COBEM-2017-2889
IMPLEMENTATION OF A JACOBIAN-FREE METHOD FOR THE INSTABILITY ANALYSIS OF A COMPRESSIBLE FLOW OVER AN OPEN CAVITY

Marlon Sproesser Mathias
Marcello A F Medeiros
São Carlos School of Engineering, University of São Paulo - Brazil
marlon.mathias@usp.br
marcello@sc.usp.br

Vassilios Theofilis
School of Engineering, University of Liverpool - United Kingdom and PGMEC, Universidade Federal Fluminense - Brazil
v.theofilis@liverpool.ac.uk

Abstract. *An algorithm based on the Arnoldi iteration is implemented to compute the instability modes of a compressible flow over an open cavity. The flow is solved by a Direct Numerical Solver (DNS), which uses high-order spectral-like numerical differentiation methods. This allows shorter wavelengths to be resolved without further refining the mesh, reducing the computational cost. To further speed-up the execution, the code is run in parallel by a domain decomposition technique. The instability analysis is based on the Arnoldi iteration method, with is directly linked to the DNS, causing this to be a very modular and flexible strategy, as the flow solver may be replaced with only minor changes to the rest of the code. The purpose of this work is to implement this algorithm for a subsonic compressible flow. Numerical parameters of the algorithms and their trade-offs are analyzed. For example, longer run times at each DNS call cause the Arnoldi to converge faster and vice-versa. The goal is to produce reliable results while reducing memory requirements and processing time.*

Keywords: *Open cavity flow, Direct numerical simulation, Hydrodynamic instability, Compressible flow, Arnoldi iteration*

1. INTRODUCTION

In this work, a subsonic compressible flow over an open cavity is modeled by a Direct Numerical Solver (DNS) and its instability modes are retrieved by a Jacobian-free eigenproblem solver. So far, most studies have considered either an incompressible or a supersonic flow over a cavity (Theofilis, 2011).

Most modern aircraft today fly at mid to high subsonic speeds, in which compressibility plays an important role (Brès and Colonius, 2008). This work focuses on this range of Mach numbers. Both two and three-dimensional cases are considered. The span-wise direction is assumed to be periodical.

Figure 1 illustrates the situation. There is a uniform flow coming from the left-hand side, a boundary layer is formed on the flat plate and there is a rectangular cavity in this plate after a certain distance. The geometry is considered uniform and infinite in the span-wise direction, as is the plate after the cavity.

In the usual case, the flow circulates inside the cavity and a shear layer is formed on the opening. If the flow is unsteady, sound is also emitted. The main parameters to be analyzed are:

- L/D - Cavity aspect ratio (length by depth)
- θ_{LE} - Boundary layer momentum thickness at the cavity leading edge
- Re - Reynolds number
- Ma - Mach number

Depending on these parameters, the flow might be either stable or unstable. A stable flow will develop into a permanent regime, while an unstable flow will develop into an oscillating regime.

One of the most straight-forward approaches to obtain the instability modes of any system is to compute its Jacobian matrix and retrieve its eigenvalues and eigenvectors. Unfortunately, this matrix grows very fast as the mesh becomes finer or the domain increases, causing this method to quickly scale into terabytes of memory even for simple geometries.

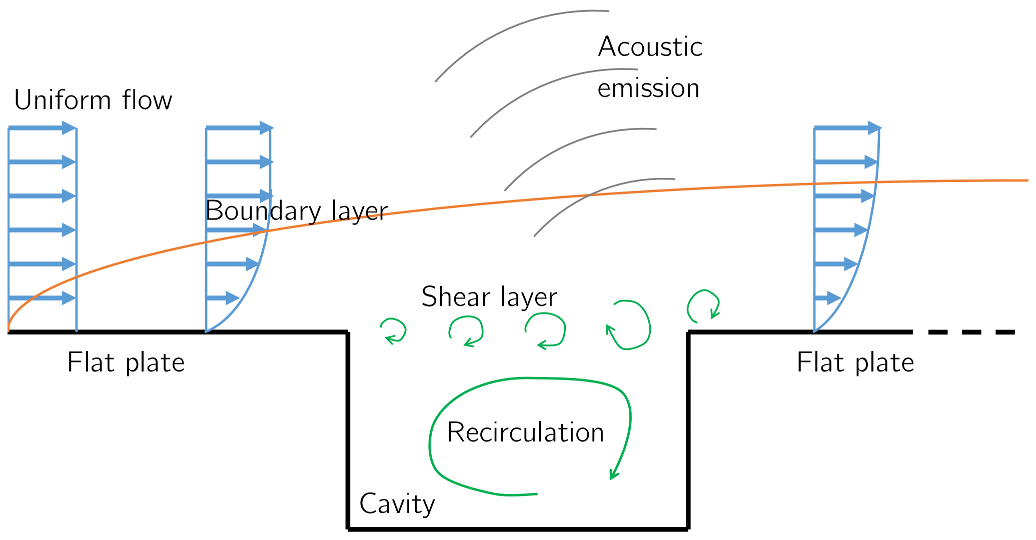

Figure 1. Illustration of the open cavity flow.

The Arnoldi iteration (Arnoldi, 1951) is an algorithm to compute the eigenvalues of large matrices based on the Krylov subspaces. One of the most interesting features of this method is that the matrix itself is not explicitly required, just the ability to compute vector multiplications with it. The system of equations can be arranged in a way that a call to the DNS code is equivalent to such matrix multiplication. Eriksson and Rizzi (1985) have implemented this method for the Euler equations. Chiba (1998) has used it for the Navier-Stokes equations. Tezuka and Suzuki (2006) have extended this analysis to three-dimensional domains. Gómez *et al.* (2014) link this algorithm to the open-source OpenFoam code to perform the flow simulation, allowing for more complex geometries.

In summary, this method takes a steady state flow as input, either stable or unstable, adds small disturbances to it and calls the DNS to run this new flow for a short time, observing the growth rate. This process is repeated with orthogonal disturbances until the desired modes are sufficiently converged by the Arnoldi iteration method. The most unstable eigenvalues are the first to converge.

