Documentation

A Galerkin method with Chandrasekhar functions for the Orr-Sommerfeld/Squire initial-value problem

Federico Fraternale

Department of Mechanical and Aerospace Engineering
Politecnico di Torino, Torino, Italy 10129

December 20, 2015

Contents

1 A Chandrasekhar eigenfunction expansion method to solve the linearized initial value problem

In this section, the mathematical formulation for a Galerkin method - based on an eigenfunctions expansion in terms of the Chandrasekhar functions [1] - to solve the Orr-Sommerfeld and Squire initial value problem, is presented. The method is valid for every parallel base flow with homogeneous (no-slip) boundary conditions for the perturbations. This is a exention to the three-dimensional case and to the *non modal* problem, of the method by Gallagher & Mercer [3] which was based on the use of Chandrasekhar's functions to solve the bidimensional *modal* perturbative problem for the Couette flow.

The IVP in the wall-normal velocity and vorticity (Orr-Sommerfeld/Squire non-modal problem) is considered:

$$\begin{aligned}\partial_t\partial_y^2\hat{v} - k^2\partial_t\hat{v} + i\alpha U(y)\partial_y^2\hat{v} - i\alpha k^2 U(y)\hat{v} - i\alpha U''(y)\hat{v} \\ -\frac{1}{Re}\left(\partial_y^4\hat{v} - 2k^2\partial_y^2\hat{v} + k^4\hat{v}\right) = 0 \qquad (1)\end{aligned}$$

$$\partial_t\hat{\omega}_y + i\alpha U(y)\hat{\omega}_y - \frac{1}{Re}\left(\partial_y^2\hat{\omega}_y - k^2\hat{\omega}_y\right) = -i\gamma U(y)'\hat{v} \qquad (2)$$

$$\hat{v}(y=\pm1,t) = \partial_y\hat{v}(y=\pm1,t) = \hat{\omega}_y(y=\pm1,t) = 0 \qquad (3)$$

$$\hat{v}(y,t=0) = \hat{v}_0(y) \qquad \hat{\omega}_y(y,t=0) = \hat{\omega}_{y0}(y) \qquad (4)$$

1.1 *Solution to $\hat{v}$ equation*

The solution of (1) can be expressed as a generalized Fourier expansion with time-dependent coefficients:

$$\hat{v}(y,t) = \sum_{n=1}^{\infty} c_n(t)X_n(y) \qquad y\in[-1,1], \qquad (5)$$

where $X_n(y)$ are orthogonal functions, and the following inverse transform applies:

$$c_n(t) = \frac{\int_{-1}^{1}\hat{v}(y,t)X_n(y)\,\mathrm{d}y}{\int_{-1}^{1}X_n(y)X_n(y)\,\mathrm{d}y}. \qquad (6)$$

Since in the initial value problem both the initial condition and the boundary conditions need to be imposed, it is worthwhile to consider functions that satisfy the boundary conditions. Moreover note that the coefficients c_n of the series are in general complex, since $\hat{v}$ is complex-valued and the spatial modes are considered as real. The particular orthogonal functions which we use are those defined by the following fourth order eigenvalue problem

$$\frac{\mathrm{d}^4}{\mathrm{d}y^4}X(y) = \lambda^4 X(y) \qquad y \in [-1,1] \tag{7}$$

$$X(y=\pm 1) = 0 \qquad \frac{\mathrm{d}}{\mathrm{d}y}X(y=\pm 1) = 0. \tag{8}$$

Two different sets of eigenvalues and the corresponding eigenfunctions are found, respectively odd and even, by numerically solving the following transcendental equations

$$tan(\lambda_n) - tanh(\lambda_n) = 0 \quad (odd\ set) \tag{9}$$

$$tan(\lambda_n) + tanh(\lambda_n) = 0 \quad (even\ set). \tag{10}$$

The corresponding normalized eigenfunctions (figure 1) are

$$X_n = \frac{1}{\sqrt{2}}\left[\frac{sinh(\lambda_n y)}{sinh(\lambda_n)} - \frac{sin(\lambda_n y)}{sin(\lambda_n)}\right] \qquad n = 1,3,5..,N-1 \tag{11}$$

$(odd\ set)$

$$X_n = \frac{1}{\sqrt{2}}\left[\frac{cosh(\lambda_n y)}{cosh(\lambda_n)} - \frac{cos(\lambda_n y)}{cos(\lambda_n)}\right] \qquad n = 2,4,6..,N \tag{12}$$

$(even\ set)$.

Similar functions, in a different domain, have been used in the study of the circular Couette flow between coaxial cylinders [1, appendix V].

Since the imaginary and the real part of the solution $\hat{v}$ usually have opposite parity, independently on the initial condition, both the odd and the even sets are necessary to completely describe the problem and obtain the correct result.
In the following paragraphs a compact notation for the space derivatives is introduced. In order to simplify the reading, the y-derivatives will be indicated with a subscript. The temporal derivatives will be indicated explicitly or with a dot.

