Reused LU factorization as a preconditioner for efficient solution of the Parabolized Stability Equations

András Szabó (aszabo@hds.bme.hu)
Budapest University of Technology and Economics

György Paál
Budapest University of Technology and Economics

Research Article

Keywords: Boundary layer stability, Parabolized stability equations, Preconditioner, Iterative solution

Posted Date: February 6th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2538432/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.
Reused LU factorization as a preconditioner for efficient solution of the Parabolized Stability Equations

András Szabó* and György Paál


*Corresponding author(s). E-mail(s): aszabo@hds.bme.hu; paal@hds.bme.hu;

Contributing authors: paal@hds.bme.hu;

Abstract

The parabolized stability equation (PSE) is a widely used, efficient method to calculate the evolution of streamwise traveling instabilities in spatially developing, weakly nonparallel flows. Although the PSE method is very economical, computational time can be significant due to the repeated solution of linear systems of equations in each marching step. The linear systems of equations are typically solved by LU factorization due to the moderate size and the sparsity of the coefficient matrices. In this paper, instead of repeated calculation of the LU factorization, a single LU factorization is calculated in each marching step, and used as a preconditioner for an iterative solver. Numerical experiments are conducted on the incompressible PSE, nonlinear PSE (NLPSE) with explicit discretization of the nonlinear terms, and NLPSE with implicit treatment of the nonlinearities. It is shown that the time spent on the solution of the linear systems of equations was reduced by a factor of 2.5, 4.5-7.5, and 20-60 for the three cases, respectively. The methodology can be generalized to the compressible PSE equations, and extended to similar boundary layer stability problems, or slowly varying parabolic partial differential equations.

Keywords: Boundary layer stability, Parabolized stability equations, Preconditioner, Iterative solution
1 Introduction

Stability analysis of spatially developing flows is an active research topic that is directly related to many engineering applications. Within the framework of linear stability theory, parabolized stability equations (PSE) [1–3] is a widely used, effective, and accurate tool to evaluate the stability characteristics of the flow. The PSE method relies on splitting the disturbances into a shape and a wave function, in which the wave function carries most of the disturbance variation in the evolution/streamwise\(^1\)/slowly varying direction, and the shape function captures the disturbance variation in direction(s) perpendicular to the direction of evolution. The disturbance equations are parabolized by neglecting second derivatives in the marching direction in the shape function, but the second derivatives are retained in the wave function part. This allows the calculation of the disturbances by marching the PSE equations downstream from a suitable initial condition. This formalism significantly reduces the required number of steps in the marching direction, which is the very reason for the efficiency of the method. Furthermore, PSE formalism can naturally incorporate the nonlinear interaction of higher harmonics through a harmonic-balance formalism (nonlinear PSE - NLPSE) [5], which allows a very economical nonlinear stability calculation.

Although the PSE method is very attractive, it has multiple shortcomings. The parabolization of the equations is ad-hoc, i.e., it does not rely on a rigorous multiple-scale expansion as boundary layer theory [6, 7] or triple-deck theory [8, 9] does. Therefore, there is some ellipticity remaining in the PSE equations, which makes the formalism ill-posed [5, 10–12]. The ill-posedness of the PSE sets a lower limit on the marching step size, limiting the accuracy of the method. The stability of the marching procedure can be enhanced by partially, or completely [13, 14] neglecting the streamwise derivative of the pressure shape function. Unfortunately, this introduces an additional cumulative error in the equations [15]. Alternatively, the PSE can be stabilized by adding a damping term [16]. However, the above remedies do not render the PSE method well-posed. The other drawback of the PSE formalism is that it can only capture the evolution of individual modes and fails to correctly capture the interaction of multiple instability modes correctly [15]. The above shortcoming led to the development of more rigorous parabolization techniques that resolve both issues but are computationally more cumbersome [17–19]. Despite these drawbacks, the PSE method has been extended to compressible and hypersonic flows [20–22], curvilinear coordinates [23] and three-dimensional boundary layers [4, 24], nonequilibrium thermo-chemical interactions and reacting flows [25–28], and surface movement [27]. Moreover, several improvements were proposed to improve the accuracy and robustness of the PSE method [4, 29–34], and it is frequently applied in boundary layer research (recently with more emphasis on hypersonic flows [35]) along with considerable success in jet noise modeling [36].