The DNS is developed in house and uses high-order spectral-like numerical differentiation methods (Lele, 1992). This allows shorter wavelengths to be resolved without refining the mesh, reducing the computational cost. To further speed-up the execution, the code can be run in parallel by using both MPI and OpenMP (Martinez and Medeiros, 2016).

2. METHODS

2.1 Flow simulation

The direct numerical simulation code was developed by a hydrodynamics instability research group in the São Carlos School of Engineering, University of São Paulo, for several uses, and was adapted for the open cavity flow. It is written in FORTRAN 90 language and uses a domain decomposition technique along with MPI and OpenMP for parallelization. (Martinez and Medeiros, 2016)

The compressible Navier-Stokes equations are used to model the flow, along with mass and energy conservation equations. A compressible flow can be entirely defined by five variables: density (ρ), internal energy (e) and the three velocity components (u,v,w), in this case. Each variable depends on location (x,y,z) and time (t). Equations are written in the non-conservative form, which solves for each individual variable mentioned above, simplifying the boundary conditions.

All values are non-dimensional, being normalized by the free flow speed, cavity depth and initial density. Reynolds, Prandtl and Mach numbers are given by:

$$Re = \frac{\rho^*_\infty \, U^*_\infty \, L_0}{\mu^*_\infty}, \quad Pr = \frac{\mu^*_\infty c^*_p}{k^*}, \quad M_\infty = \frac{U^*_\infty}{\sqrt{\gamma \frac{p^*_\infty}{\rho^*_\infty}}} \tag{1}$$

The code uses a structured mesh in a rectangular domain. The grid is much finer in the wall-normal direction close to the flat plate and the cavity. There is also stretching in the stream-wise direction, concentrating nodes on the cavity.

Both two-dimensional and three-dimensional flows can be run in this analysis. In a 3D case, the span-wise direction is considered periodical. In this implementation, a single wave number in this direction is used each time, allowing for smaller meshes, considerably reducing the computational cost.

The inflow boundary is defined as a uniform flow at constant temperature and a Neumann condition is placed for the pressure so that its derivative is null in the flow direction. In the outflow, pressure is kept constant and Neumann homogeneous conditions are used for temperature and velocities.

Walls have no-slip and no-penetration conditions for velocity, the pressure gradient is set to zero in the normal direction and the temperature is fixed. Just downstream of the inflow, there is a free-slip region in the wall, it is necessary to accommodate the flow before the boundary layer starts forming.

It is worth noting that pressure and temperature are not directly available as variables in the code and they should be computed from the density and energy values. The temperature depends solely on the energy and pressure depends on both density and energy. This way, the boundary conditions routine first changes the energy to observe the temperature conditions and, later, the density to keep the pressure conditions.

The two top corners on the cavity were points of particular attention for the boundary conditions. The Neumann homogeneous condition for pressure in other points is observed by computing the density at each point of the boundary so that the pressure's derivative is null in the normal direction on the wall. Both corner nodes present a problem as walls in distinct directions meet and the density that observes the Neumann condition in one direction might violate it for the other. The pressure in both nodes is set to the mean of the pressures that meet the boundary condition for each direction.

A buffer-zone is placed in each of the open boundaries. It presents an stretched mesh and a Selective Frequency Damping filter to damp out higher oscillation frequencies (Akervik *et al.*, 2006). This region is needed to avoid wave reflections back to the domain.

The initial condition is defined as free-flow velocity above the flat plate and zero velocity in the cavity, at constant pressure and temperatures. Results from previous runs can also be used as initial condition as long as they share the same mesh; Reynolds and Mach numbers and the numerical methods may be changed.

Spatial derivatives are computed by a finite differences method chosen before runtime. Explicit second and fourth order methods are implemented, as well as compact fourth order spectral-like and sixth order schemes described by Lele (1992). It is possible to use a distinct differentiation method for the buffer zone, usually of a lower order, the transition from one method to the next happens smoothly.

Time integration is performed at a fixed time step by a forth order Runge-Kutta scheme, with allows larger time steps when compared to lower order methods due to its excellent numerical stability properties. The boundary conditions routine is called after each sub-step of the method.

In this type of simulation, due to the non-dissipative methods employed, high-frequency numerical noise may quickly build up, causing the simulation to diverge from the physical solution. A numerical low-pass filter attenuates this noise (Gaitonde and Visbal, 1998). The filter strength can be adjusted before runtime and, ideally, should be as low as possible, minimizing its effect on results. Filtering was turned off close to the boundaries due to the large derivatives present.

For the parallel execution, the domain is decomposed in slices and each one is sent to a different process with the use of MPI. Each slice can be further parallelized by using OpenMP, which allows multiple iterations of loops to be run concurrently.

2.2 Instability Analysis

In order to obtain the oscillation modes of a certain flow, one has to compute the eigenvalues and eigenvectors of its Jacobian matrix. It is, given that the Navier-Stokes equations and all boundary conditions are described by a function $\boldsymbol{f}$ such that:

$$\frac{\partial \boldsymbol{U}}{\partial t} = \boldsymbol{f}(\boldsymbol{U}) \tag{2}$$

Where $\boldsymbol{U}$ is a vector that contains all flow variables for each node in the mesh, the Jacobian is given by

$$\boldsymbol{A} = \frac{\partial \boldsymbol{f}(\boldsymbol{U})}{\partial \boldsymbol{U}} \tag{3}$$

It can be used to linearize the Navier-Stokes around a given permanent base flow $\boldsymbol{U}_0$. The flow $\boldsymbol{U}$, in this case, is given by the base flow plus a disturbance: $\boldsymbol{U} = \boldsymbol{U}_0 + \boldsymbol{u}$. After linearizing, the time evolution of $\boldsymbol{u}$ is given, analytically, by

$$\boldsymbol{u}_t = e^{t\boldsymbol{A}}\boldsymbol{u}_0 \tag{4}$$

u_0 and u_t represent the flow state at the initial condition and at time t, respectively.