The numerical solution to the $\hat{v}$ equation (1) is obtained by applying the variational Galerkin method. Truncating the series (5) at N functions and substituting yields

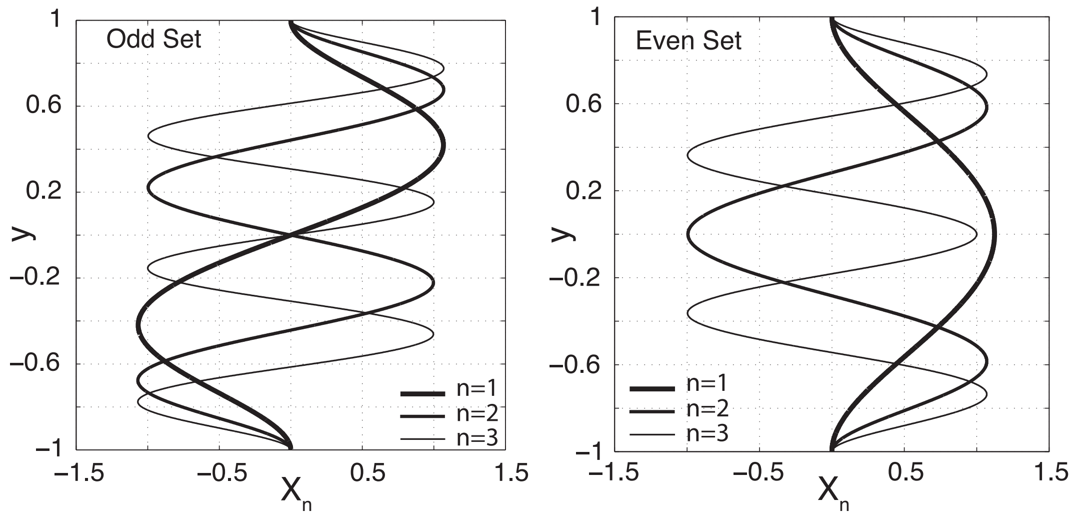


FIG. S 1: The basis eigenfunctions

$$\begin{aligned}\epsilon(y,t;\ \alpha,\ \gamma) = {} & \sum_{n=1}^{N} \frac{\mathrm{d}}{\mathrm{d}t} c_n(t) X_{n_{yy}} - k^2 \sum_{n=1}^{N} \frac{\mathrm{d}}{\mathrm{d}t} c_n(t) X_n \\ & + i\alpha U(y) \sum_{n=1}^{N} c_n(t) X_{n_{yy}} - i\alpha k^2 U(y) \sum_{n=1}^{N} c_n(t) X_n \\ & - i\alpha \frac{\mathrm{d}^2 U(y)}{\mathrm{d}y^2} \sum_{n=1}^{N} c_n(t) X_n - \frac{1}{Re} \sum_{n=1}^{N} c_n(t) X_{n_{yyyy}} \\ & + \frac{2k^2}{Re} \sum_{n=1}^{N} c_n(t) X_{n_{yy}} - \frac{k^4}{Re} \sum_{n=1}^{N} c_n(t) X_n. \end{aligned} \tag{13}$$

The error functional ϵ is minimized when it is orthogonal to the space of the linearly independent trial functions X_n with $n = 1, 2, ..N$. In this context, given two functions $u(y)$ and $v(y)$ with $y \in\ \Omega = [-1,\ 1]$, the following definition of scalar product applies

$$\langle u, v \rangle = \int_{\Omega} u \cdot v \, \mathrm{d}y. \tag{14}$$

With above notation the Galerkin orthogonality condition is expressed as

$$\langle \epsilon, X_m \rangle = 0 \qquad\qquad m = 1, 2, ..., N. \tag{15}$$

The Orr-Sommerfeld PDE is now reduced to a system of N ODEs of the first order, where the time dependent coefficients $c_n(t)$ are the only unknowns.

$$
\begin{aligned}
0 = &\sum_{n=1}^{N} \frac{\mathrm{d}}{\mathrm{d}t} c_n(t)\langle X_{n_{yy}}, X_m\rangle - k^2 \sum_{n=1}^{N} \frac{\mathrm{d}}{\mathrm{d}t} c_n(t)\langle X_n, X_m\rangle \\
&+ i\alpha \sum_{n=1}^{N} c_n(t)\langle U(y)X_{n_{yy}}, X_m\rangle - i\alpha k^2 \sum_{n=1}^{N} c_n(t)\langle U(y)X_n, X_m\rangle \\
&- i\alpha \sum_{n=1}^{N} c_n(t)\langle \frac{\mathrm{d}^2U(y)}{\mathrm{d}y^2} X_n X_m\rangle - \frac{1}{Re} \sum_{n=1}^{N} c_n(t)\langle X_{n_{yyyy}}, X_m\rangle \\
&+ \frac{2k^2}{Re} \sum_{n=1}^{N} c_n(t)\langle X_{n_{yy}}, X_m\rangle - \frac{k^4}{Re} \sum_{n=1}^{N} c_n(t)\langle X_n, X_m\rangle \\
&n,\ m = 1, 2, 3, ..., N. \qquad (16)
\end{aligned}
$$

The scalar products can be evaluated analytically or computed by numerical integration:

$$D_{m,n} = \langle X_n, X_m\rangle = \delta_{m,n} \qquad (17)$$

$$S_{m,n} = \langle X_{n_{yy}}, X_m\rangle = \qquad (18)$$

$$
= \begin{cases} +4\frac{\lambda_n^2\lambda_m^2}{\lambda_n^4-\lambda_m^4}(\lambda_n\mu_n - \lambda_m\mu_m) & \text{if } (n+m) \text{ is even, } n \neq m \\ 0 & \text{if } (n+m) \text{ is odd} \\ -\lambda_n^2\mu_n^2 + \lambda_m\mu_m & \text{if } n = m \end{cases}
$$

$$
\begin{aligned}
F_{m,n} &= \langle X_{n_{yyyy}}, X_m\rangle = \lambda_n^4\delta_{m,n} && (19) \\
U^{(1)}_{m,n} &= \langle U(y)X_{n_{yy}}, X_m\rangle && (20) \\
U^{(2)}_{m,n} &= \langle U(y)X_n, X_m\rangle && (21) \\
U^{(3)}_{m,n} &= \langle \frac{\mathrm{d}^2U(y)}{\mathrm{d}y^2} X_n X_m\rangle, && (22)
\end{aligned}
$$

where

$$\mu_n = \frac{cosh(2\lambda_n) - cos(2\lambda_n)}{sinh(2\lambda_n) - sin(2\lambda_n)} \qquad \lim_{n\to\infty} \mu_n = 1. \qquad (23)$$