\(^1\)Note that in practice, the disturbance evolution direction is not necessarily aligned with the free-stream flow [4].
The extension of the PSE formalism to fully three-dimensional flows is especially motivated by an exceptionally large number of engineering applications in boundary layer flows. In low-speed boundary layers, the growth of Tollmien-Schlichting (TS) waves can be dampened by streamwise streaks [37], which can be introduced by Miniature Vortex Generators [38–40] or by free-stream vortices [41–43]. In the case of swept wings, stationary crossflow vortices can be attenuated with discrete roughness elements [44, 45]. In hypersonic flow, appropriately tuned streamwise streaks can stabilize Mack-modes (Mode-2 instabilities) both on flat plates [46] and cones [47]; a control technique that can be achieved by suitable vortex generators [48, 49]. The PSE method could be used to improve the Technology Readiness Level of any of the above methods, with sufficient improvements in the computational cost.

Although the PSE method is quite efficient, the computational time can still be substantial, especially in three-dimensional flows, as also noted by Moyes and Reed [50]. The extension of the PSE formalism to 3D boundary layers is possible in multiple ways [4, 33]. In the plane marching PSE, a fully three-dimensional boundary layer is considered, and 2D disturbance shape functions are calculated at each marching step. This approach was implemented by several different authors: Broadhurst and Sherwin used spectral-element discretization [51]; Paredes et al. combined high order finite difference in the wall-normal direction with Fourier-collocation in the spanwise direction [52]; Theiss et al. discretized the equations with high order finite different in both directions normal to the marching direction [53]; finite difference method was also used by Liu et al. [54], and the same approach was chosen by Rodríguez and colleagues [55, 56], and there also exists the commercial code STABL3D [57, 58]. Alternatively, a locally 2D base flow can be considered, and the three-dimensionality of the disturbances is accounted for by a spanwise wavenumber $\beta$ in the disturbance wave function, which can either be constant (line marching PSE) or vary in the marching direction (surface marching PSE). These approaches are considered in the PSE codes NOLOT [23], LASTRAC3D [4], JoKHeR [59], and EPIC [60]. In both formalisms, the computational time is considerably since the spanwise direction needs to be resolved. Extending the 3D PSE equations into the nonlinear regime is especially challenging for two reasons. First, the system size increases with each additional mode. Second, the nonlinear terms can slow down the convergence speed of the iterative procedure that splits the disturbance variation into shape and wave functions. At a high degree of nonlinearity, an NLPSE calculation can easily diverge. Although explicit treatment of the nonlinear terms is used in almost all studies (since this way, the equation of each mode can be solved separately), implicit discretization of the nonlinear terms may improve the convergence of NLPSE [5]. However, this is a computationally demanding approach and is seldom used except for a few studies study of streak evolution in hypersonic boundary layers [46, 47, 61, 62]. Alternatively, implicit discretization of the mean flow distortion (MFD) in the explicit formulation may also improve convergence [4, 30]. In the case of high mode amplitudes, relaxation of the nonlinear terms
can improve the robustness of the method, but it increases the computational time. Although the computational time is generally not reported in studies, we note that Moyes et al. [32] reported several hundreds of iterations when they considered the surface-marching NLPSE with the explicit discretization of the nonlinear terms. This example highlights the challenges that may arise in the NLPSE method.

The above examples display the need for efficient techniques to solve the PSE problem. The repeated solution of the linear system of equations (that arises from an implicit backward Euler scheme in the marching direction, which is necessary due to the ill-posedness of the PSE formalism) is a computationally demanding task. It takes most of the computational time in the incompressible PSE method and also a significant time in the compressible formalism (in which the evaluation of the nonlinearities is also cumbersome as they consist of several terms). The discretized equations are solved with LU factorization in all the implementations in the literature, in most cases combined with relatively low order discretization methods. This is because sparse LU solvers scale very well to the system size $O(10^3) - O(10^5)$, much better than dense direct solvers [63–65]. For systems this size, sparse LU solvers outperform iterative methods, as iterative methods require efficient preconditioning to be effective. Although the general consensus in the scientific community is that the efficient numerical techniques for the solution of the PSE equations are well understood [32], the following considerations suggest that additional improvements can be made:

- An iterative method, to be efficient, requires a good preconditioner. A preconditioner is an approximation of the inverse of the matrix. An ideal preconditioner is easy to calculate and provides a good approximation of the inverse and it is also computationally cheap to apply. Recently, it was noted in the boundary layer research community by [18, 66] that without a good preconditioner, general iterative methods indeed perform very poorly. In the case of the PSE, within each marching step, the system matrix is changed only slightly as the equation system is solved over and over due to the shape function-wave function splitting. Therefore, an LU factorization from a previous iteration does not solve the equation system, but it almost solves it. This meaning that a previous LU factorization could serve as a preconditioner for an iterative method. This procedure provides a very good approximation of the inverse if the equation changes only slightly, and it is easy to evaluate since only the forward and backward substitutions are required; these are the two exact requirements that make a good preconditioner. If this preconditioner is very efficient, the iterative method may converge in a few steps, meaning that the methodology might outperform LU factorization. This could be especially effective in the NLPSE formalism, as nonlinear interaction may significantly increase the number of iterations.
- In the case of the NLPSE formalism, explicit treatment of the nonlinear terms is much more common than an implicit one, although the implicit formulation offers faster convergence and enhanced robustness. This is mainly because the explicit formalism allows the systems to remain decoupled -
each mode could be solved one-by-one. Conversely, in implicit formalism, a single but much larger matrix has to be inverted, which is computationally more demanding. We note, however, that the hierarchical harmonic-balance construction of the NLPSE formalism yields a block-structured matrix. This block structure allows the construction of simple preconditioners that may be very efficient: block Jacobi (bJ) and block Gauss-Seidel (bGS), which are just the block-wise inverse of the diagonal blocks or the lower (or upper) triangular part of the matrix \([67]\). The efficiency of these preconditioners stems from the fact that for their calculation, only the diagonal blocks need to be inverted, just like in the explicit formalism. These ideas are very common: they were used recently by Moulin et al. \([68]\) for global three-dimensional stability calculation; by Sierra et al. \([69]\) for sensitivity calculation of periodic orbits in fluid flows, and also by \([70]\) for the calculation of nonlinear optimal disturbances in a flat plate boundary layer. These techniques may provide very effective preconditioning to build an efficient iterative procedure that renders implicit NLPSE feasible and favorable with respect to the explicit discretization of the nonlinear terms. This is especially promising, if we consider that the inverses of the diagonal blocks may be reused as outlined above.

In this paper, we set out to conduct well-documented numerical experiments to test the effectiveness of the above-outlined ideas. The plane-marching PSE formalism is applied to an incompressible flat-plate boundary layer perturbed by counter-rotating streamwise vortices. This is a prototypical configuration of streamwise streak generation to suppress the growth of TS waves and delay the laminar-turbulent transition, which was examined in many studies \([41–43]\). Although the examined example has great engineering importance, we perform the calculation to set up cases that demonstrate the challenging aspects of the NLPSE formalism. Therefore, our focus is not on modeling the physical phenomena accurately to learn more about transition delay. The incompressible framework simplifies the formalism, but as the key underlying assumptions hold for high-speed flow, the present methodology is expected to generalize for compressible boundary layers, which have been the focus of recent studies. Furthermore, the generalization of the preconditioning method to the BRE is also discussed. The equations are discretized using the finite element library FreeFem \([71]\), and the linear algebraic operations are performed using the high-performance libraries PETSc \([72, 73]\) and SLEPc \([74]\). FreeFem has been extensively used in global stability calculations and offers a convenient, versatile and flexible framework to solve partial differential equations. PETSc and SLEPc are actively maintained, sparse linear algebra libraries that offer various efficient numerical methods, and the high level of parallelization enables the user to use HPC resources. Therefore, the implementation provides a modern and powerful framework for stability calculations.

This paper is organized as follows. In Section 2, the governing equations, and the preconditioning technique is outlined. In Section 3, the details of the numerical methods are discussed. In Section 4, the problem setup, and the
results of the numerical experiments are discussed. Concluding remarks are made in Section 5.

2 Theory

This section covers the theoretical and modelling aspects of the paper. After a short discussion of the flow configuration, in Section 2.1, the calculation of the base flow - the boundary layer modified by counter-rotating vortices - is covered. Section 2.2 discusses the local stability equations that are used to specify the initial condition of PSE in this study, the BiGlobal stability equations. Section 2.3 is dedicated to the PSE formalism is introduced. Finally, Section 2.4 discusses the preconditioner method.

