3], they can still be used as initial conditions to *nonlinear* methods. Since both the *Local Stability Theory* (LST) and the *Linear Parabolized Stability Equations* (LPSE) (see section 4.3) are derived from equations (9)-(11), only the growth rate and the form of the perturbation vectors obtained with these methods have a physical meaning [4].

4.3 Nonlocal stability analysis

In the *nonlocal* stability analysis, the *history effects* associated with the initial conditions and the varying properties of the flow are taken into account [2] which results in a stream-wise dependency of both the amplitude and the phase functions. However, the evolution of the perturbation in the streamwise direction is still considered small and $\partial_x^2 \hat{q}$ is assumed to scale with $O(Re_\delta^{-2})$ and thus negligible compared to the first derivative ($O(Re_\delta^{-1})$), which leads to a system of parabolic equations. First introduced by Hall [5], the idea of studying the evolution of T-S waves through parabolized equations was further developed by Itoh [6] and by Bertolotti [7],[8],[9],[10], who proposed the *Nonlinear Parabolized Stability Equations (NPSE)*. More recently, Kuehl [11] and Moyes [12] used the NPSE approach to study the secondary instability of crossflow in hypersonic conditions. The nonlinear effects are adressed in section 4.6. The perturbation vector in the LPSE approach takes the following form:

$$q' = \hat{q}(x,y) \exp\left(\int_0^x i\alpha(x)dx + i\beta z - i\omega t\right) \tag{17}$$

Introducing equation (17) into (9)-(11) leads to the following system of equations

$$\mathbf{L}\vec{\phi} + \mathbf{S}\frac{\partial\vec{\phi}}{\partial y} + \mathbf{T}\frac{\partial^2\vec{\phi}}{\partial y^2} + \mathbf{P}\frac{d\alpha}{dx} = \mathbf{M}\frac{\partial\vec{\phi}}{\partial x} \tag{18}$$

Where the matrices **L**, **S** and **T** are the same than in (14) while the matrices **P** and **M** account for the stream-wise derivatives. The boundary conditions are also extremely similar to those of the LST except that $\hat{p}$ at the wall was relaxed.

$$\hat{u}(0) = \hat{v}(0) = \hat{w}(0) = \hat{T}(0) = 0 \tag{19}$$

$$\hat{u}_y(\infty) = \hat{v}_y(\infty) = \hat{w}_y(\infty) = \hat{p}_y(\infty) = \hat{T}_y(\infty) = 0 \tag{20}$$

4.4 Normalization condition

In the PSE approach, the stream-wise wave number α is also a variable in the problem; we thus lack one more equation to close the system. For this

purpose, we introduce the so-called *Normalization condition* (or *Auxiliary condition*) [2] [13]. Physically, the role of the normalization condition is to transfer energy from the amplitude function to the phase function so that $\vec{\phi}_{xx} \sim O(Re_{\delta}^{-2})$.

$$\int_0^{\infty} \vec{\phi}^{\dagger} \frac{\partial \vec{\phi}}{\partial x} dy = 0 \tag{21}$$

This supplementary condition prevents the use of a fully implicit method. We must therefore solve this system with a combination of numerical methods; a monolithic discretization in the normal direction and an explicit or semi-implicit scheme in the stream-wise direction.

4.5 Initial conditions

Traditionally, the PSE equations have been used to study the evolution in space of Tollmien-Schlichting (T-S) waves obtained with the help of the LST. However, the PSE approach has been recently used to study the interaction between acoustic and vortical waves in transitional flows [14],[15],[16],[17]. In such case, the initialization of the acoustic waves should be made with experimental data, asymptotic expansion or by directly imposing the K-H eigenmodes [16].

4.6 Nonlinear forcing terms

In the *Nonlinear Parabolized Stability Equations (NPSE)* approach, nonlinearities and mode interactions are no longer neglected. Therefore, instead of solving every mode separately, all modes must be solved simultaneously as they are coupled through nonlinear forcing terms. In this regards, the disturbance vector is expanded in terms of its truncated Fourier components. Assuming periodicity in time and in the span-wise direction, ϕ' now takes the following form:

$$\boldsymbol{\phi}'(\mathbf{x}, t) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} \tilde{\boldsymbol{\phi}}_{m,n}(x, y) \exp i(m\beta z - n\omega t) \tag{22}$$

$$\tilde{\boldsymbol{\phi}}(x, y) = \hat{\boldsymbol{\phi}}_{m,n}(x, y) \exp \left[i \int_{x_0}^{x} \alpha_{m,n}(x) dx \right] \tag{23}$$

Indeed, since NPSE account for the nonlinearities, equations (9)-(11) must be modified to take into account the perturbation cross-products:

$$\frac{\partial u_i'}{\partial x_i}+\left(\frac{T'}{\overline{T}}+\frac{p'}{\overline{p}}\right)\frac{\partial \overline{u}_i}{\partial x_i}=\frac{p'}{\overline{p}\overline{T}}\frac{D\overline{T}}{Dt}+\frac{1}{\overline{T}}\left[\frac{DT'}{Dt}-u_i'\frac{\partial \overline{T}}{\partial x_i}\right]-\frac{T'}{\overline{p}\overline{T}}\frac{D\overline{p}}{Dt}-\frac{1}{\overline{p}}\left[\frac{Dp'}{Dt}+u_i'\frac{\partial \overline{p}}{\partial x_i}\right]+F_{cont} \tag{24}$$

$$\left(\frac{\overline{\rho}}{\overline{p}}p'-\frac{\overline{\rho}}{\overline{T}}T'\right)\frac{D\overline{u}_i}{Dt}+\overline{\rho}\left(\frac{Du_i'}{Dt}+u_j'\frac{\partial \overline{u}_i}{\partial x_j}\right)=-\frac{\partial p'}{\partial x_i}+\frac{1}{Re}\frac{\partial \sigma_{ij}'}{\partial x_j}+F_{mom} \tag{25}$$

$$\begin{aligned}\left(\frac{\overline{\rho}c_p}{\overline{p}}p'-\frac{\overline{\rho}c_p}{\overline{T}}T'\right)\frac{D\overline{T}}{Dt}+\overline{\rho}c_p\left(\frac{DT'}{Dt}+u_j'\frac{\partial \overline{T}}{\partial x_j}\right)&=\frac{1}{RePr}\frac{\partial}{\partial x_i}\left(\overline{\lambda}\frac{\partial T'}{\partial x_i}+\lambda'\frac{\partial \overline{T}}{\partial x_i}\right)\\&+(\gamma-1)Ma^2\left(\overline{u}_i\frac{\partial p'}{\partial x_i}+u_i'\frac{\partial \overline{p}}{\partial x_i}\right)+\frac{(\gamma-1)Ma^2}{Re}\phi_{ij}'+F_{ener}\end{aligned} \tag{26}$$