It is convenient to express the ODE system (16) in a compact notation: in the following, vectors will be indicated either explicitly using braces or with bold lower case letters; matrices will be indicated with bold capital letters; constants with roman capital letters and physical parameters in italic. The system can be written as

$$\mathbf{H}\,\dot{\mathbf{c}} - \mathbf{G}\,\mathbf{c} = 0, \tag{24}$$

where

$$\mathbf{H} = \mathbf{S} - k^2\,\mathbf{D} \tag{25}$$

$$\begin{aligned}\mathbf{G} &= -i\alpha\,\mathbf{U^{(1)}} + i\alpha k^2\,\mathbf{U^{(2)}} + i\alpha\,\mathbf{U^{(3)}} \\ &\quad + \frac{1}{Re}\,\mathbf{F} - \frac{2k^2}{Re}\,\mathbf{S} + \frac{k^4}{Re}\,\mathbf{D},\end{aligned} \tag{26}$$

where $\mathbf{D} = [D_{m,n}]$ etc., i.e. the element $D_{m,n}$ is placed at the n^{th} column and at the m^{th} row of the matrix. $\mathbf{H}$ is invertible, so denoting $\mathbf{A} = \mathbf{H}^{-1}\,\mathbf{G}$ yields

$$\dot{\mathbf{c}} - \mathbf{A}\,\mathbf{c} = 0. \tag{27}$$

The Orr-Sommerfeld equation is eventually reduced to a system of ODEs. The complex eigenvalues μ_i of $\mathbf{A}$ constitute the spectrum. The general solution to the system (27) in the case of matrix $\mathbf{A}$ having N distinct eigenvalues μ_i [5], reads

$$\mathbf{c}(t) = \mathrm{K}_1\,\mathbf{l}_1 e^{\mu_1 t} + \mathrm{K}_2\,\mathbf{l}_2 e^{\mu_2 t} + \ldots + \mathrm{K_N}\,\mathbf{l}_N e^{\mu_N t} \tag{28}$$

where $\mathbf{l}_i$ are the eigenvectors corresponding to μ_i and $\mathrm{K_i}$ are constants to be determined by imposing the initial condition The coefficients at the initial time, $\mathbf{c_0}$, can be obtained from the inverse transformation (6) since the initial condition $\hat{v}(t=0)$ is known, so finally the solution 28 is get by solving the algebraic system

$$\mathbf{c_0} = \mathrm{K}_1\,\mathbf{l}_1 + \mathrm{K}_2\,\mathbf{l}_2 + \ldots\ldots.. + \mathrm{K_N}\,\mathbf{l}_N \tag{29}$$

$$\{K_i\} = \mathbf{L}^{-1}\,\mathbf{c_0}. \tag{30}$$

1.2 *Solution to the nonhomogeneous $\hat{\omega}_y$ equation*

In order to solve the Squire equation 2, a set of normal functions different from the one adopted for the velocity is needed, since the second order PDE only requires $\hat{\omega}_y$ to vanish at the boundaries, but not its first derivative. A simple choice for the basis functions, here adopted, is the following

$$Y_n = sin(\xi_n y) \qquad n = 1, 3, 5, \ldots N-1 \qquad (odd\ set) \tag{31}$$

$$Y_n = cos(\xi_n y) \qquad n = 2, 4, 6, \ldots N \qquad (even\ set), \tag{32}$$

where

$$\xi_n = \frac{(n+1)\pi}{2} \qquad n = 1, 3, 5, \ldots N-1 \qquad (odd\ set) \tag{33}$$

$$\xi_n = \frac{(n-1)\pi}{2} \qquad n = 2, 4, 6, \ldots N \qquad (even\ set). \tag{34}$$

Also in this case, note that two sets of eigenfunctions are put together to form a unique set, since both are necessary to completely describe the complex-valued normal vorticity. The general solution is then obtained as the sum of a particular solution $\hat{\omega}_{y_p}$ and the solution to the corresponding homogeneous equation $\hat{\omega}_{y_h}$:

$$\hat{\omega}_y(y,t) = \hat{\omega}_{y_h}(y,t) + \hat{\omega}_{y_p}(y,t), \tag{35}$$

$$\hat{\omega}_y(y,t) = \sum_{n=1}^{\infty}(b_{h_n} + b_{p_n})(t) Y_n(y). \tag{36}$$