By computing the eigenvalues $\boldsymbol{L}$ and eigenvectors $\boldsymbol{V}$ of $\boldsymbol{A}$, it can be rewritten as

$$\boldsymbol{u}_t = \boldsymbol{V}e^{t\boldsymbol{L}}\boldsymbol{V}^{-1}\boldsymbol{u}_0 \tag{5}$$

By solving the eigenproblem, one can obtain the flow modes and their respective amplification rates and oscillation frequencies. Each column $\boldsymbol{v}_i$ of $\boldsymbol{V}$ corresponds to a flow mode and is associated to a value σ_i in $\boldsymbol{L}$.

The real part of σ_i represents the amplification rate of each mode. Negative values mean that the mode is stable and will be attenuated; positive values relate to an unstable mode, that will be amplified. In a perfectly linear system, this amplification would happen indefinitely; in real systems, after reaching a certain amplitude, non-linear effects would appear and cause it to saturate.

An interesting way of looking at Eq. 5 is to read it from right to left. The last two terms, $\boldsymbol{V}^{-1}\boldsymbol{u}_0$, compute how much of each mode is present in the initial disturbance. The term $e^{t\boldsymbol{L}}$ shows how much each mode is amplified or attenuated through time. Thus, $e^{t\boldsymbol{L}}\boldsymbol{V}^{-1}\boldsymbol{u}_0$ means how much of each mode is present in the flow at any given time. Finally, by multiplying it by $\boldsymbol{V}$, the modes are combined and transformed into the actual flow.

The size of the Jacobian matrix is $4N\times4N$ for two-dimensional flows and $5N\times5N$ for three-dimensional. 4 and 5 are the numbers of variables for each node in a compressible flow, respectively. N being the number of nodes in the domain.

The mesh size is usually in the order of hundreds of points in both the stream-wise and wall-normal directions and a few dozens of points in the span-wise direction. Therefore, the total number of variables to solve for is in the order of 10^5 to 10^6 in two-dimensional cases and 10^6 to 10^7 when accounting for the third dimension.

Due to the implicit differentiation methods used, in which each node in the domain influences every other at all steps, the Jacobian of this system would be a full matrix. At the double precision used by this code, which takes 8 bytes of memory per scalar value, simply storing the Jacobian for a system with 10^6 variables, for example, would take 8×10^{12} bytes, or 8 terabytes of memory.

Because of this, it is not feasible to directly compute the Jacobian matrix. It is also not necessary, as the vast majority of its modes are very stable and not in the scope of this work, only some of the most unstable modes are needed.

The Arnoldi iteration is an algorithm to obtain the eigenvalues and eigenvectors of a system iteratively, it has the very interesting property of not explicitly needing the matrix in order to solve the eigenproblem, all it needs is the capability of multiplying vectors by it. (Arnoldi, 1951)

As implemented, this method has the memory usage scale linearly with the mesh size, instead of quadratically. For the same system with 10^6 variables, if 1000 iterations are used, 8×10^9 bytes, or 8 gigabytes, of memory would be needed for the largest matrix.

The concept behind this method is that by repeatedly multiplying a matrix $\boldsymbol{M}$ by a vector $\boldsymbol{v}$, the eigenvectors of $\boldsymbol{M}$ that are present in $\boldsymbol{v}$ will be filtered depending on their eigenvalues.

After enough iterations, only the eigenvectors corresponding to eigenvalues with the greatest absolute value will remain.

This simple algorithm, known as the power iteration, would be capable of finding a single eigenvector and would take a considerable amount of iterations to converge.

The Arnoldi iteration differs in a way that, after each multiplication, the resulting vector is projected to be orthonormal to all previous vectors before being multiplied again. This allows multiple eigenvectors to be retrieved. Eigenvalues with the greatest modules tend to converge with fewer iterations.

This method was proposed and implemented by Eriksson and Rizzi (1985) for the Euler equations, Edwards *et al.* (1994) and Chiba (1998) are among the first to implement it for the incompressible Navier-Stokes equations. It is based on Arnoldi's method for solving the eigenproblem. The DNS itself is embedded into the method instead of the Jacobian matrix.

A matrix $\boldsymbol{B}$ is defined as $e^{t\boldsymbol{A}}$ from Eq. 4, thus, the system becomes

$$\boldsymbol{u}_t = \boldsymbol{B}\boldsymbol{u}_0 \tag{6}$$

The Arnoldi iteration is used to obtain the eigenvalues and eigenvectors of $\boldsymbol{B}$, which can be easily related to those of $\boldsymbol{A}$.

Numerically integrating $\boldsymbol{f}$ from Eq. 2 for a time t is equivalent to multiplying the initial flow by $\boldsymbol{B}$, excluding the non-linear terms. It is important to note that the linearized system operates on the disturbance $\boldsymbol{u}$ around the equilibrium point $\boldsymbol{U}_0$, while $\boldsymbol{f}$ operates on the whole flow $\boldsymbol{U} = \boldsymbol{U}_0 + \boldsymbol{u}$.

$$\boldsymbol{U}_0 + \boldsymbol{u}_t = \boldsymbol{U}_0 + \boldsymbol{B}\boldsymbol{u}_0 = \boldsymbol{U_0} + \int_0^t \boldsymbol{f}\left(\boldsymbol{U}_0 + \boldsymbol{u}_0\right) dt - N.L.T. \tag{7}$$

This is only valid if the base flow $\boldsymbol{U}_0$ is an equilibrium point, it is, $\boldsymbol{f}(\boldsymbol{U}_0) = \boldsymbol{0}$. Which, given the numerical nature of this work, may not be completely true, as there are usually some residuals involved when computing the base flow.

To overcome this problem, as well as to reduce the errors from non-linear terms, two numerical integrations are performed, with symmetrical disturbances, as shown in the following equations. To simplify the notation, from now on the integral of $\boldsymbol{f}$ from 0 to t will be written as $\boldsymbol{F}$, which is equivalent to a DNS call.