Where the nonlinear terms are defined as:

$$\begin{aligned}F_{cont}=\;&p'T'\frac{\partial \overline{u}_i}{\partial x_i}+p'\overline{T}\frac{\partial u_i'}{\partial x_i}+\overline{p}T'\frac{\partial u_i'}{\partial x_i}+p'T'\frac{\partial u_i'}{\partial x_i}\\&+T'\left[\frac{\partial p'}{\partial t}+\overline{u}_i\frac{\partial p'}{\partial x_i}+u_i'\frac{\partial \overline{p}}{\partial x_i}+u_i'\frac{\partial p'}{\partial x_i}\right]\\&-p'\left[\frac{\partial T'}{\partial t}+\overline{u}_i\frac{\partial T'}{\partial x_i}+u_i'\frac{\partial \overline{T}}{\partial x_i}+u_i'\frac{\partial T'}{\partial x_i}\right]\end{aligned} \tag{27}$$

$$\begin{aligned}F_{mom}=\;&\frac{\mu'}{Re}\left[\frac{\partial^2 u_j'}{\partial x_j^2}+\frac{\partial^2 u_j'}{\partial x_i\partial x_j}-\frac{2}{3}\frac{\partial^2 u_k'}{\partial x_k^2}\delta_{ij}\right]\\&+\frac{1}{Re}\frac{\partial \mu'}{\partial x_j}\left[\frac{\partial u_i'}{\partial x_j}+\frac{\partial u_j'}{\partial x_i}-\frac{2}{3}\frac{\partial u_k'}{\partial x_k}\delta_{ij}\right]\\&-\rho'\left[\frac{\partial u_i'}{\partial t}+\overline{u}_j\frac{\partial u_i'}{\partial x_j}+u_j'\frac{\partial \overline{u}_i}{\partial x_j}\right]\\&-\overline{\rho}u_j'\frac{\partial u_i'}{\partial x_j}-\rho'u_j'\frac{\partial u_j'}{\partial x_j}\end{aligned} \tag{28}$$

$$
\begin{aligned}
F_{ener} = & \frac{\lambda'}{RePr}\frac{\partial^2 T'}{\partial x_i^2} + \frac{1}{RePr}\frac{\partial \lambda'}{\partial x_i}\frac{\partial T'}{\partial x_i} + (\gamma-1)Ma^2\left(u_i'\frac{\partial p'}{\partial x_i}\right) \\
& + \frac{(\gamma-1)Ma^2}{Re}\left[\overline{\mu}\left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} - \frac{2}{3}\frac{\partial u_k'}{\partial x_k}\delta_{ij}\right)\frac{\partial u_j'}{\partial x_j}\right] \\
& + \frac{(\gamma-1)Ma^2}{Re}\left[\mu'\left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} - \frac{2}{3}\frac{\partial u_k'}{\partial x_k}\delta_{ij}\right)\frac{\partial \overline{u}_j}{\partial x_j}\right] \\
& + \frac{(\gamma-1)Ma^2}{Re}\left[\mu'\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} - \frac{2}{3}\frac{\partial \overline{u}_k}{\partial x_k}\delta_{ij}\right)\frac{\partial u_j'}{\partial x_j}\right] \\
& + \frac{(\gamma-1)Ma^2}{Re}\left[\mu'\left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} - \frac{2}{3}\frac{\partial u_k'}{\partial x_k}\delta_{ij}\right)\frac{\partial u_j'}{\partial x_j}\right] \\
& - \left[\rho' c_p\left(\frac{\partial T'}{\partial t} + \overline{u}_j\frac{\partial T'}{\partial x_j} + u_j'\frac{\partial \overline{T}}{\partial x_j}\right) + \overline{\rho} c_p u_j'\frac{\partial T'}{\partial x_j} + \rho' c_p u_j'\frac{\partial T'}{\partial x_j}\right]
\end{aligned}
\tag{29}
$$

Introducing equation (22) into equations (24)-(26) leads to the following system:

$$
\left\{\mathbf{L}\vec{\phi} + \mathbf{S}\frac{\partial \vec{\phi}}{\partial y} + \mathbf{T}\frac{\partial^2 \vec{\phi}}{\partial y^2} + \mathbf{P}\frac{d\alpha}{dx}\right\}_{m,n} = \Lambda_{m,n}\exp\left(-i\int_x \alpha_{m,n}dx\right)\left\{\tilde{\mathbf{F}} + \mathbf{M}\frac{\partial \vec{\phi}}{\partial x}\right\}_{m,n} \tag{30}
$$

The system (30) is mode-specific and must therefore be solved for every {m,n} combination. Moreover, every mode is subject to the Normalization condition (21) Concerning the nonlinear forcing terms, they can either be evaluated in the physical or the spectral space. The direct method expands the nonlinearities and evaluates the forcing terms using a Cauchy summation. The alternative is to use an Inverse Fast Fourier Transform (IFFT), compute the forcing terms in the physical space using equations (27)-(29) and finally use a Fast Fourier Transform (FFT) to convert the forcing terms back into the spectral space. The FFT and IFFT algorithms are well optimized while the direct method requires the derivation and numerical evaluation of dozens of intermediate terms. Airiau even found that the FFT method was not only simpler but also faster [18]. For these reasons, we began to implement the FFT method in the code.

The main advantage of integrating nonlinear effects and mode interactions within the modal analysis is that the perturbation vector ϕ' is predicted with the correct amplitude. In other words, one can simply use an IFFT to reconstruct the perturbation ϕ' from its eigenmodes and predict transition on the basis of the friction coefficient from the Reynolds stresses or from the turbulent kinetic energy.

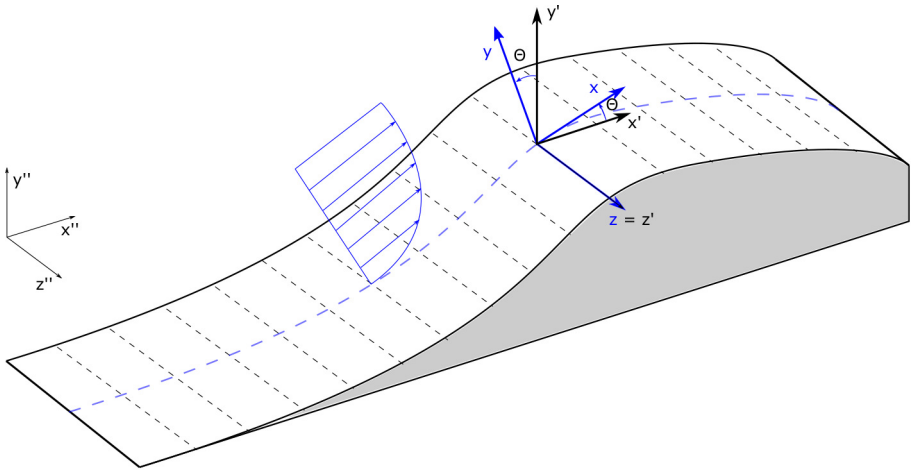

Figure 1: Illustration of the ortho-curvilinear coordinate system

4.7 Curvilinear coordinates

One of the main objectives of the project is to develop a tool to predict transition in compressible flows over complex geometries. The marching procedure is performed in the stream-wise direction and is therefore incompatible with general Cartesian coordinates, except in the case where x is perfectly aligned with the streamlines (which is the case for very simple flows such as Blasius, Poiseuille and Couette flows). To overcome this difficulty, we define the following orthocurvilinear coordinate system as illustrated on figure 1. For conciseness, the Cartesian and the translated Cartesian systems are referred by the superscripts $''$ and $'$ while the ortho-curvilinear system is simply noted $\{x, y, z\}$, without any superscript.