Applying the Galerkin method (and truncating the expansion to N terms) as done in the previous section for the velocity, yields to the following forced ODE set:

$$\dot{\mathbf{b}} - \underbrace{\Big(-i\alpha\,\mathbf{U}^* + \frac{1}{Re}\mathbf{S}^* - \frac{2k^2}{Re}\mathbf{D}^*\Big)}_{\mathbf{G}^*}\mathbf{b} = \underbrace{-i\gamma\,\mathbf{F}^*}_{\mathbf{B}}\mathbf{c} \tag{37}$$

$$\dot{\mathbf{b}} - \mathbf{G}^*\,\mathbf{b} = \mathbf{B}\,\mathbf{c}, \tag{38}$$

where the eigenvalues of $\mathbf{G}^*$ constitute the spectrum of the Squire equation and

$$D^*_{m,n} = \langle Y_n, Y_m\rangle = \delta_{m,n} \tag{39}$$

$$S^*_{m,n} = \langle Y_{n_{yy}}, Y_m\rangle = -\xi_n^2\delta_{m,n} \tag{40}$$

$$U^*_{m,n} = \langle U(y)Y_n, Y_m\rangle \tag{41}$$

$$F^*_{m,n} = \langle\frac{\mathrm{d}U(y)}{\mathrm{d}y}X_n, Y_m\rangle. \tag{42}$$

For the plane Couette flow, analytical expressions are computed:

$$U^*_{m,n} = \langle U(y)Y_n, Y_m\rangle = \tag{43}$$

$$= \begin{cases} 0 & \text{if } (n+m) \text{ is even, or } n=m \\ \frac{(-1)^{\frac{n+m+1}{2}} 4\xi_n\xi_m}{(\xi_n^2-\xi_m^2)^2} & \text{if } (n+m) \text{ is odd} \end{cases}$$

$$F^*_{m,n} = \langle\frac{\mathrm{d}U(y)}{\mathrm{d}y}X_n, Y_m\rangle = \tag{44}$$

$$= \begin{cases} \frac{2\sqrt{2}\xi_m\xi_n^2(-1)^{\frac{m+1}{2}}}{\xi_m^4-\xi_n^4} & \text{if } n,m \text{ are odd} \\ \frac{2\sqrt{2}\xi_m\xi_n^2(-1)^{\frac{m}{2}}}{\xi_m^4-\xi_n^4} & \text{if } n,m \text{ are even} \end{cases}$$

1.2.1 *Particular solution* $\mathbf{b}_p$

Since the solution (28) in terms of the expansion coefficients $\mathbf{c}(t)$ is a combination of exponentials and represents the forcing term in (38), a particular solution $\mathbf{b}_p$ in the following form is sought:

$$b_{p_n}(t) = \sum_{j=1}^{N} a_{nj} e^{\mu_j t}, \tag{45}$$

where a_{nj} are constants and μ_j are the eigenvalues of $\mathbf{A}$ (see eq. 28), through which the forcing term is expressed. Yields:

$$\hat{\omega}_{y_p}(y,t) = \sum_{n=1}^{N} b_{p_n}(t) Y_n(y). \tag{46}$$

Diagonalizing the system (27), the coefficients of the normal-velocity result

$$\mathbf{c}(t) = \mathbf{L}\,\mathbf{h} = \mathbf{L} \begin{Bmatrix} h_{0_1} e^{\mu_1 t} \\ h_{0_2} e^{\mu_2 t} \\ \vdots \\ h_{0_N} e^{\mu_N t} \end{Bmatrix} \qquad \mathbf{h_0} = \mathbf{L}^{-1}\,\mathbf{c}(t=0) \tag{47}$$

Substituting the particular solution (45) in (38) and leads to

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{Bmatrix} a_{11} e^{\mu_1 t} + \dots a_{1N} e^{\mu_N t} \\ \vdots \\ a_{N1} e^{\mu_1 t} + \dots a_{NN} e^{\mu_N t} \end{Bmatrix} + \mathbf{G}^* \begin{Bmatrix} a_{11} e^{\mu_1 t} + \dots a_{1N} e^{\mu_N t} \\ \vdots \\ a_{N1} e^{\mu_1 t} + \dots a_{NN} e^{\mu_N t} \end{Bmatrix} = \mathbf{B}\,\mathbf{L} \begin{Bmatrix} h_{0_1} e^{\mu_1 t} \\ h_{0_2} e^{\mu_2 t} \\ \vdots \\ h_{0_N} e^{\mu_N t} \end{Bmatrix} \tag{48}$$

It is straightforward to find the unknown constants a_{nj} by comparing terms with the same exponential factor. This is equivalent to solve the following set of N algebraic systems

$$(\mu_j \mathbb{I} - \mathbf{G}^*) \begin{Bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{Nj} \end{Bmatrix} = h_{0j} \begin{Bmatrix} B^*_{1j} \\ B^*_{2j} \\ \vdots \\ B^*_{Nj} \end{Bmatrix} \qquad j = 1, 2, ...N \tag{49}$$

where $\mathbb{I}$ is the identity matrix and B^*_{ij} are the elements of the matrix $\mathbf{B}\,\mathbf{L}$. As usual, the first subscript indicates the row and the second one indicates the column.

Finally we get the matrix of coefficients column by column as:

$$\begin{Bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{Nj} \end{Bmatrix} = (\mu_j \mathbb{I} - \mathbf{G^*})^{-1} h_{0j} \begin{Bmatrix} B^*_{1j} \\ B^*_{2j} \\ \vdots \\ B^*_{Nj} \end{Bmatrix} \qquad j = 1, 2, ...N. \tag{50}$$

1.2.2 *Homogeneous and complete solution* **b**

The homogeneous solution of (38) takes the same form of the of equation (28). Indicating with μ^* and $\mathbf{l}^*$ respectively the eigenvalues and eigenvectors of the matrix $\mathbf{G^*}$, it follows

$$\mathbf{b}_h(t) = \mathrm{C}_1\, \mathbf{l}^*_1 e^{\mu^*_1 t} + \mathrm{C}_2\, \mathbf{l}^*_2 e^{\mu^*_2 t} + \ldots + \mathrm{C}_\mathrm{N}\, \mathbf{l}^*_N e^{\mu^*_N t} \tag{51}$$