An idealized, zero angle of incidence, zero pressure gradient incompressible flat-plate boundary layer is considered. At some distance from the leading edge, counter-rotating streamwise vortices are introduced, which generate streamwise streaks in the boundary layer through the lift-up mechanism [75]. The nondimensional form of the governing equations is used. $x$, $y$, and $z$ denote the streamwise, wall-normal and spanwise directions, respectively. As customary in stability analysis, the flow quantities are separated into a stationary base flow and small amplitude perturbations. Let $\mathbf{U} = [U, V, W]^T$ and $P$ denote the nondimensional base flow velocity and pressure, and $\mathbf{u'} = [u', v', w']^T$ and $p'$ the velocity and pressure of the small-amplitude perturbations. The flow quantities and governing equations are nondimensionalised with the free-stream velocity $\hat{U}_0$ and boundary layer length scale $\hat{\delta}_0 = \sqrt{\hat{\mathcal{L}}_0 \hat{\nu} / \hat{U}_0}$, where $\hat{\nu}$ is the kinematic viscosity, $\hat{\mathcal{L}}_0$ is the streamwise location where the perturbing vortices are introduced, measured from the leading edge; dimensional variables are denoted with $\hat{\square}$. $\text{Re} = \hat{\delta}_0 \hat{U}_0 / \hat{\nu}$ is the Reynolds number. The nondimensional pressure is $\hat{P} = \hat{P} / (\hat{\rho} \hat{U}_0^2)$.

2.1 Base flow calculation

Regarding the base flow, it can be assumed that the flow varies slowly in the streamwise direction, while it varies more rapidly in the wall-normal and spanwise directions. The spanwise scaling is a consequence of the vortices that perturb the boundary layer, while the wall normal scaling also holds for an unperturbed boundary layer. Furthermore, the spanwise and wall-normal velocities can be assumed to be smaller than the streamwise velocity. Therefore, the stationary Navier-Stokes equations that govern the evolution of the base flow, can be simplified by using an alternative nondimensionalization for the different variables. Introducing $\varepsilon = 1/\sqrt{\hat{\mathcal{L}}_0 \hat{\nu} / \hat{U}_0}$, $[\tilde{X}, \tilde{Y}, \tilde{Z}]^T = [\hat{x}/\hat{\mathcal{L}}_0, \hat{y}/(\varepsilon \hat{\mathcal{L}}_0), \hat{z}/(\varepsilon \hat{\mathcal{L}}_0)]^T$, $[\tilde{U}, \tilde{V}, \tilde{W}]^T = [\hat{U}/\hat{U}_0, \hat{V}/(\varepsilon \hat{U}_0), \hat{W}/(\varepsilon \hat{U}_0)]^T$ and $\tilde{\mathcal{P}} = \hat{P} / (\varepsilon^2 \hat{\rho} \hat{U}_0^2)$, we get

$$
\tilde{U} \frac{\partial \tilde{U}}{\partial \tilde{X}} + \tilde{V} \frac{\partial \tilde{U}}{\partial \tilde{Y}} + \tilde{W} \frac{\partial \tilde{U}}{\partial \tilde{Z}} = \frac{\partial^2 \tilde{U}}{\partial \tilde{Y}^2} + \frac{\partial^2 \tilde{U}}{\partial \tilde{Z}^2},
$$

(1)
\[ 
\dot{U} \frac{\partial \tilde{V}}{\partial X} + \dot{V} \frac{\partial \tilde{V}}{\partial Y} + \dot{W} \frac{\partial \tilde{V}}{\partial Z} = -\frac{\partial \tilde{P}}{\partial Y} + \frac{\partial^2 \tilde{V}}{\partial Y^2} + \frac{\partial^2 \tilde{V}}{\partial Z^2}, \]  
\[ (2) \]
\[ 
\dot{U} \frac{\partial \tilde{W}}{\partial X} + \dot{V} \frac{\partial \tilde{W}}{\partial Y} + \dot{W} \frac{\partial \tilde{W}}{\partial Z} = -\frac{\partial \tilde{P}}{\partial Z} + \frac{\partial^2 \tilde{W}}{\partial Y^2} + \frac{\partial^2 \tilde{W}}{\partial Z^2}, \]  
\[ (3) \]
\[ \frac{\partial \tilde{U}}{\partial X} + \frac{\partial \tilde{V}}{\partial Y} + \frac{\partial \tilde{W}}{\partial Z} = 0. \]  
\[ (4) \]