$$x = x' \cos(\theta(x)) - y' \sin(\theta(x)) \tag{31}$$

$$y = x' \sin(\theta(x)) + y' \cos(\theta(x)) \tag{32}$$

$$z = z' \tag{33}$$

The geometric Jacobian associated with equations (31) -(33) can be written in matrix format:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial x'} & \frac{\partial x}{\partial y'} & 0 \\ \frac{\partial y}{\partial x'} & \frac{\partial y}{\partial y'} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{34}$$

Where the coefficients are given by:

$$\frac{\partial x}{\partial x'} = \frac{\cos\theta}{1 + y\frac{d\theta}{dx}} = J_{xx} \qquad \frac{\partial x}{\partial y'} = -\frac{\sin\theta}{1 + y\frac{d\theta}{dx}} = J_{yx} \tag{35}$$

$$\frac{\partial y}{\partial x'} = \sin\theta + x\frac{d\theta}{dx}\frac{\partial x}{\partial x'} = J_{xy} \qquad \frac{\partial y}{\partial y'} = \cos\theta + x\frac{d\theta}{dx}\frac{\partial x}{\partial y'} = J_{yy} \tag{36}$$

Using the chain rule, the Cartesian derivatives of a scalar function F can be expressed in the curvilinear coordinate system using equations (37)-(41). The complete compressible Navier-Stokes equations as well as the modal stability equations in curvilinear coordinates can be found in appendix.

$$\frac{\partial F}{\partial x'} = \frac{\partial F}{\partial x}J_{xx} + \frac{\partial F}{\partial y}J_{yx} \tag{37}$$

$$\frac{\partial F}{\partial y'} = \frac{\partial F}{\partial x}J_{xy} + \frac{\partial F}{\partial y}J_{yy} \tag{38}$$

$$\frac{\partial^2 F}{\partial x'^2} = \frac{\partial^2 F}{\partial x^2}J_{xx}^2 + \frac{\partial^2 F}{\partial x\partial y}2J_{yx}J_{xx} + \frac{\partial^2 F}{\partial y^2}J_{yx}^2 \tag{39}$$

$$\frac{\partial^2 F}{\partial y'^2} = \frac{\partial^2 F}{\partial x^2}J_{xy}^2 + \frac{\partial^2 F}{\partial x\partial y}2J_{xy}J_{yy} + \frac{\partial^2 F}{\partial y^2}J_{yy}^2 \tag{40}$$

$$\frac{\partial^2 F}{\partial x'\partial y'} = \frac{\partial^2 F}{\partial x^2}J_{xx}J_{xy} + \frac{\partial^2 F}{\partial x\partial y}\left(J_{yy}J_{xx} + J_{yx}J_{xy}\right) + \frac{\partial^2 F}{\partial y^2}J_{yx}J_{xy} \tag{41}$$

At the moment, the geometric Jacobian is only defined for the normal and tangential directions, but we plan to extend its definition in the third direction to account for cross-flow. The detail of the numerical implementation of the modal stability equations as well as the Compressible Navier-Stokes equations are addressed in the next section.

5 Code structure, numerical implementation

5.1 How it works

The figures 2 and 3 illustrate how the different module communicate together within Krypton. First, all the input of the simulation, including the flow properties (Re, Ma, T_w, ω, β, crossflow intensity) and the specification of the numerical methods (BDF order, discretization, geometry) are specified in the main.py module. These input are then passed to the Discret.py module which is used to compute the relevant differentiation matrices (spectral collocation in y and FD/BDF in x). The module Discret.py also includes different classes and methods for multidimensional interpolation and numerical integration that were developed directly from the basis functions to ensure consistency with the numerical schemes.

5.1.1 Base flow: Laminar Compressible Navier-Stokes solver

The next step is to generate the base flow. For this purpose, the module LaminarFlow.py first create an object associated with the discretized Navier-Stokes equations. This object, defined in NS.py, contains all the relevant methods to build the linear operator and iteratively solve the problem. The CNSE are solved using a Picard iterative method –also referred to as the fixed point algorithm. In the Picard's method, the nonlinear terms are linearized with the solution at the previous step. For instance, the convective terms in the momentum equation are defined as:

$$u_j^k \frac{\partial u_i^k}{\partial x_j} \approx u_j^{k-1} \frac{\partial u_j^k}{\partial x_j} \tag{42}$$

The Picard's method is relatively simple and easy to implement, but only has a linear convergence. In a near future, this method will be supplemented by the Newton's method to increase the rate of convergence to second order. The present algorithm usually takes around 20 iterations to converge for the computation of a simple zero pressure gradient (ZPG) flat plate simulation and has a time complexity of $O(n^{1.2})$, meaning that if the number of equations is doubled, the computation time will increase by a factor $2^{1.2}$.

For the boundary condition, we impose the velocity and temperature profiles at the entrance using tabulated Compressible Falkner-Skan-Cooke data. The inlet pressure is extrapolated from the solution. We impose a traction-free, given by equation (43) boundary condition at the outlet and in the free-stream. In most cases, the wall temperature is kept constant ($\frac{T_w}{T_\infty} = C$).