Finally the complete solution is

$$\mathbf{b}(t) = \mathrm{C}_1\, \mathbf{l}^*_1 e^{\mu^*_1 t} + \mathrm{C}_2\, \mathbf{l}^*_2 e^{\mu^*_2 t} + \ldots + \mathrm{C}_\mathrm{N}\, \mathbf{l}^*_N e^{\mu^*_N t} + \mathbf{b}_p(t) \tag{52}$$

The unknown constants C_i depends on the initial condition, and can be calculated setting $t = 0$ in the above expression, leading to

$$\mathbf{h}^*_\mathbf{0} = \{\mathrm{C}_i\} = \mathbf{L^*}^{-1}(\mathbf{b_0} - \mathbf{b_0}_p) \tag{53}$$

1.3 Convergence

The Galerkin method was first applied to the Orr-sommerfeld modal equation by [2]. They used normal functions that guarantee a $1/N^4$ convergence ratio. Gallagher [3] used, for the modal problem, the Chandrasekhar-Reid functions and the error decreased as $1/N^5$ with $N \to \infty$ as shown in [4]. The fifth order of accuracy is ensured for the present formulation as well, as shown in figure 2.

The method results to be fast and accurate in time and space. Since the time evolution is analytically represented, the complete wave transient, up to the asymptote, can be simulated without typical drawbacks of time-marching techniques. Arbitrary initial conditions can be specified for bounded flows. The limits of the method are related to the non-normality of the Orr-Sommerfefd and Squire operators. The non-normal effects act on the numerical procedure by worsening the condition number of the eigenvector matrices. Anyway, the sensibility of the spectrum (especially at high Re and k values) is a property of the stability operator and it is thus independent on the numerical scheme.

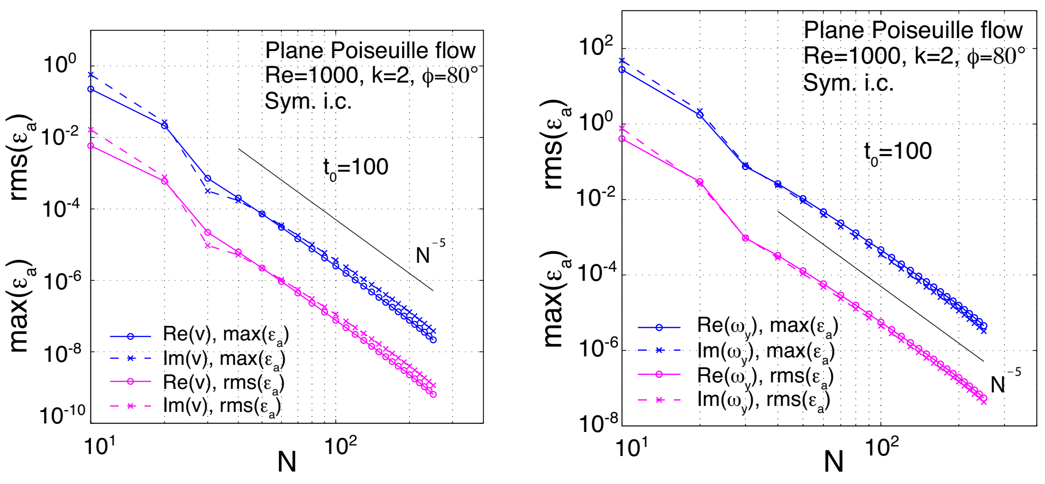


FIG. S 2: Maximum and rms of the absolute error of $\hat{v}$ (left panel) and $\hat{\omega}_y$ (right panel) as a function of the number of modes N for channel flow with $t_0 = 100$, $Re = 1000$, $k = 2$, $\phi = 80°$ and symmetrical initial condition. Continuous line: real part. Dashed line: imaginary part. Magenta line: maximum absolute error. Blue line: rms of the absolute error. Black line: accuracy trend N^{-5} [4]. Since the exact solution is not known, the residuals are defined as the difference between the solution and an accurate solution computed with 350 modes. $\epsilon_a(y,t) = |\hat{v}_N(y,t) - \hat{v}_{N=350}(y,t)|$, $rms(\epsilon_a)(t) = \frac{1}{N_y}\sqrt{\sum_{i=1}^{N_y} \epsilon_a^2(y,t)}$, $max(\epsilon_a)(t) = \max_{y_i}(\epsilon_a(y,t))$

1.3.1 *Derivatives of the solution*

The analytical expressions for the derivatives of the basis functions are known. Thereby, it is easy to investigate the convergence of the series obtained by termwise differentiation. The Chandrasekhar functions series results to be derivable up to the 3^{rd} derivative. The 4^{th} derivative does not converge at the boundaries. About the Fourier series for the vorticity, the series can be derived termwise un to the second derivative.

2 The numerical code (Matlab®)

The method described above is implemented in matlab. Here we report a brief description for each routine

2.1 compute_eigenvalues.m

This routine creates the file *eigenvalues.mat*, containing the solutions (λ_i, stored in the variable `gtot`) to equation 7. This file needs to be computed just once, we already provide a file contaning the first 500 eigenvalues for both the symmetric and antisymmetric sets. This allows to use up to 500 Chandrasekhar modes in the main code, which is usually enough.