Eqs. (1-4) are called the stationary boundary region equations (BRE). These are essentially the extension of the boundary layer equations and are well known and were previously used for several different flow problems as discussed in multiple studies \cite{42, 43, 76}. These equations are parabolic, and with suitable initial- and boundary conditions, well-posed. Thus, they can be marched downstream without any step size restriction. Note that the wall-normal and spanwise coordinates are nondimensionalized with \( \hat{\delta}_0 \), but the streamwise coordinate is nondimensionalized with \( \hat{L}_0 > \hat{\delta}_0 \); furthermore, the wall-normal and spanwise velocities are magnified. Essentially, the appropriate stretching and scaling of the independent and dependent variables make the governing equations Reynolds number independent. Note that the flow quantities calculated with the solution of BRE were rescaled to the nondimensionalization with \( \hat{\delta}_0 \) for the stability calculations.

The initial condition is the Blasius boundary layer with counter rotating Rankine vortices, which is specified in Section 4.1. The boundary conditions are zero velocity at the plate, periodicity in the spanwise direction, while in the far-field, the velocity components are specified as

\[ [\tilde{U}, \tilde{dV}/\tilde{Y}, \tilde{W}] \rightarrow [1, 0, 0], \text{ as } \tilde{Y} \rightarrow \infty. \]  
\[ (5) \]

### 2.2 BiGlobal stability equations

In this section, the stability equations that are used to calculate the initial condition for the PSE are discussed. The general, the three-dimensional linearized Navier-Stokes equations can be simplified in several ways. In the case of a spatially developing flow that slowly evolves in the streamwise direction, neglecting the streamwise variation of the base flow allows a significant simplification. Therefore, if \( U = U(y,z) \), the disturbances can be sought in the form

\[ q'(x,y,z,t) = q(y,z)e^{i(\alpha x - \omega t)} + \text{c.c.,} \]  
\[ (6) \]
where \( i \) is the imaginary unit, \( \omega \) is the angular frequency, \( \alpha \) is the wavenumber, \( q = [u, v, w, p]^T \), and c.c. means complex conjugate. In this formalism, the total instability \( \tilde{q}' \) is decomposed into a shape function \( \tilde{q} \) and a wave function; the streamwise variation of the instability wave is captured in the latter. In general, both \( \alpha \) and \( \omega \) can be complex quantities. In the case of a spatially developing flow such as a boundary layer, spatial stability analysis is appropriate, which
Reusing LU factorization for efficient solution of the PSE

means that \( \omega \) is real, while \( \alpha \) is complex. Therefore, the disturbance wave is assumed to be time-periodic which is exponentially growing or decaying in space. Substituting the above ansatz to the linearized Navier-Stokes equations, we get

\[
(\alpha^2 A_2 + \alpha A_1 + A_0)q = 0,
\]

where the coefficient matrices can be found in [39]. Eq. (7) is a polynomial eigenvalue problem, which can be transformed to a generalized eigenvalue problem using the companion matrix method [77]. This is essentially the same as introducing additional variables the are the state vector \( q \) multiplied by powers of \( \alpha \). The resulting generalized eigenvalue problem is straightforward to solve with standard numerical techniques.

2.3 Parabolized stability equations

In the PSE approximation, we assume that the base flow is not strictly parallel but varies slowly in the streamwise direction. This way, the derived stability equations are still reasonably simple but they preserve more accuracy than Eq. (6). Furthermore, in the NLPSE framework, the total disturbance may consists of multiple frequencies. This allows us to seek the disturbances in the form

\[
q'(x, y, z, t) = \sum_{n=0}^{N} A_n q_n(x, y, z) e^{i(\int_{x_0}^{x} \alpha_n(\xi) d\xi - \omega_n t)} + \text{c.c.}
\]

(8)

In the above equations, \( n \) is the frequency index, and \( x_0 \) is the start of the computational domain, \( \xi \) is a dummy variable for integration, and \( A_n \) is the initial amplitude of the instability mode. This form decomposes the perturbation into a slowly varying shape function and an exponential variation; furthermore, the streamwise variation of the streamwise wavenumber is also allowed. Regarding the frequencies, either a base mode and its many harmonics [5], or narrow- or broadband disturbances may be considered [78]. Substituting this into the perturbation form of the Navier-Stokes equations (where the flow quantities are split into a stationary base flow and disturbances), and neglecting terms that are assumed to be small due to the slow streamwise variation of the flow yields

\[
A_n q_n + B_n \frac{\partial q_n}{\partial x} = F_n.
\]

(9)

where the formula of the coefficient matrices \( A_n, B_n \), and the nonlinear forcing term \( F_n \) can be found in [79]. By specifying a single frequency and neglecting the nonlinear interaction, the linear form of the PSE equations is recovered.