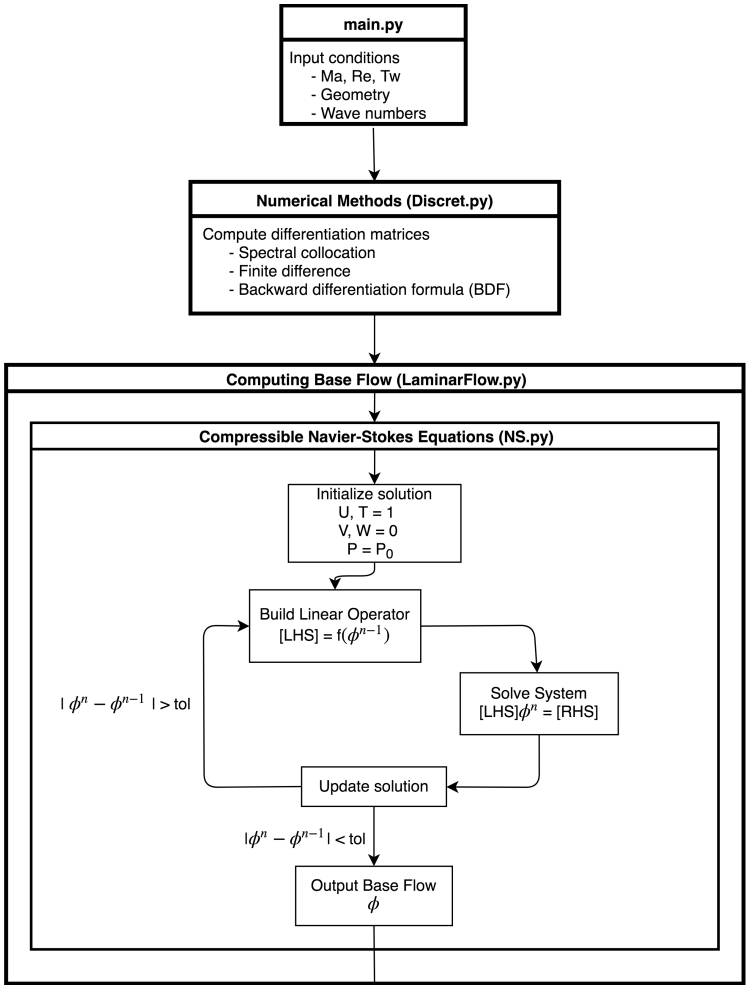

Figure 2: Krypton block diagram (part 1 of 2)

$$\frac{\mu}{Re}\left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right] - p\delta_{ij} = 0 \tag{43}$$

5.1.2 Fluctuating flow : Modal Stability Solver

Once the tolerance on the residual is met, the solution derivatives are computed and the base flow is passed to the modal stability solver. The modal stability solver is, by construction, very similar to the laminar flow solver, the main difference lies in the definition of the object associated with the PSE. First, since the equations are parabolized, the problem is essentially divided into several smaller problems that will be solved sequentially through a marching procedure. Both the local (LST) and the nonlocal (LPSE) procedure are implemented in the PSE.py module since they roughly only differ from each other by the addition of stream-wise derivatives and a different convergence criteria. The local stability problem is generally solved as a matrix eigenvalue problem [19, 2]:

$$[A]\Phi = \omega[B]\Phi \tag{44}$$

In such case, the system can be solved with classical numerical methods, for instance, the standard QZ algorithm or inverse Rayleigh iterations. However, these methods are computationally expensive and often lead to inaccurate results. We instead chose to follow a simpler and more robust algorithm: rather than finding ω such that (44) is satisfied, we impose ω and find α such that the following boundary conditions are satisfied:

$$\hat{u}(0) = \hat{v}(0) = \hat{w}(0) = \hat{T}(0) = \hat{p}(0) - 1 = 0 \tag{45}$$

$$\hat{u}_y(\infty) = \hat{v}_y(\infty) = \hat{w}_y(\infty) = \hat{T}_y(\infty) = \hat{p}_y(\infty) = 0 \tag{46}$$

To do so, we impose the boundary conditions on all variables except $\hat{v}$ at the wall and iterate on α with the Newton's method until $|\hat{v}(0)| < \epsilon$ (we set ϵ to machine zero in our calculations). This approach is robust, lightweight and converge quadratically. The results obtained from the local stability theory are then used as initial conditions for the LPSE calculation.

The procedure for the LPSE calculations is almost identical to the LST procedure; there are only slight procedural differences. First, the boundary conditions are slightly different: $\hat{v}(0)$ is now explicitly set to zero while the condition on $\hat{p}$ at the wall is relaxed. In the LPSE approach, the normalization condition (see section 4.4) replaces the condition on $\hat{v}$ at the wall in the Newton's iterative process; the objective is to find α for which the condition (21) is satisfied. Once the solution is converged at the first station, the marching procedure begins. For this purpose, we opted for an implicit BDF-based ap-

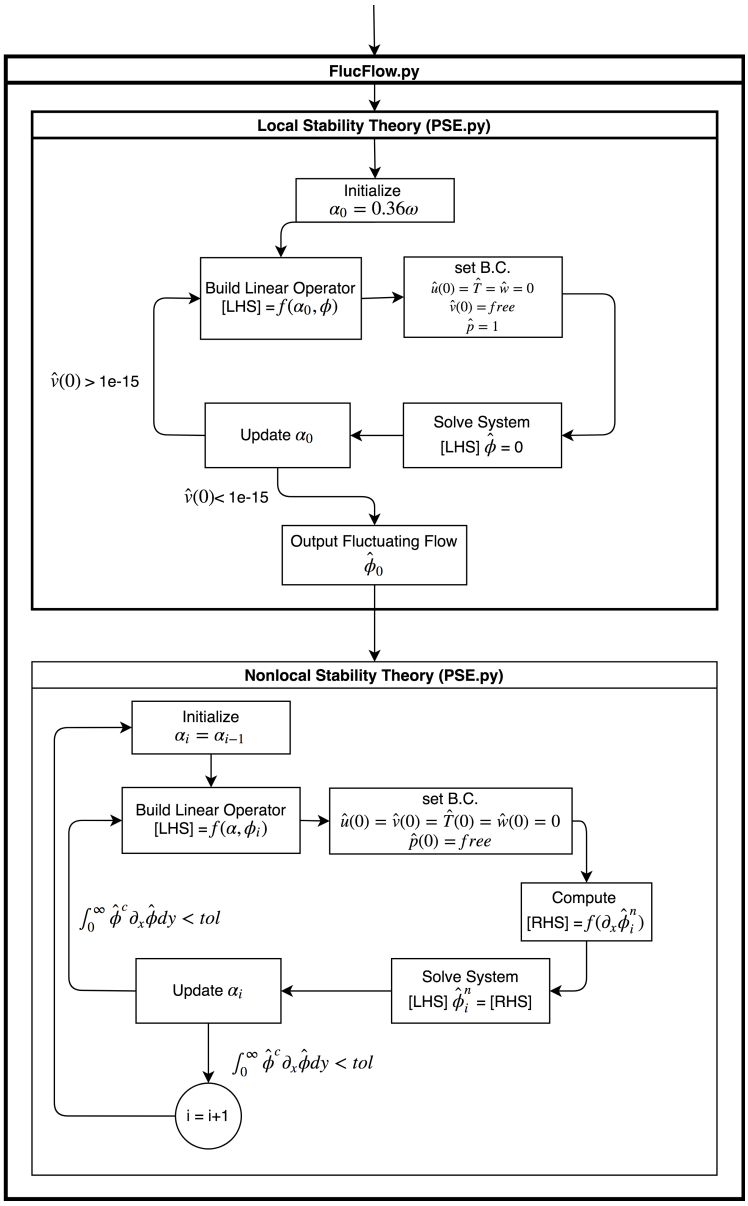

Figure 3: Krypton block diagram (part 2 of 2)

proach, as described in the previous section. We define a station as a position on the x-axis.