Equation 7 is rewritten as:

$$\boldsymbol{U}_0 + \boldsymbol{B}\boldsymbol{u}_0 \approx \boldsymbol{U_0} + \boldsymbol{F}\left(\boldsymbol{U}_0 + \boldsymbol{u}_0\right) \tag{8}$$

Analogically, by using the opposite disturbance:

$$\boldsymbol{U}_0 - \boldsymbol{B}\boldsymbol{u}_0 \approx \boldsymbol{U_0} + \boldsymbol{F}\left(\boldsymbol{U}_0 - \boldsymbol{u}_0\right) \tag{9}$$

Therefore, subtracting Eq. 9 from Eq. 8:

$$2\boldsymbol{B}\boldsymbol{u}_0 \approx \boldsymbol{F}\left(\boldsymbol{U}_0 + \boldsymbol{u}_0\right) - \boldsymbol{F}\left(\boldsymbol{U}_0 - \boldsymbol{u}_0\right) \tag{10}$$

In this analysis, the initial disturbance u_0 is given by a normalized vector $\boldsymbol{\zeta}$ multiplied by a scaling factor ε. Equation 10 becomes:

$$2\varepsilon\boldsymbol{B}\boldsymbol{\zeta} \approx \boldsymbol{F}\left(\boldsymbol{U}_0 + \varepsilon\boldsymbol{\zeta}\right) - \boldsymbol{F}\left(\boldsymbol{U}_0 - \varepsilon\boldsymbol{\zeta}\right) \tag{11}$$

$$\boldsymbol{B}\boldsymbol{\zeta} \approx \frac{1}{2\varepsilon}\left(\boldsymbol{F}\left(\boldsymbol{U}_0 + \varepsilon\boldsymbol{\zeta}\right) - \boldsymbol{F}\left(\boldsymbol{U}_0 - \varepsilon\boldsymbol{\zeta}\right)\right) \tag{12}$$

Small values of ε reduce the non-linear terms, providing better approximations in terms of the truncation error, but are likely to increase the round-off error caused by the computer's precision, which is already critical in this situation, as the disturbances are orders of magnitude smaller than the base flow.

Eriksson and Rizzi (1985) have noted that the error in Eq. 12 scales with ε^2 and developed a similar equation, but with fourth order accuracy:

$$\boldsymbol{B}\boldsymbol{\zeta} \approx \frac{1}{12\varepsilon}\left(-\boldsymbol{F}\left(\boldsymbol{U}_0 + 2\varepsilon\boldsymbol{\zeta}\right) + 8\boldsymbol{F}\left(\boldsymbol{U}_0 + \varepsilon\boldsymbol{\zeta}\right) - 8\boldsymbol{F}\left(\boldsymbol{U}_0 - \varepsilon\boldsymbol{\zeta}\right) + \boldsymbol{F}\left(\boldsymbol{U}_0 - 2\varepsilon\boldsymbol{\zeta}\right)\right) \tag{13}$$

Note that Eq. 13 requires four DNS calls for the numerical integration, instead of just two.

Both Eqs. 12 and 13 can be used by the Arnoldi iteration as the form to obtain $\boldsymbol{B\zeta}$. In this work, only the former is used, as it nearly halves the computational cost. Tezuka and Suzuki (2006) have also used this lower order method and noted that the difference in results from ε = 0.01 to ε = 1 is negligible, leading to the conclusion that even at this lower order of accuracy, the truncation error is very small.

After the iterations are concluded, the eigenvalues and eigenvectors of $\boldsymbol{A}$ are finally computed. Arnoldi's method converges faster on the eigenvalues that are further away from the origin of the complex plane, which, after the exponential transform, translate into the most unstable modes of the flow.

In summary, the method adds small disturbances to the steady-state flow and uses the DNS to observe how the flow reacts to them. After a series of orthogonal disturbances, the Hessenberg matrix in Arnoldi's method is obtained, which is several orders of magnitude smaller than the flow's Jacobian matrix would be. The eigenvalues and eigenvectors of this matrix are computed and, finally, used to approximate the flow's modes and their respective amplification or attenuation rates. This is shown in the flowchart in Fig. 2.

In step (3) of Fig. 2, ε controls the perturbation's norm, as noted above. It is worth noting that the number of mesh nodes influences the perturbation's module in each one. In this work, to overcome this issue, the perturbation norm is defined as a fixed parameter multiplied by the square root of the number of nodes in the mesh: $\varepsilon = \varepsilon_0 \sqrt{N}$. This is done so the RMS of the disturbance is kept constant for any number of variables in the system.

If higher order schemes are used, as Eriksson and Rizzi (1985) have noted, step (5) should be changed accordingly and previous steps should include the extra DNS runs required.

It should also be noted that in step (12), the equation for eigenvalues is not fully reversible for the imaginary part, as the exponential function of a complex number is not injective. This means that the method brings a correct approximation only for the real part of the eigenvalues, related to the amplification rate.

The imaginary part, related to the frequency, must be checked by other means if t is too large, for example by using the DNS to simulate each of the modes identified and observing the oscillation frequencies. The difference between the imaginary part found by the code and the actual value is an integer multiplied by $2\pi/t$.

3. RESULTS

3.1 Code performance

For the code performance analysis, a sample case was created. Its mesh has a total of 380 nodes in the stream-wise direction, 170 nodes in the wall-normal direction and 10 nodes in the span-wise direction, including both the physical domain and the buffer zones.

The 4th order compact spectral-like finite differences were used for the spatial derivatives. The 4th order Runge-Kutta method was used for time integration. Numerical filtering was turned on for all directions at every step. Selective Frequency Damping was turned on only at the buffer zones.