As mentioned in the Introduction, the PSE is ill-posed due to the residual ellipticity. This requires implicit (or semi-implicit [51]) discretization in the streamwise direction, therefore a linear system of equations has to be solved to advance the solution. Furthermore, the ambiguity which arises from the splitting the total disturbance into a shape and wave function part is resolved by introducing a normalization condition. Here, the following normalization
Reusing LU factorization for efficient solution of the PSE

condition is used:
\[
\int \int q_n^* \frac{\partial q_n}{\partial x} dydz = 0,
\]
which ensures that all the exponential variation is absorbed in the wave function [5]. In an iterative procedure, the wavenumber \( \alpha \) is updated in each marching step until the normalization condition is satisfied. In the above equation, \(*\) denotes complex conjugation.

The boundary conditions of the PSE are that the disturbance velocities vanish at the wall and in the far field, and periodicity in the spanwise direction. However, the mean flow distortion (MFD), the zero frequency term that arises from the modal interactions, requires a special treatment. First, in the case of this mode, the wall-normal derivative of the wall-normal velocity approaches zero, to be consistent with boundary layer theory [5]. Furthermore, for this mode, all the streamwise variation is contained in the shape function, meaning that the wavenumber of this mode is set to zero.

2.4 Preconditioning

In this section, the ideas of accelerating the PSE solution with a preconditioned iterative method are outlined. As the methodology relies more on engineering intuition than rigorous mathematical proofs, the discussion is merely set out to define the method and argue for its feasibility in a formal manner. First, the preconditioned linear system is defined; then, the direct solution of the linear systems of equation within the framework of iterative methods is discussed, which is directly applicable to the LPSE; these can be found in Section 2.4.1. Then, the block-preconditioners are introduced, which are applicable to the NLPSE, which is covered in Section 2.4.2. Finally, the generalization of the technique to the BRE is discussed in Section 2.4.3.

2.4.1 Linear PSE

We consider a general linear systems of equations, that, within the context of the present study, arise from the discretization of the LPSE, and is being solved over and over during a marching step until Eq. (10) is satisfied:
\[
M_j q_j = b_j.
\]

In the equation above, \( M_j \), \( b_j \) and \( q_j \) are the left-hand side (LHS), right-hand-side (RHS) and unknown vector in the \( j \)th iteration within a marching step. We also introduce the following terminology: the iteration which is denoted by the index \( j \) is referred to as PSE iteration or outer iteration, while the iterative solution of a single linear system of equations is called inner iteration or simply iteration.

In the framework of iterative methods, we define the right preconditioned system:
\[
(M_j P_j^{-1}) (P_j q_j) = b_j,
\]
where $P_j$ is the preconditioner. Then, the actual system solved is

$$M_j P_j^{-1} y_j = b_j,$$

and the original unknown is recovered from the solution $y_j$ of the modified system by solving

$$P_j q_j = y_j.$$

The primary reason for the introduction of the preconditioner is that it can accelerate the convergence of an iterative method. As also mentioned in the introduction, to enhance the convergence, the preconditioner requires two conditions to be satisfied: (i) its inverse need to provide a good approximation of the inverse of the LHS and (ii) it is easily invertible. The first criterion is quite intuitive: if $P_j^{-1} \approx M_j^{-1}$, then $M_j P_j^{-1}$ in Eq. (13) is close (in some norm) to the identity matrix; therefore, iterative methods converge fast. The second criterion is more practical. The iterative solution of Eq. (13) requires repeated application of $P_j^{-1}$. If $P_j^{-1}$ is expensive to evaluate, the solution will be slow, regardless the fact that the iterative solution converges in a few iterations.

There are three additional remarks for the sake of completeness. First, since both Eq. (13) and Eq. (14) require the application of the inverse of $P_j$, it is also customary to define $P_j^{-1}$ as the preconditioner. Second, it is noted that left preconditioning is also possible, when in Eq. (13) both the matrix $L_j$ and the RHS $b_j$ is left-multiplied with $P_j^{-1}$. Right preconditioning was chosen in this study as it is more efficient and simple for the proposed method. Third, the preconditioner, in general, can change during the inner iterations; however, in the present study it stays constant because of the nature of the proposed method.