6 Numerical implementation

6.1 Numerical Methods

One of the main advantage of the Parabolized Stability Equations is their ability to be solved using a computationally advantageous marching procedure. This approach allows the use of two different numerical methods in the stream-wise and normal direction. For instance, a spectral collocation method in the normal direction and Backward Differentiation Formulas (BDF) in the stream-wise direction. This approach was also preferred by several other authors [20, 21, 22].

6.2 Spectral collocation method

The spectral collocation method is an accurate class of numerical techniques for the solution of linear/nonlinear partial differential equations (PDE). The main idea behind the spectral collocation method is to expand the solution in terms of global basis functions so that the numerical solution satisfies the PDE at the so-called *collocation points*, or also called *Chebyshev Gauss-Lobatto points* [23].

$$\xi_j = \cos\frac{\pi j}{n} \qquad\qquad \xi_j \in [-1, 1] \tag{47}$$

The general spectral representation of a solution to a PDE takes the following form.

$$u^*(\xi) = \sum_{j=0}^{n} \phi_j(\xi) u(\xi_j) \tag{48}$$

Where $\phi_j(\xi)$ are the basis functions. In the case of modal stability theory, as the solution is not periodic, we use Chebyshev polynomials of the following form:

$$\phi_j(\xi) = \left(\frac{1-\xi_j^2}{\xi-\xi_j}\right)\frac{T_n'(\xi)}{n^2 c_j}(-1)^{j+1}; \qquad c_0 = c_n = 2,\ \ c_j = 1 \tag{49}$$

Where T_n and T_n' are the Chebyshev polynomials

$$T_n(\xi) = \cos(n\cos^{-1}\xi) \tag{50}$$

$$T_n'(\xi) = \frac{n\sin(n\cos^{-1}\xi)}{\sqrt{1-\xi^2}} \tag{51}$$

6.2.1 Differentiation

The Chebyshev polynomials' derivatives can be easily computed from the following expression

$$\left.\frac{du^*}{d\xi}\right|_i = \sum_{j=0}^{n} \Phi_{ij}(\xi) u(\xi_j) \tag{52}$$

Where

$$\Phi_{ij} = \frac{c_i}{c_j}\frac{(-1)^{i+j}}{\xi_i - \xi_j} \tag{53}$$

$$\Phi_{ii} = -\frac{\xi_i}{2(1-\xi_i^2)} \tag{54}$$

$$\Phi_{00} = \frac{2m^2+1}{6} \tag{55}$$

$$\Phi_{nn} = -\frac{2m^2+1}{6} \tag{56}$$

The collocation points ξ_j are defined on $[-1, 1]$, to retrieve the derivative on the physical domain $y_j \in [y_0, y_f]$, a scaling function S_i must be applied.

$$\left.\frac{du^*}{dy}\right|_i = \left.\frac{du^*}{d\xi}\frac{d\xi}{dy}\right|_i = \sum_{j=0}^{n} S_i \Phi_{ij} u(\xi_j) = \sum_{j=0}^{n} D_{ij} u(y_j) \tag{57}$$

Higher derivatives are defined through the powers of the matrix D. For example, the second derivative is D^2, the third is D^3, etc. [24].

$$\left.\frac{d^2u^*}{dy^2}\right|_i = \sum_{j=0}^{n} D_{ik} D_{kj} u(y_j) \tag{58}$$

Finally, the solution and its derivatives can be represented in a more convenient matrix-vector form:

$$\mathbf{u}^* = \mathbf{I} \cdot \mathbf{u} \qquad \frac{d\mathbf{u}^*}{dy} = \mathbf{D} \cdot \mathbf{u} \qquad \frac{d^2\mathbf{u}^*}{dy^2} = \mathbf{D}^2 \cdot \mathbf{u} \tag{59}$$

6.2.2 Multi-Domain Spectral collocation method

The spectral collocation method leads to very accurate results as n is increased. However, in practice, the collocation points are not necessarily well distributed for the shape of the solution and increasing the degree of the polynomial interpolation beyond 200 or 300 may lead to numerical instabilities. Moreover, the classical spectral collocation method requires the resolution

of a dense matrix which can be a problem for performance optimization. The use of a multi-domain spectral collocation approach helps to alleviate these issues by converting the system matrix into a sparse matrix and allowing the user to define a custom mesh. The general idea behind this approach is to divide the computation domain into multiple sub-domain and apply the classical spectral collocation method locally, on every *element*. Appropriate boundary conditions at the element interfaces ensure the smoothness of the solution. It is thus possible to define a global differentiation matrix $\mathcal{D}$. For instance, for 3 sub-domains $\mathcal{D}$ is defined as

$$\mathcal{D} = \begin{bmatrix} \mathbf{D}_0 & 0 & 0 \\ 0 & \mathbf{D}_1 & 0 \\ 0 & 0 & \mathbf{D}_2 \end{bmatrix} \tag{60}$$

6.3 Finite Difference (FD) scheme and Backward Differentiation Formulas (BDF)

For the base flow, we solve the compressible Navier-Stokes equations without any assumption, the system is thus elliptic and requires an implicit numerical method. We use a high-order central finite difference scheme in the stream-wise direction and a multi-domain spectral collocation method in the normal direction. We chose this hybrid approach instead of a pure spectral or a pure finite difference method so both the fluctuating and the base flow are solved on the same grid, avoiding any interpolation error. Moreover, the use of the spectral collocation method in the stream-wise direction induces slight oscillations in the solution which affect the stability analysis. Whereas these oscillations usually vanish as the number of points is increased, we solved the problem by taking advantage of the dissipative character of the FD approach.

The finite difference coefficients are computed using the *Findiff* Python package [25]. Findiff computes the finite difference coefficients of any desired order on uniform and non-uniform grids. Most of our calculations were made with sixth-order central finite difference scheme on uniform grids, but the code has the potential to be used on non-uniform grids with higher order schemes.

The PSE are solved using an implicit spatial-marching procedure. For this purpose, we chose a classical Backward Differentiation Formulas (BDF) approach. Just as for the FD coefficients, the BDF coefficients are computed using Findiff which is particularly useful for complex geometries since Findiff supports variable stepsize.

7 Sample results

7.1 Base flow

As mentioned earlier, the base flow is solved in curvilinear coordinates using an hybrid numerical approach; finite differences in the stream-wise direction and a spectral multi-element collocation method in the normal direction. To better illustrate this feature, a typical mesh for a forward facing step (FFS) is shown on figure 4. The equations are solved at the line junctions.