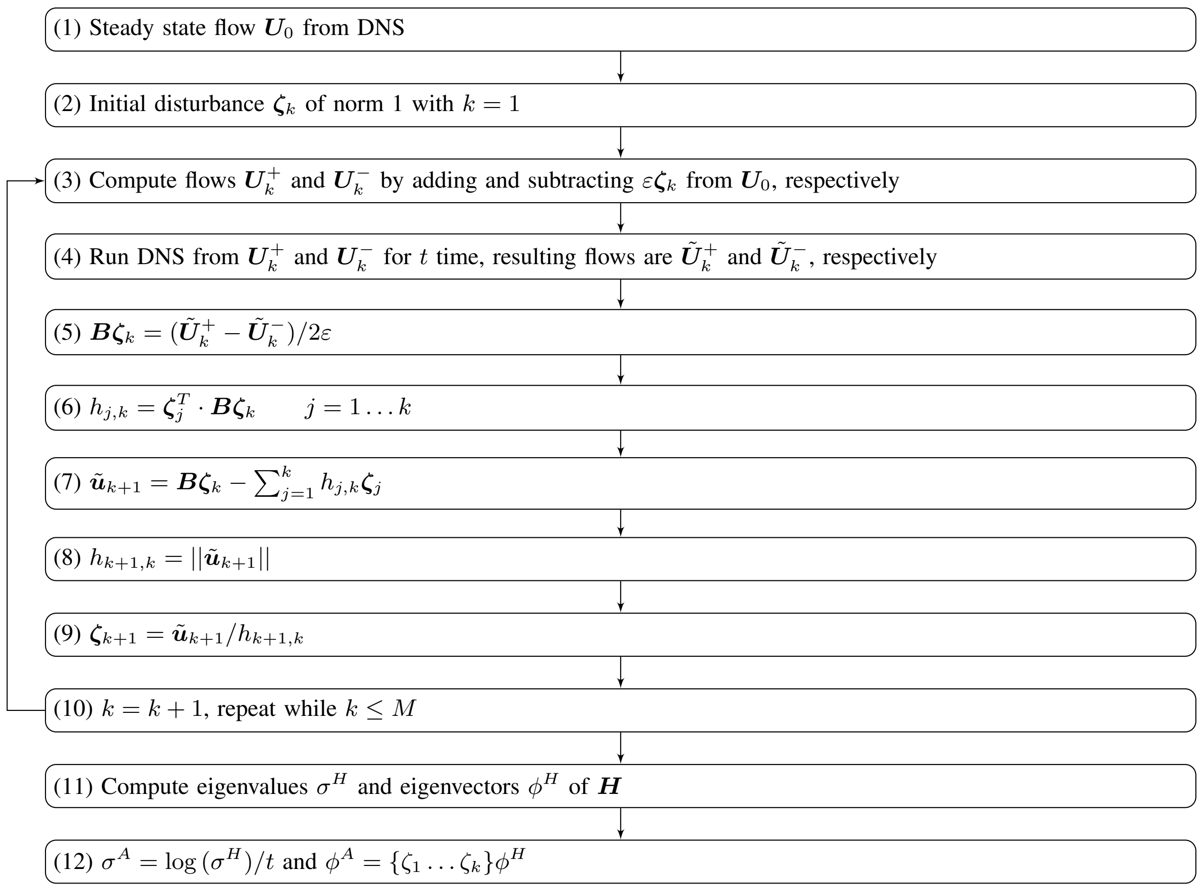
(1) Steady state flow $\boldsymbol{U}_0$ from DNS
(2) Initial disturbance $\boldsymbol{\zeta}_k$ of norm 1 with $k = 1$
(3) Compute flows $\boldsymbol{U}_k^+$ and $\boldsymbol{U}_k^-$ by adding and subtracting $\varepsilon\boldsymbol{\zeta}_k$ from $\boldsymbol{U}_0$, respectively
(4) Run DNS from $\boldsymbol{U}_k^+$ and $\boldsymbol{U}_k^-$ for t time, resulting flows are $\tilde{\boldsymbol{U}}_k^+$ and $\tilde{\boldsymbol{U}}_k^-$, respectively
(5) $\boldsymbol{B}\boldsymbol{\zeta}_k = (\tilde{\boldsymbol{U}}_k^+ - \tilde{\boldsymbol{U}}_k^-)/2\varepsilon$
(6) $h_{j,k} = \boldsymbol{\zeta}_j^T \cdot \boldsymbol{B}\boldsymbol{\zeta}_k \qquad j = 1 \dots k$
(7) $\tilde{\boldsymbol{u}}_{k+1} = \boldsymbol{B}\boldsymbol{\zeta}_k - \sum_{j=1}^{k} h_{j,k}\boldsymbol{\zeta}_j$
(8) $h_{k+1,k} = ||\tilde{\boldsymbol{u}}_{k+1}||$
(9) $\boldsymbol{\zeta}_{k+1} = \tilde{\boldsymbol{u}}_{k+1}/h_{k+1,k}$
(10) $k = k + 1$, repeat while $k \leq M$
(11) Compute eigenvalues σ^H and eigenvectors ϕ^H of $\boldsymbol{H}$
(12) $\sigma^A = \log(\sigma^H)/t$ and $\phi^A = \{\zeta_1 \dots \zeta_k\}\phi^H$

Figure 2. Flowchart of the Jacobian-free method for computing the flow modes.

A server running the Ubuntu 14.04 operating system was used. It features 4 Intel® Xeon® E7-4820 v3 processors, each with 10 cores at 1.90 GHz. A total of 128 GB of DDR3 RAM is available.

The Intel® FORTRAN Compiler is used in all cases.

For the two-dimensional performance analysis, the code was run for 1000 time steps. About 60 MB of RAM were used for the simulation. Table 1 brings the results for different types of parallel execution.

In the three-dimensional case, the code was run for 100 steps, using about 600 MB of RAM. The results are in Tab. 2.

Table 1. Computation time for the two-dimensional case.

Domain slices	OpenMP workers	Total threads	Runtime (s)	Time per node per step (s)	Speed up	Efficiency
1	1	1	288.9	4.47e-6	-	-
2	1	2	144.9	2.24e-6	99%	99%
1	4	4	122.4	1.89e-6	136%	59%
2	4	8	67.5	1.05e-6	328%	53%

Table 2. Computation time for the three-dimensional case.

Domain slices	OpenMP workers	Total threads	Runtime (s)	Time per node per step (s)	Speed up	Efficiency
1	1	1	622.7	9.64e-6	-	-
2	1	2	328.2	5.08e-6	89%	95%
4	1	4	169.6	2.63e-6	267%	92%
8	1	8	93.8	1.45e-6	564%	83%

It can be noted that the parallel execution by slicing the domain and using MPI is much more efficient than simply increasing the number of OpenMP workers.