Before the discussion of the proposed preconditioning method, the direct solution of the linear systems of equations with LU factorization is discussed. Within this framework, the preconditioner is the following:

$$P_j = L_j U_j,$$

where $L_j$ and $U_j$ are lower and upper triangular matrices arising from the LU decomposition $M_j = L_j U_j$. With these notations, it is trivial from Eq. (13) that $y_j = b_j$. This can be viewed as in the direct solution, zero iterations required with an iterative method. Then, the solution can be recovered from Eq. (14) by applying the inverse of the preconditioner only a single time, essentially solving the original equation.

This generalization of the direct solution technique to the iterative framework can rightfully be considered cumbersome, yes it has practical applications: in the PETSc library, which is primarily designed for iterative solution of sparse linear system, this is the way external direct solvers are interfaced. Nevertheless, the above outlined framework highlights two stages of the direct solution method, which will be essential to the performance of the preconditioner. The first stage is setting up the preconditioner - calculating the LU
factorization of $M_j$, which requires the largest computational effort. The second stage, application of the preconditioner, requires much less computational resources: it involves inverting the lower and upper triangular matrices, which is also called forward-backward substitution.

Using the above outlined definitions, the proposed preconditioning method is incredibly simple:

$$P_j = P = L_0 U_0.$$  \hfill (16)

This means that within each spatial PSE marching step, the preconditioner is only set up a single time, in the first PSE iteration, and it remains the same throughout the rest of the iterations within that marching step. This way, the most expensive part of the direct solution method - the calculation of the LU factorization - is performed only once. It is trivial that the application of the preconditioner is economic: application of the inverse using the backward-forward substitution is computationally efficient. Therefore, this preconditioner satisfied the second criterion: it is easy to evaluate (once it has been set up).

The other crucial question is how well the preconditioner can approximate the inverse. This is trivial in the first PSE iteration, when essentially a direct method is used. The performance of the preconditioner method can be assessed using the fact that it remains the same within each marching step. The dependence of the matrix on the solution parameters is the following: $M_i(U, \text{Re}, \omega, \alpha_i)$. During the PSE iterations in each marching step, only the wavenumber $\alpha$ is changed - every other quantity, and the structure of the system matrix remains the same. In the nondimensionalized equations, $U = O(1)$.

The auxiliary condition Eq. (10) absorbs the growth into the imaginary part of the wavenumber, which is typically $\alpha_i = O(10^{-2})$ for boundary layer problems. During the PSE iterations, if the slowly varying assumption holds, the change in the wavenumber is around a few percent, roughly $O(10^{-4})$ (e.g., see Fig. 3) for the values of the present study. The small variation of $\alpha$ strongly suggests that a previous LU factorization provides a good approximation of the inverse, especially since $\alpha$ is included only in a few terms of the matrix $L_i$. Under these considerations, the preconditioning technique is expected to improve the LPSE solution significantly.

2.4.2 Nonlinear PSE

Next, the preconditioning technique is extended to the NLPSE method. Two cases are discussed: the explicit treatment of the nonlinearities, when they appear in the RHS (as in Eq. (9)), or the implicit treatment, where the nonlinearities appear in the LHS. These two cases are referred to as explicit and implicit NLPSE, respectively. Furthermore, the following case is considered as an illustrative example. A single, primary mode with frequency $\omega$ generates two harmonics $2\omega$ and $3\omega$, and the zero-frequency mean-flow distortion (MFD). These modes are denoted with indices $q_1$, $q_2$, $q_3$ and $q_0$, respectively.

The explicit case is trivial: since the equations are decoupled, they can be solved individually, one-by-one, and all the observations made in the LPSE
Reusing LU factorization for efficient solution of the PSE

It is noted, however, that if the preconditioner method is effective for the LPSE, it is especially effective in the case of the explicit NLPSE. This is because the nonlinearity usually slows down the convergence of the PSE iteration. Therefore, a single LU factorization can be reused multiple times as a preconditioner within each marching step.