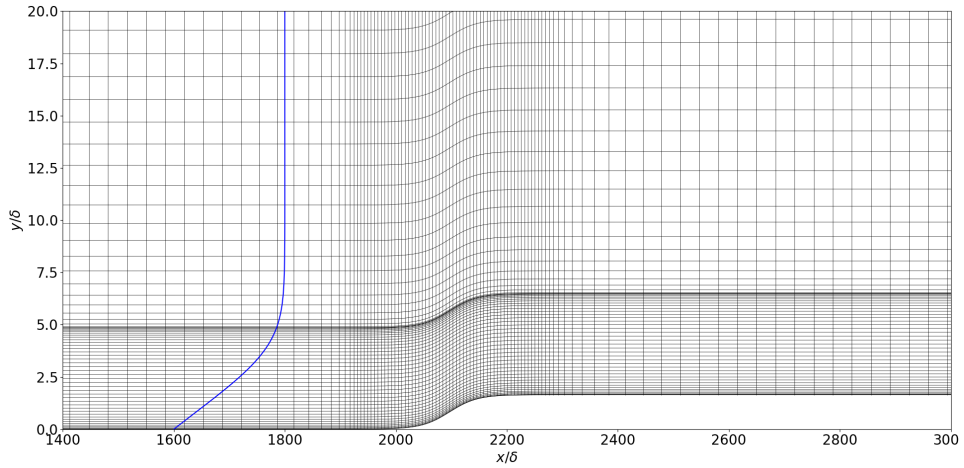

Figure 4: A typical mesh for the forward facing step case (FFS), only a small portion of the domain is displayed. The U-velocity profile is displayed in blue for scaling

Indeed, the solver can only handle cases in which the streamlines are parallel to the surface. That is to say, only fully attached or locally reversed flows can be studied. However, this is in accordance with the PSE assumptions; in case of detachment or near stagnation points, the streamlines are not well defined and it become unclear whether or not the PSE is applicable.

7.2 Measure of stability

The growth rate of the disturbance σ is computed from α (stream-wise wavenumber) and E_k (disturbance kinetic energy) with equation (62). The first term, $-Im(\alpha)$, corresponds to the contribution of the exponential part of the disturbances while the second term is the correction related to the change of the amplitude function [2]. When $\sigma > 0$, the disturbances are amplified while when $\sigma < 0$ they are damped. The figure 6 shows the evolution of the

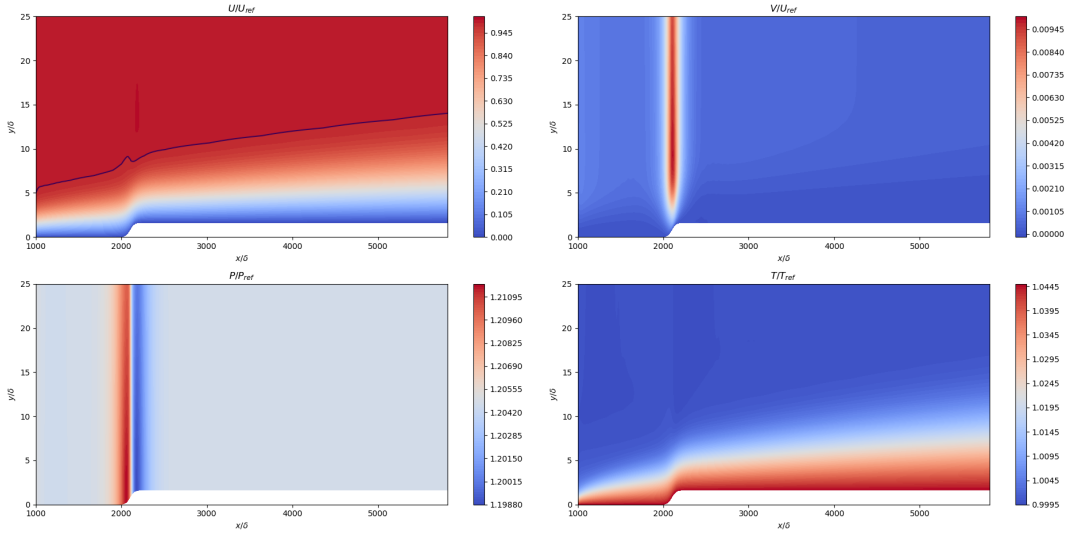

Figure 5: Base flow for the case of a FFS at Ma = 0.77, T_w/T_∞ = 1.044 and Re_δ = 1000 (inlet), Re_δ = 2400 (outlet). These results were obtained using a 8^{th} order central finite differences on a non-uniform grid.

Figure 6: Comparison of the growth rate obtained with the LPSE approach at Mach 0.1 (F = 100), compared with incompressible results from [2]

growth rate along the x-direction. The results are compared with Juniper's results (incompressible) [2].

$$\sigma = -Im(\alpha) + \frac{\partial}{\partial x}\left[\ln\left(\sqrt{E_k}\right)\right] \tag{61}$$

$$E_k = \int_0^\infty \overline{\rho}\left(|\hat{u}|^2+|\hat{v}|^2+|\hat{w}|^2\right) dy \tag{62}$$

The neutral stability curve provides valuable information about the stability of the flow in given conditions (Re_δ, ω, β, Ma). One can compute the neutral stability curve by running multiple PSE simulations for different values of ω while keeping β to a fixed value. -2cm-2cm As seen on the figure 7.2, we have a good agreement with the experimental data, except for the peak height. However, the PSE results agree almost perfectly with the DNS predictions. The disparities between the experiments and the DNS/PSE results are probably due to the experimental setups themselves: background noise, wall vibration and/or surface roughness may induce perturbation in the free-stream and extend the zone of non-stability further than the idealized DNS/PSE case.