During the instability analysis, the vast majority of the computational time is spent by waiting for the DNS to execute. The main concern of this routine is memory usage as it must store multiple states of the domain at once in the ζ matrix of Fig. 2.

This matrix has each value stored as a double, which takes 8 bytes of memory. Therefore, its size is given by the amount of nodes in the domain times the number fo variables times the number of Arnoldi iterations times 8 bytes.

In this 380 $\times$ 170 $\times$ 10 domain, for example, this matrix would use 2.6 GB of RAM if 1000 Arnoldi iterations are performed.

4. Code validation

The validation in this work is divided into three parts. First, the DNS is checked for mesh and domain convergence by using an unsteady flow described by Colonius *et al.* (1999).

Then, the methods parameters are checked for convergence, such as number of Arnoldi iterations and length of the time span run at each Arnoldi iteration for a 2D flow based on de Vicente *et al.* (2014).

Finally, the instability analysis is validated by comparing the modes found by the Arnoldi algorithm to the ones retrieved by the Residual Algorithm (Theofilis and Colonius, 2003).

The differentiation scheme used is the 4th compact spectral-like finite differences. Time stepping is done by the 4th order Runge-Kutta method. Buffer zones are placed on all open boundaries and have the finite differences changed to a explicit second order scheme, SFD is also turned on at these regions.

Figure 3 shows the wall-normal velocity plotted against time at a fixed point at the flat plate height, three quarters across the opening. The velocity obtained by the two meshes is plotted along with the reference data extracted from the paper by Colonius *et al.* (1999). Note that the phases were manually adjusted. Mesh 1 is 300 by 150 nodes (stream-wise and wall-normal directions, respectively), Mesh 2 is 400 by 200 nodes. The amount of nodes in the cavity for each mesh are 144 by 52 and 192 by 70, respectively.

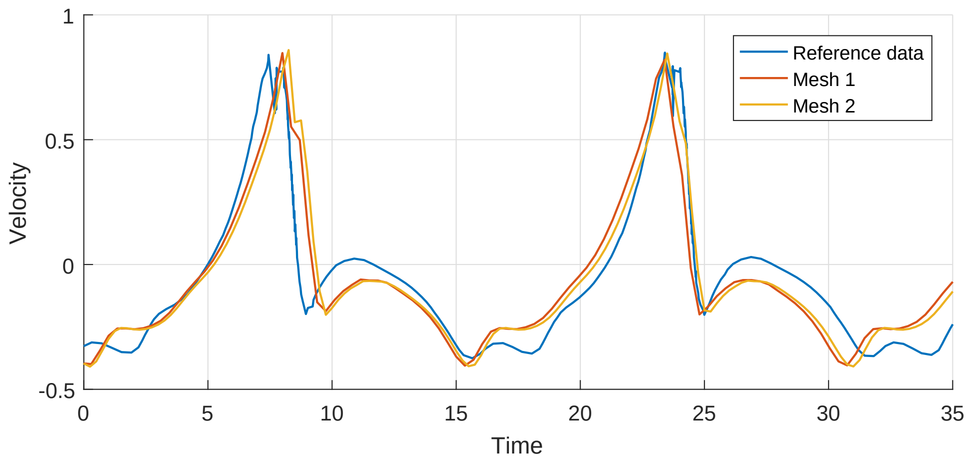

Figure 3. Velocity plotted against time at a fixed point in an unsteady case.

Both meshes have agreed on the same velocity profile, which closely matches the frequency and the wave form of the reference.

To validate the instability analysis, a case described by de Vicente *et al.* (2014) was used.

In this case, the cavity's aspect ratio is $L/D = 2$, the Reynolds number is $Re_D = 1149$ and the boundary layer thickness at the cavity's leading edge is $\theta = 0.0337$. As before, all lengths are normalized by the cavity depth and all velocities, by the inflow velocity.

The reference case has assumed the flow as incompressible. To approximate this assumption, the Mach number was set as $Ma = 0.1$. Low Mach numbers require a shorter time step to maintain numerical stability, which would considerably increase the computational cost.

A mesh containing 300 by 150 nodes was created for this case. It was deemed enough after a mesh and domain convergence analysis. The mesh inside the cavity is 114 by 52.

The algorithm was also run for 200, 400 and 1000 steps at each Arnoldi iteration. Fig. 4 shows the complex plane for these cases. All cases were run for 400 Arnoldi iterations, enough to converge the 12 least stable modes.

It can be seen that the $nT = 200$ case is not converged, as it has resulted in a different set of eigenvalues. The $nT = 400$

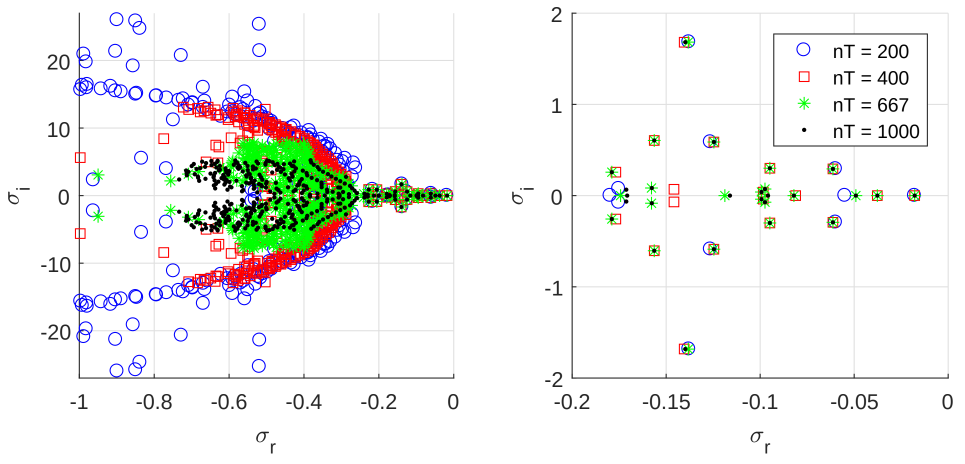

Figure 4. Eigenvalues computed for the convergence analysis of the number of DNS steps.

case diverges at the 3rd mode. The other two results match, usually up to the 8th decimal place for the first few eigenvalues. From the 13th mode onwards, the nT = 667 case also starts diverging.