2.2 compute_matrices.m

This is a matlab function invoked by the main program. It computes:

- the Chandrasekhar eigenfunctions (11,12) and their derivatives (remind that the series 5 can be derivated termwise 3 times);
- the matrices (17-22) `D`, `S`, `F`,`U1`, `U2`,`U3` . For the Couette flow, the analytical expressions are used. Where these are not available, a numerical integration is implemented using a fine grid (see main program);
- the eigenfunctions 31 for the wall-normal vorticity expansion, and their derivatives (the Fourier series is derivable termwise once);
- the matrices (36-39), named `DD`, `SS`, `UU` and `FORZ` respectively, in the codes;

2.3 main_ivp_galerkin_1.0.m

Main program. Input parameters:

- `k_polar`: polar wavenumber
- `Re`: Reynolds number
- `phi`: waveangle
- `type`: base flow type; in this version it can be either `'Poiseuille'`,`'Couette'`, or `'Wake'`, but the user could specify any parallel base flow
- `ic`: initial condition label, in this version can be either `'sym'`, or `'asym'`
- `N`: number of modes used for the expansion. 200-300 usually works fine. N must be even. Don't exceed the number of eigenvalues saved in the file *eigenvalues.mat* needed by the main code (we provide a file with 500 modes).

- `h`: step size of y-grid . This is the grid where the final solution is computed. The user can specify whatever value for h. Indeed, the accuracy of the solution won't be affected by the numer of points of `y`, since the numerical integrations are computed on a fine grid named `yy`. Notice however that the integral energy, `G` is affected.
- `hh`: step size of fine y-grid (`yy`). This grid is used to compute the initial coefficients, and the matrices needed by the galerkin method. A value of 2e-3 works fine. The accuracy of the solution depends on this grid
- `y0`: y observation point for the frequency temporal evolution
- `tmax`: maximum (nondimensional) time simulated
- `nt`: number of points of the coarse time-grid (see the flow-chart for further detail on the temporal grids)
- `dtf`: spacing of fine time grid. See scheme in the flow-chart. The user can choose if to use a regular time grid (NB: the semi-analytical method allows to get the solution even at a single instant, nt can be an arbitrary number ≥ 2), or a different grid built to allow an accurate computation of time derivatives (4th order). We use this grid when we compute the wave frequency transients, indeed the wave frequency in obtained as the time derivative of the unwrapped phase. In this way, the user can compute the correct frequency at the time instants he wants.
- `x`: stramwise position for the wake base flow
- `realbound`: the real limits of the domain. This is needed for the wake flow, for which a wider domain (in terms of nondimensional units) is required. For the bounded flows realbound is set equal to 1 automatically. NB: some scalings are required in the procedure to take account of this domain stretching, since the eigenfunctions (both Chandrasekhar and Fourier) are defined in [-1,1] - see for example line 198 of the main code and line 7 of *solve_squire.m*. For the wake flow, we typically set realbound in [60,100] to get a good approximation of the unbounded domain.

Please refer to the flow-chart and to the comments in the scripts for further details. The main code calls the function *solve_squire.m* for 3D waves. Find below a table with the references between the names of variables in the code and their names in the first section of this pdf file. Just a few general comments:

1. The code allows to compute the solution $(\hat{v}, \hat{u}, \hat{w}, \hat{\omega}_x, \hat{\omega}_y, \hat{\omega}_z)$ at arbitrary points (y,t). The accuracy of the solution is determined by the fine y-grid,

`yy`. For instance:
`t=linspace(0,tmax,nt);`$\rightarrow$ solution from $t = 0$ to $t =$ `tmax`
The user may want to observe only a limited range of time, for the same simulation: `t=linspace(20,30,nt);`$\rightarrow$ the solution (with the same initial condition as the above case) is computed only from $t = 20$ to $t = 30$

2. The bounudary conditions for the perturbations are only no-slip. For unbounded flows, as the wake flow, other boundary conditions are theoretically possible (for example the perturbations may be just bounded as $y \rightarrow \infty$). These boundary conditions are not implemented in this code.

3. The code is versatile, as the user could easily specify any other expression of base flow, and initial conditions (satisfying the no-slip b.c.).

4. The initial y-vorticity is set to zero. This is done at lines 50-51 of the function *solve_squire.m*

5. ...