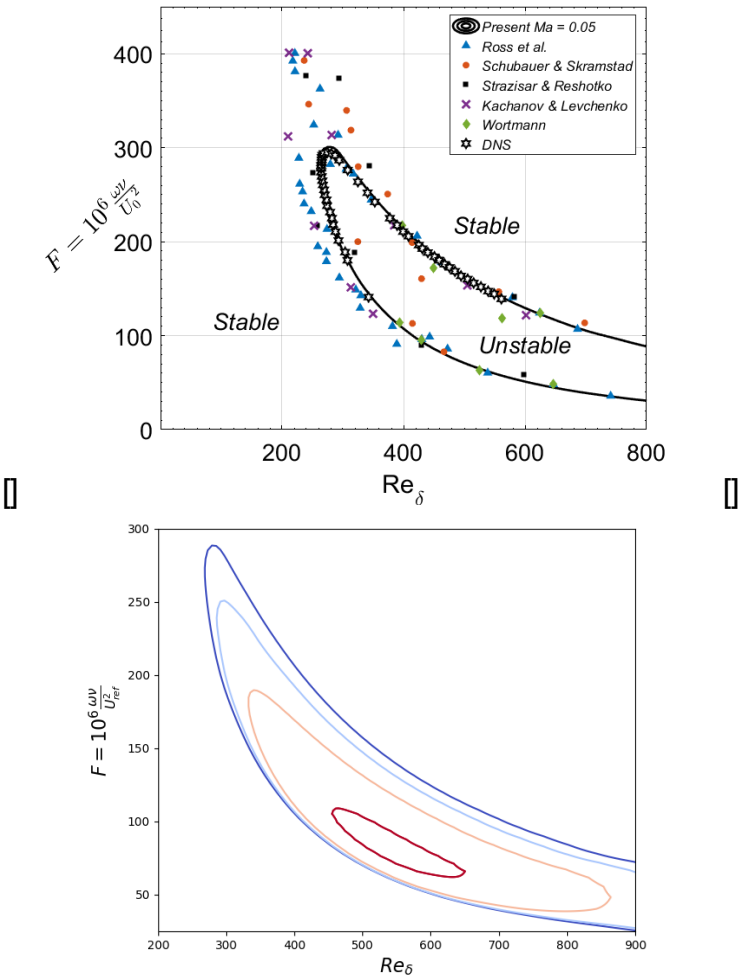

Figure 7: (a) Neutral stability curve, Ma = 0.05 ZPG flat plate (b) Neutral stability curve for different Ma numbers ranging from 0.3 (blue) to 0.9 (red)

7.3 N-factor

The e^N method is widely used in aerodynamic design. Based on empirical measurements, colossal efforts have been deployed by the industry to collect the experimental data needed for the transition correlations [26]. The n-factor can be seen as a measure of the amplification of a particular mode along the stream path. It should however be noted that the n-factor is function of the temporal and span-wise wavenumbers (ω and β); every combination of wavenumbers has a corresponding n-factor curve that can not be directly correlated to the transition position.

$$\text{n-factor} = -\int_{x_0}^{x_i} \sigma dx \tag{63}$$

To remove this ambiguity, Ingen [27], [28] defined the *N-factor* corresponding to the envelope of the n-factor curves (see figure 7.3).

From a physical point of view, the e^N method is nothing more than a correlation between the transition position (obtained experimentally) and the N-factor (obtained with the modal stability theory). The method is thus highly data-driven and specific to the geometry of the experiment: a critical N-factor must be correlated for every geometry and flow conditions. The use of the method beyond its range of applicability may lead to considerable error. For the Blasius boundary layer, Ingen [28] arrived to the following critical N-factors for the beginning and end of transition.

$$N_B = 2.13 - 6.18 \log_{10}(Tu) \tag{64}$$

$$N_E = 5.00 - 6.18 \log_{10}(Tu) \tag{65}$$

The relations (64) and (65) are based on the experiments of Schubauer and Skramstad [29] and are valid for $Tu > 0.1\%$. Combining the relations 64 and 65 and the results plotted on the figure 7.3, one can obtain an estimate of the the transition location. The figure 7.3 shows the prediction of the transition location for an incompressible zero pressure gradient (ZPG) flat-plate boundary layer. The RANS and the e^N method lead to similar results for the *End of transition line*, which is expected since both of the approaches have been optimized for this exact test case (ZPG transitional flat plate boundary layer).

References

[1] M. Montagnac, Variable Normalization (Nondimensionalization and Scaling) for Navier-Stokes Equations: A Practical Guide (2013) 1–14.

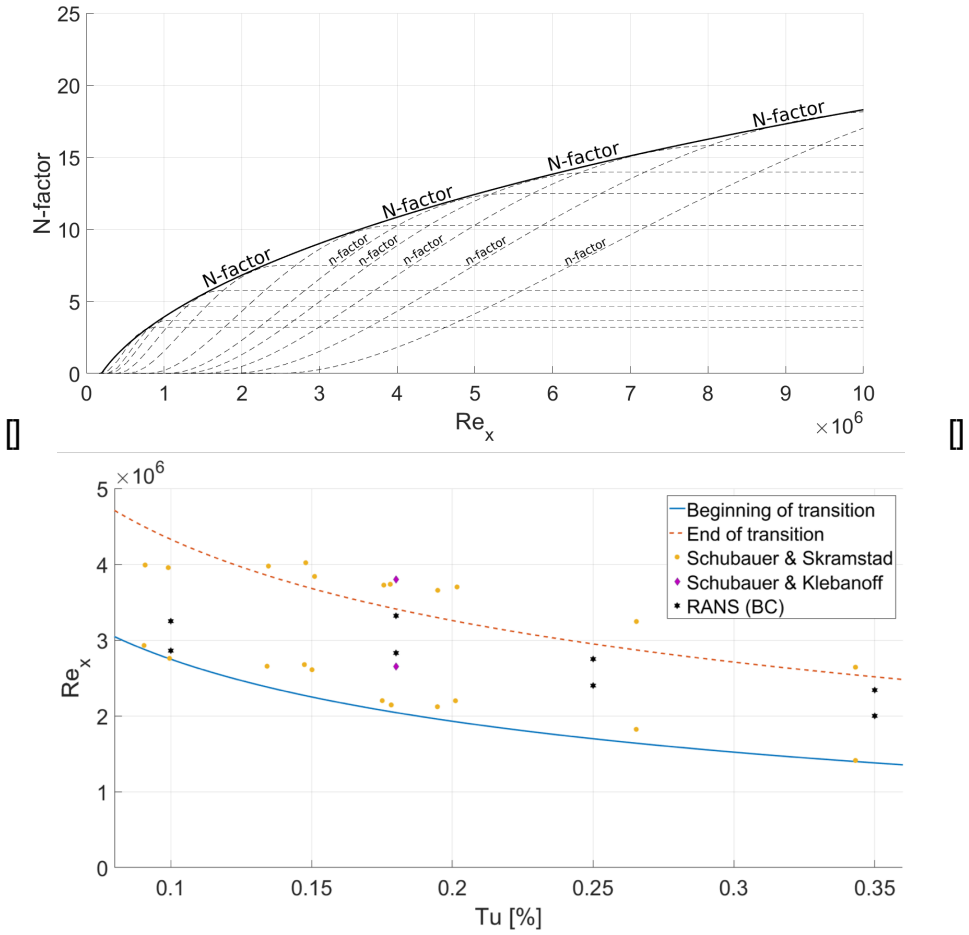

Figure 8: (a) N-factor Vs n-factor, (b) PSE predictions of the beginning and end of transition as a function of the turbulence intensity (Tu [%]), Ma = 0.1 ZPG flat plate

[2] M. P. Juniper, A. Hanifi, V. Theofilis, Modal Stability TheoryLecture notes from the FLOW-NORDITA Summer School on Advanced Instability Methods for Complex Flows, Stockholm, Sweden, 2013 ¹, Applied Mechanics Reviews 66 (2) (2014) 021004. doi:10.1115/1.4026604.
URL http://appliedmechanicsreviews.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4026604

[3] M. K. Ibrahim, Y. Nakamura, Correlation between Linear stability theory and transition of compressible jet shear layer, Transactions of the Japan Society for Aeronautical and Space Sciences 45 (147) (2002) 35–44. doi:10.2322/tjsass.45.35.