By plotting the eigenfunctions in Fig. 5, it can be seen that, as in the ε_0 convergence analysis, despite correctly computing the first eigenvalue up to 4 decimal places, the nT = 200 case has failed to find the corresponding eigenfunction.

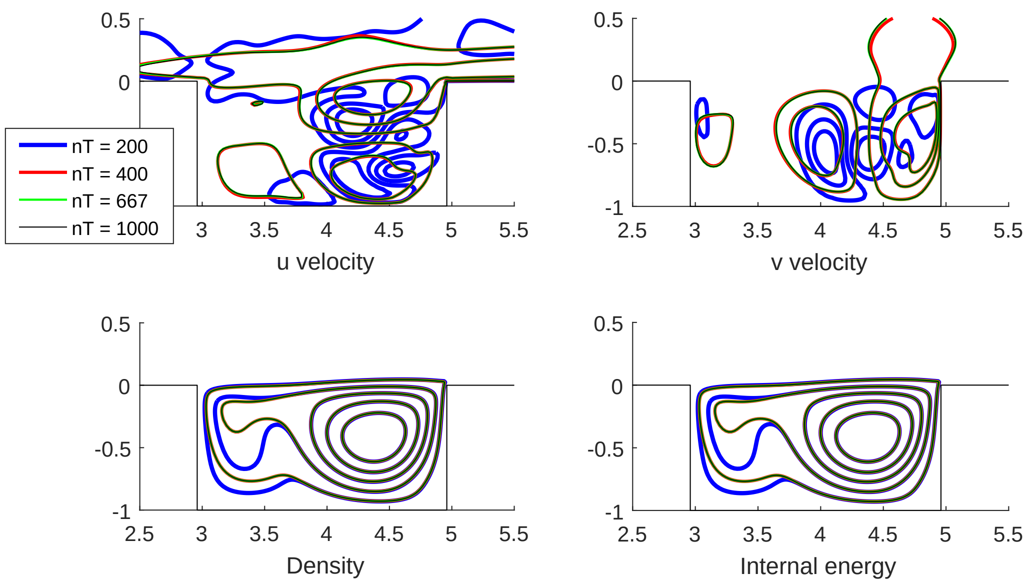

Figure 5. Isocountours of mode 1 eigenfunctions for the convergence analysis of the number of DNS steps.

It is interesting to note how the range of values for σ_i reduces as the number of steps is increased. This is due to the last equation in the flowchart of Fig 2. This means that smaller values of nT have the advantage of being able to resolve a wider spectrum of modes.

This case has a particularly short time step for the DNS due to the low Mach number. At higher Mach numbers, the amount of time steps can be significantly reduced due to the increased step length.

The residual algorithm (RA). was used to make sure the eigenvalues and eigenfunctions found by the Arnoldi iterations matched the ones present in the code. As mentioned earlier, this algorithm is only capable of retrieving the least stable mode. (Theofilis and Colonius, 2003)

The time series from the uniform initial condition towards the converged base flow was used as input for the residual algorithm.

35 probes were placed in the domain, both inside and outside the cavity. They stored the value of each variable every 1000 time steps. Data from time steps 1 to 2×10^6 was used. The initial condition for this run was a uniform flow outside

the cavity.

Figure 6 shows the results obtained by the well-placed probes for each time sample, as well as the mean value and standard deviation.

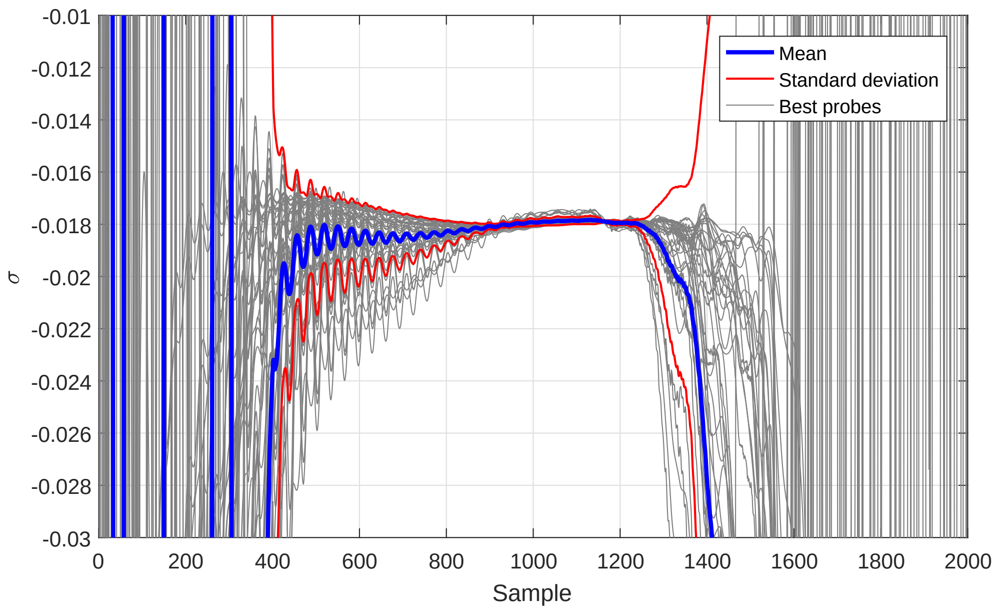

Figure 6. Least stable eigenvalue as computed by the residual algorithm.

This algorithm has computed the eigenvalue to be σ = -0.0179. Adding and subtracting one standard deviation, it is somewhere between -0.0178 and -0.0180. This matches the Arnoldi method, which has computed this value to be σ = -0.01781. These values were extracted from samples 1000 to 1200, just before the numerical noise started growing.