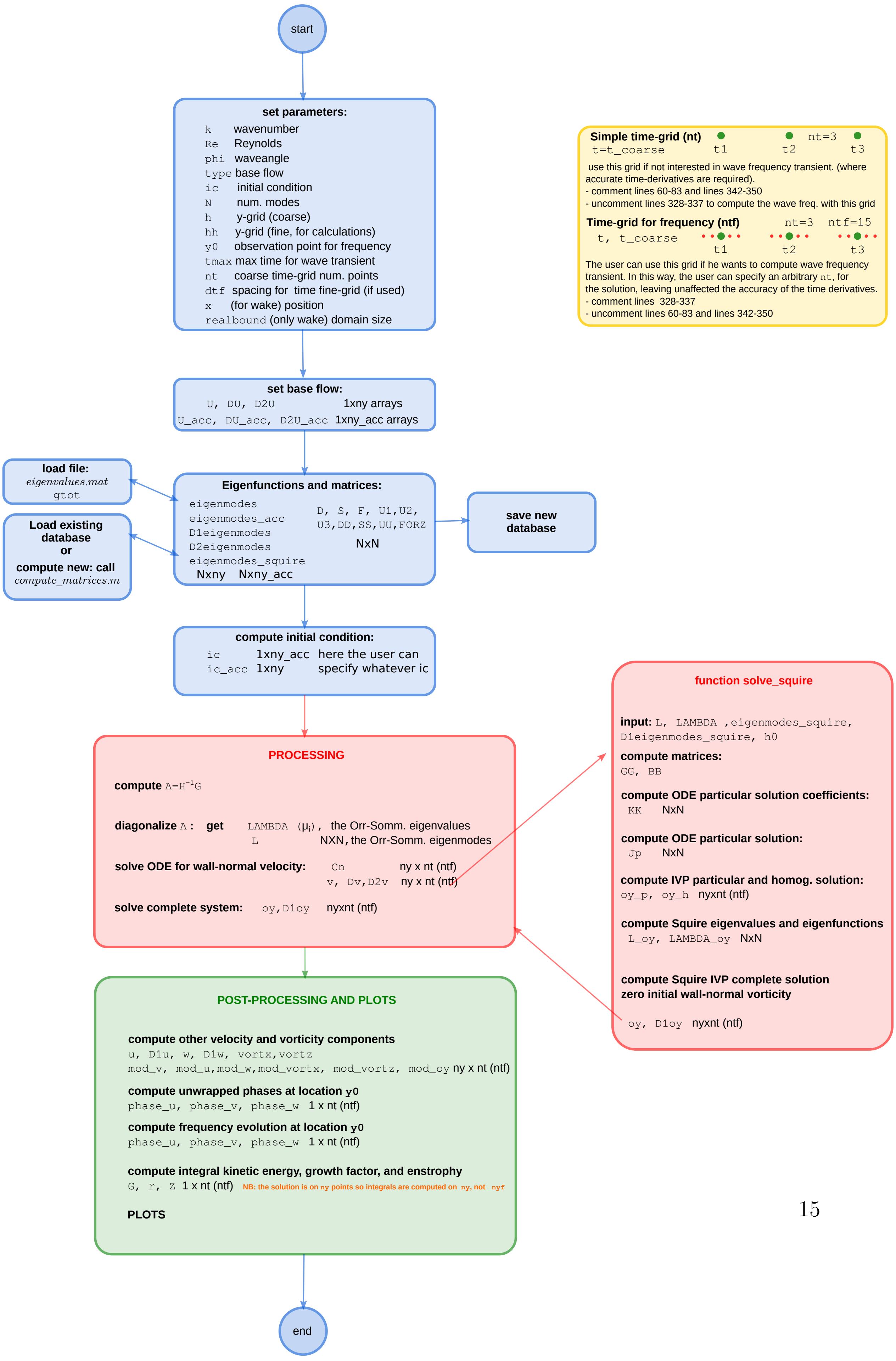
2.4 solve_squire.m

This function is called my the main program after the solution of the Orr-Sommerfeld ivp, at line 282. It is devoted to the solution of the forced Squire equation.

2.5 Variables and flowchart

For the user convenience, we report in the following table the main variables in the code, their size, and their corresponding variables as introduced in the first section of this file.

code variable	variable	size
`k_polar`	k	`1`
`Re`	Re	`1`
`phi`	ϕ	`1`
`N`	N	`1`
`y,y2`	y	`1 x ny`
`y_acc,y2_acc`	y	`1 x ny_acc`
`t`	t	`1 x nt (ntf)`
`U,DU,D2U`	$U, \frac{dU}{dy}, \frac{d^2U}{dy^2}$	`1 x ny`
`U_acc,DU_acc,D2U_acc`	$U, \frac{dU}{dy}, \frac{d^2U}{dy^2}$	`1 x ny_acc`
`g`	λ_n (eq. 7)	`1 x N`
`eigenmodes`	$X_n(y)$ (eq. 11-12)	`ny x N`
`D1eigenmodes`	$X_{n_y}(y)$	`ny x N`
`D2eigenmodes`	$X_{n_{yy}}(y)$	`ny x N`
`eigenmodes_acc`	$X_n(y)$ (eq. 11-12)	`ny_acc x N`
`D1eigenmodes_acc`	$X_{n_y}(y)$	`ny_acc x N`
`D2eigenmodes_acc`	$X_{n_{yy}}(y)$	`ny_acc x N`
`eigenmodes_squire`	$Y_n(y)$ (eq. 31-32)	`ny x N`
`D1eigenmodes_squire`	$Y_{n_y}(y)$	`ny x N`
`eigenmodes_squire_acc`	$Y_n(y)$ (eq. 31-32)	`ny_acc x N`
`D1eigenmodes_squire_acc`	$Y_{n_y}(y)$	`ny_acc x N`
`D`	$\mathbf{D}$ (eq. 17)	`N x N`
`S`	$\mathbf{S}$ (eq. 18)	`N x N`
`F`	$\mathbf{F}$ (eq. 19)	`N x N`
`U1`	$\mathbf{U}^{(1)}$(eq. 20)	`N x N`
`U2`	$\mathbf{U}^{(2)}$(eq. 21)	`N x N`
`U3`	$\mathbf{U}^{(3)}$(eq. 22)	`N x N`
`DD`	$\mathbf{D}^{*}$(eq. 39)	`N x N`
`SS`	$\mathbf{S}^{*}$(eq. 40)	`N x N`
`UU`	$\mathbf{U}^{*}$(eq. 41)	`N x N`
`FORZ`	$\mathbf{F}^{*}$(eq. 42)	`N x N`
`G`	$\mathbf{G}$(eq. 26)	`N x N`
`H`	$\mathbf{H}$(eq. 25)	`N x N`
`A`	$\mathbf{A}$(eq. 27)	`N x N`
`L`	$\mathbf{L}$(eq. 30)	`N x N`
`GG`	$\mathbf{G}^{*}$(eq. 38)	`N x N`
`B`	$\mathbf{B}$(eq. 38)	`N x N`
`KK`	a_{nj}(eq. 45)	`N x N`
`eigenvalues_v`	μ_i(eq. 28)	`1 x N`
`Cn`	$\mathbf{c}(t)$(eq. 28)	`N x nt(ntf)`
`h`	$\mathbf{L^{-1}c}$(eq. 28)	`N x nt(ntf)`
`Jp`	$\mathbf{b}_p(t)$(eq. 36)	`N x nt(ntf)`
`Jh`	$\mathbf{b}_h(t)$(eq. 36)	`N x nt(ntf)`
`oy_p`	$\hat{\omega}_{y_p}(y,t)$(eq. 35)	`ny x nt(ntf)`
`oy_h`	$\hat{\omega}_{y_h}(y,t)$(eq. 35)	`ny x nt(ntf)`
`oy=oy_h+oy_p`	$\hat{\omega}_y(y,t)$(eq. 35)	`ny x nt(ntf)`
`L_oy`	$\mathbf{L}^{*}$(eq. 53)	`N x N`
`eigenvalues_oy`	$\mu_{\mathbf{i}}^{*}$(eq. 51)	`1 x N`
`v`	$\hat{v}(y,t)$(eq. 5)	`ny x nt(ntf)`
`u`	$\hat{u}(y,t)$	`ny x nt(ntf)`
`w`	$\hat{w}(y,t)$	`ny x nt(ntf)`
`vortx`	$\hat{\omega}_x(y,t)$	`ny x nt(ntf)`
`vortz`	$\hat{\omega}_z(y,t)$	`ny x nt(ntf)`