[4] T. Colonius, A. Samanta, K. Gudmundsson, Parabolized stability equation models of large-scale jet mixing noise, Procedia Engineering 6 (2010) 64–73. doi:10.1016/J.PROENG.2010.09.008.

[5] P. Hall, The linear development of Görtler vortices in growing boundary layers, Journal of Fluid Mechanics 130 (-1) (1983) 41. doi:10.1017/S0022112083000968.
URL http://www.journals.cambridge.org/abstract_S0022112083000968

[6] N. Itoh, Secondary Instability of Laminar Flows, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 375 (1763) (1981) 565–578. doi:10.1098/rspa.1981.0068.
URL http://rspa.royalsocietypublishing.org/cgi/doi/10.1098/rspa.1981.0068

[7] F. Bertolotti, The Effect of Approximations to the Thermodynamic Properties on the Stability of Compressible Boundary Layer Flow, in: Instability and Transition, Springer New York, New York, NY, 1990, pp. 83–98. doi:10.1007/978-1-4612-3432-6{_}6.
URL http://link.springer.com/10.1007/978-1-4612-3432-6_6

[8] F. P. Bertolotti, T. Herbert, Analysis of the linear stability of compressible boundary layers using the PSE, Theoretical and Computational Fluid Dynamics 3 (2) (1991) 117–124. doi:10.1007/BF00271620.

[9] V. Tolieng, B. Prasirtsak, J. Sitdhipol, N. Thongchul, S. Tanasupawat, Identification and lactic acid production of bacteria isolated from soils and tree barks, Malaysian Journal of Microbiology 13 (2) (2017) 100–108. doi:10.1017/CBO9781107415324.004.

[10] F. Bertolotti, Linear and nonlinear stability of the Blasius boundary layer, Journal of Fluid Mechanics 242 (-1) (1992) 441–474. doi:10.1017/S0022112092002453.
URL http://journals.cambridge.org/abstract_S0022112092002453

[11] J. Kuehl, E. Perez, H. Reed, JoKHeR: NPSE Simulations of Hypersonic Crossflow Instability, 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition (January) (2012) 1–14. doi:10.2514/6.2012-921.
URL http://arc.aiaa.org/doi/abs/10.2514/6.2012-921

[12] A. J. Moyes, P. Paredes, T. S. Kocian, H. L. Reed, Secondary instability analysis of crossflow on a hypersonic yawed straight circular cone, Journal of Fluid Mechanics 812 (2017) 370–397. doi:10.1017/jfm.2016.793.
URL https://www.cambridge.org/core/product/identifier/S002211201600793X/type/journal_article

[13] Y. M. Zhang, H. Zhou, Verification of parabolized stability equations for its application to compressible boundary layers, Applied Mathematics and Mechanics (English Edition) 28 (8) (2007) 987–998. doi:10.1007/s10483-007-0801-3.

[14] K. Gudmundsson, T. Colonius, Parabolized Stability Equation Models for Turbulent Jets and Their Radiated Sound *.

[15] M. Itasse, J.-P. Brazier, O. Léon, G. Casalis, Parabolized Stability Equations analysis of nonlinear interactions with forced eigenmodes to control subsonic jet instabilities, Physics of Fluids 27 (8) (2015) 084106. doi:10.1063/1.4928472.

[16] D. Rodriguez, A. Sinha, G. A. Brès, T. Colonius, Acoustic field associated with parabolized stability equation models in turbulent jets, in: 19th AIAA/CEAS Aeroacoustics Conference, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2013. doi:10.2514/6.2013-2279.

[17] O. T. Schmidt, A. Towne, T. Colonius, A. V. G. Cavalieri, P. Jordan, G. A. Brès, Wavepackets and trapped acoustic modes in a turbulent jet: coherent structure eduction and global stability, Journal of Fluid Mechanics 825 (2017) 1153–1181. doi:10.1017/jfm.2017.407.

[18] C. Airiau, Non-parallel acoustic receptivity of a Blasius boundary layer using an adjoint approach, Flow, Turbulence and Combustion 65 (3-4) (2000) 347–367. doi:10.1023/A:1011472831831.

[19] B. Wasistho, Spatial Direct Numerical Simulation of Compressible Boundary Layer Flow, no. december, 1996.

[20] T. Herbert, Secondary instability of boundary layers, Annual Review of Fluid Mechanics 20 (1988) 487–526. doi:10.1146/annurev.fluid.20.1.487.

[21] M. R. Malik, Numerical methods for hypersonic boundary layer stability, Journal of Computational Physics 86 (2) (1990) 376–413. doi:10.1016/0021-9991(90)90106-B.

[22] F. Pinna, VESTA toolkit: a Software to Compute Transition and Stability of Boundary Layers, 43rd Fluid Dynamics Conference (2013) 1–9doi:10.2514/6.2013-2616.
URL http://arc.aiaa.org/doi/abs/10.2514/6.2013-2616

[23] M. Y. Hussaini, D. A. Kopriva, A. T. Patera, Spectral collocation methods, Applied Numerical Mathematics 5 (3) (1989) 177–208. doi:10.1016/0168-9274(89)90033-0.

[24] L. N. Trefethen, Spectral Methods in MATLAB (2000). doi:10.1137/1.9780898719598.

[25] Matthias Baer, Findiff: A Python package for finite difference derivatives in any number of dimensions (2018).
URL https://pypi.org/project/findiff/

[26] T. Herbert, Parabolized Stability Equations, Annual Review of Fluid Mechanics 29 (1) (1997) 245–283. doi:10.1146/annurev.fluid.29.1.245.

[27] J. L. v. Ingen, A suggested semi-empirical method for the calculation of the boundary layer transition region (1956). doi:V.T.H.-74.

[28] J. L. V. Ingen, Historical review of work at TU Delft, 38th Fluid Dynamics Conference and Exhibit (June) (2008) 1–49. doi:AIAA2008-3830.

[29] G. B. Schubauer, H. K. Skramstad, Laminar-boundary-layer oscillations and transition on a flat plate, Journal of Research of the National Bureau of Standards 38 (February) (1947) 251–292. doi:10.6028/jres.038.013.
URL http://hdl.handle.net/2060/19930091976