5. DISCUSSION AND CONCLUSION

The tool developed in this work is able to access the instability modes of open cavity flows with relative ease. It takes from some hours to a day to generate a base flow. For a two-dimensional case, a few hours are needed for the instability analysis, while a three-dimensional analysis may take a few days. The same base-flow may be used for both two and three-dimensional cases.

One advantage of this method is that the DNS could be easily replaced to represent other types of geometries and minimal recoding would be needed in the instability analysis routines.

This tool is intended to be used on a sweep of various Mach numbers and boundary layer thicknesses in order to better understand how these parameters affect the stability. Rossiter-like modes were observed at higher Mach numbers, such as the ones shown by Yamouni *et al.* (2013).

Data from the resulting eigenfunctions can be used to understand the physical phenomena behind the effects of both these parameters to the flow modes.

6. ACKNOWLEDGMENTS

The authors would like to thank the Brazilian National Council for Scientific and Technological Development (CNPq) for their financial support (Grants 134722/2016-7 and 304859/2016-8).

7. REFERENCES

Akervik, E., Brandt, L., Henningson, D.S., Hœpffner, J., Marxen, O. and Schlatter, P., 2006. "Steady solutions of the Navier-Stokes equations by selective frequency damping". *Physics of Fluids*, Vol. 18, No. 6, p. 068102. ISSN 10706631. doi:10.1063/1.2211705. URL `http://scitation.aip.org/content/aip/journal/pof2/18/6/10.1063/1.2211705`.

Arnoldi, W.E., 1951. "The principle of minimized iterations in the solution of the matrix eigenvalue problem". *Quarterly of Applied Mathematics*, Vol. 9, No. 1, pp. 17–29.

Brès, G.A. and Colonius, T., 2008. "Three-dimensional instabilities in compressible flow over open cavities". *Journal of Fluid Mechanics*, Vol. 599. ISSN 0022-1120. doi:10.1017/S0022112007009925. URL `http://www.journals.cambridge.org/abstract_S0022112007009925`.

Chiba, S., 1998. "Global Stability Analysis of Incompressible Viscous Flow". *Journal of Japan Society of Computational Fluid Dynamics*, Vol. 7, No. 1, pp. 20–48.

Colonius, T., Basu, A.J. and Rowley, C.W., 1999. "Computation of sound generation and flow-acoustic instabilities in the flow past an open cavity". In *Proceedings of the Joint Fluids Engineering Conference*. San Francisco, USA.

de Vicente, J., Basley, J., Meseguer-Garrido, F., Soria, J. and Theofilis, V., 2014. "Three-dimensional instabilities over a rectangular open cavity: from linear stability analysis to experimentation". *Journal of Fluid Mechanics*, Vol. 748, pp. 189–220. ISSN 0022-1120. doi:10.1017/jfm.2014.126.

Edwards, W.S., Tuckerman, L.S., Friesner, R.A. and Sorensen, D.C., 1994. "Krylov Methods for the Incompressible Navier-Stokes Equations". *Journal of Computational Physics*, Vol. 110, No. 1, pp. 82–102. ISSN 00219991. doi: 10.1006/jcph.1994.1007.

Eriksson, L.E. and Rizzi, A., 1985. "Computer-aided analysis of the convergence to steady state of discrete approximations to the euler equations". *Journal of Computational Physics*, Vol. 57, No. 1, pp. 90–128. ISSN 10902716. doi:10.1016/0021-9991(85)90054-3.

Gaitonde, D.V. and Visbal, M.R., 1998. "High-Order Schemes for Navier-Stokes Equations: Algorithm and Implementation Into FDL3DI". Technical report, Wright-Patterson Air Force Base.

Gómez, F., Gómez, R. and Theofilis, V., 2014. "On three-dimensional global linear instability analysis of flows with standard aerodynamics codes". *Aerospace Science and Technology*, Vol. 32, No. 1, pp. 223–234. ISSN 12709638. doi:10.1016/j.ast.2013.10.006. URL `http://dx.doi.org/10.1016/j.ast.2013.10.006` `http://linkinghub.elsevier.com/retrieve/pii/S1270963813001879`.

Lele, S.K., 1992. "Compact finite difference schemes with spectral-like resolution". *Journal of Computational Physics*, Vol. 103, No. 1, pp. 16–42. ISSN 00219991. doi:10.1016/0021-9991(92)90324-R.

Martinez, A. and Medeiros, M.F., 2016. "Direct numerical simulation of a wavepacket in a boundary layer at Mach 0.9". In *46th AIAA Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, Reston, Virginia, Vol. 414, pp. 1–33. ISBN 978-1-62410-436-7. ISSN 00221120. doi:10.2514/6.2016-3195. URL `http://arc.aiaa.org/doi/10.2514/6.2016-3195`.

Tezuka, A. and Suzuki, K., 2006. "Three-dimensional global linear stability analysis of flow around a spheroid". *AIAA journal*, Vol. 44, No. 8, pp. 1697–1708. ISSN 0001-1452. doi:10.2514/1.16632. URL `http://arc.aiaa.org/doi/pdf/10.2514/1.16632`.

Theofilis, V. and Colonius, T., 2003. "An Algorithm for the Recovery of 2- and 3D BiGlobal Instabilities of Compressible Flow Over 2D Open Cavities". In *33rd AIAA Fluid Dynamics Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Reston, Virigina. ISBN 978-1-62410-095-6. doi:10.2514/6.2003-4143. URL `http://arc.aiaa.org/doi/10.2514/6.2003-4143`.

Theofilis, V., 2011. "Global Linear Instability". *Annual Review of Fluid Mechanics*, Vol. 43, No. 1, pp. 319–352. ISSN 0066-4189. doi:10.1146/annurev-fluid-122109-160705. URL `http://www.annualreviews.org/doi/suppl/10.1146/annurev-fluid-122109-160705`.

Yamouni, S., Sipp, D. and Jacquin, L., 2013. "Interaction between feedback aeroacoustic and acoustic resonance mechanisms in a cavity flow: a global stability analysis". *Journal of Fluid Mechanics*, Vol. 717, pp. 134–165. ISSN 0022-1120. doi:10.1017/jfm.2012.563.

8. RESPONSIBILITY NOTICE