start
set parameters:
k wavenumber
Re Reynolds
phi waveangle
type base flow
ic initial condition
N num. modes
h y-grid (coarse)
hh y-grid (fine, for calculations)
y0 observation point for frequency
tmax max time for wave transient
nt coarse time-grid num. points
dtf spacing for time fine-grid (if used)
x (for wake) position
realbound (only wake) domain size
set base flow:
U, DU, D2U 1xny arrays
U_acc, DU_acc, D2U_acc 1xny_acc arrays
load file:
eigenvalues.mat
gtot
Load existing database
or
compute new: call
compute_matrices.m
Eigenfunctions and matrices:
eigenmodes
eigenmodes_acc
D1eigenmodes
D2eigenmodes
eigenmodes_squire
Nxny Nxny_acc
D, S, F, U1,U2,
U3,DD,SS,UU,FORZ
NxN
save new database
compute initial condition:
ic 1xny_acc here the user can
ic_acc 1xny specify whatever ic
PROCESSING
compute $A=H^{-1}G$
diagonalize A : get LAMBDA (μ_i), the Orr-Somm. eigenvalues
L NXN, the Orr-Somm. eigenmodes
solve ODE for wall-normal velocity: Cn ny x nt (ntf)
v, Dv,D2v ny x nt (ntf)
solve complete system: oy,D1oy nyxnt (ntf)
function solve_squire
input: L, LAMBDA ,eigenmodes_squire, D1eigenmodes_squire, h0
compute matrices:
GG, BB
compute ODE particular solution coefficients:
KK NxN
compute ODE particular solution:
Jp NxN
compute IVP particular and homog. solution:
oy_p, oy_h nyxnt (ntf)
compute Squire eigenvalues and eigenfunctions
L_oy, LAMBDA_oy NxN
compute Squire IVP complete solution
zero initial wall-normal vorticity
oy, D1oy nyxnt (ntf)
POST-PROCESSING AND PLOTS
compute other velocity and vorticity components
u, D1u, w, D1w, vortx,vortz
mod_v, mod_u,mod_w,mod_vortx, mod_vortz, mod_oy ny x nt (ntf)
compute unwrapped phases at location y0
phase_u, phase_v, phase_w 1 x nt (ntf)
compute frequency evolution at location y0
phase_u, phase_v, phase_w 1 x nt (ntf)
compute integral kinetic energy, growth factor, and enstrophy
G, r, Z 1 x nt (ntf)
NB: the solution is on ny points so integrals are computed on ny, not nyf
PLOTS
end
Simple time-grid (nt)
nt=3
t=t_coarse
t1
t2
t3
Time-grid for frequency (ntf)
nt=3 ntf=15
t, t_coarse
t1
t2
t3

Simple time-grid (nt)

use this grid if not interested in wave frequency transient. (where accurate time-derivatives are required).
- comment lines 60-83 and lines 342-350
- uncomment lines 328-337 to compute the wave freq. with this grid

Time-grid for frequency (ntf)

The user can use this grid if he wants to compute wave frequency transient. In this way, the user can specify an arbitrary `nt`, for the solution, leaving unaffected the accuracy of the time derivatives.
- comment lines 328-337
- uncomment lines 60-83 and lines 342-350

References

[1] S. Chandrasekhar. *Hydrodynamic and Hydromagnetic Stability.* Oxford University Press, 1961.

[2] C. L. Dolph and D. C. Lewis. On the application of infinite systems of ordinary differential equations to perturbations of plane poiseuille flow. *Quart. Appl. Math*, 16:97–110, 1958.

[3] A. Gallagher and McD. Mercer. On the behaviour of small disturbances in plane couette flow. *J. Fluid Mech.*, 13:91–100, 1962.

[4] S. A. Orszag. Accurate solution of the orr-sommerfeld stability equation. *J. Fluid Mech.*, 50:689–703, 1971.

[5] D. G. Zill and M. R. Cullen. *Differential Equations with Boundary Value Problems.* Brooks/Cole, 2005.