In a more general framework, the explicit NLPSE system can be collected into a large block-diagonal matrix, and treated as a new system:

$$
\begin{bmatrix}
L_{3,j} & 0 & 0 & 0 \\
0 & L_{2,j} & 0 & 0 \\
0 & 0 & L_{1,j} & 0 \\
0 & 0 & 0 & L_{0,j}
\end{bmatrix}
\begin{bmatrix}
q_3 \\
q_2 \\
q_1 \\
q_0
\end{bmatrix} = \tilde{b}.
$$  \hspace{1cm} (17)

In the above equation, \( \tilde{\square} \) is introduced to distinguish the NLPSE system from the LPSE. In this case, the preconditioner is the same matrix with \( j = 0 \). Then, the preconditioner can be set up and inverted by calculating the LU factorization of the diagonal blocks, same as when the systems are treated individually. However, the iterative linear solver is applied to the whole equation. In the case of block matrices, this preconditioner is called the block Jacobi (bJ) preconditioner with no overlap between the blocks \cite{67}. This is the generalizations of the well-known Jacobi iteration, which can be viewed as fixed-point iteration on the linear system preconditioned by the Jacobi preconditioners, which is the diagonal part of the matrix \cite{67}. In the block version of this preconditioner, the operations are performed on the matrix blocks instead of real/complex numbers. Therefore, the bJ preconditioner requires inversion of the matrix blocks instead of division of the diagonal entries of a matrix. For the sake of generality, this is how the bJ preconditioner is implemented in the present study.

Now our attention is turned to the implicit NLPSE using the same example. First, the structure of the equations is discussed; then, the preconditioning methods are covered. The equations are the same as Eq. (17), but the block structure of the matrix is different:

$$
\tilde{M}_j = \begin{bmatrix}
L_{3,j} + F_{(3,0),j} & F_{(2,1),j} & 0 & 0 \\
F_{(3,-1),j} & L_{2,j} + F_{(2,0),j} & F_{(1,1),j} & 0 \\
0 & F_{(3,-2),j} & L_{1,j} + F_{(1,0),j} & 0 \\
F_{(3,-3),j} & F_{(2,-2),j} & F_{(1,-1),j} & L_{0,j} + F_{(0,0),j}
\end{bmatrix}
$$  \hspace{1cm} (18)

In the above equation, \( F_{(m,n),j} \) denotes the nonlinear term arising from the interaction of the modes \((m, n)\), and negative indices denote the interaction with the conjugate mode. Note, that although most of the nonlinearity is contained in the LHS, the RHS also has some leftover from the nonlinear term that cannot be discretized implicitly; these terms are associated with the streamwise derivative of the unknown solution. Although the position of the
nonlinear terms unequivocally determines the implicitly discretized mode, it is also highlighted with the color red for the sake of clarity. The location of the nonlinear terms involving conjugation is predetermined (the conjugate terms must be implicit), but in the case of the positive modal interactions, there are two alternatives. Here, we choose to cluster the terms near the diagonal, that is, $F_{3,0}$ is in block $(1,1)$ instead of $(1,4)$, and $F_{2,1}$ is in block $(1,2)$ instead of $(1,3)$, etc. This choice includes lower modes, which generally have higher amplitudes, in the matrix, which is likely to enhance robustness.

The preconditioning of the implicit NLPSE is also possible ‘naively’ as in the case of the LPSE: the LU factorization of the matrix $\mathbf{M}_0$ could serve as a preconditioner. However, because of the large number of modes, the size of the matrix can be very large, and LU factorization of the system can require a large amount of computation and memory. Therefore, preconditioners based on the block-structure of the system are investigated. As just discussed, the simple bJ preconditioner may be applicable to the implicit NLPSE just as in the explicit case: $\mathbf{P}$ consists of the diagonal blocks of Eq. (18), and its inverse can be applied economically after a single LU factorization of the diagonal blocks. Furthermore, similar to the bJ method, the block Gauss-Seidel (bGS) preconditioner can also be applied. The bGS preconditioner is formed from the block upper/lower triangular part of the matrix. Therefore, inversion of the preconditioner is just block forward/backward substitution of the preconditioner, which only requires the inversion of the diagonal blocks, in the same way, as the inversion of a triangular matrix requires division only with the diagonal entries.

Turning our attention to the efficiency of the implicit discretization, the following remarks can be made. In the above-outlined formalism, the mode amplitudes scale as $O(10^{-1})$, $O(10^{-1})$, $O(10^{-2})$, $O(10^{-3})$ for the MFD, base mode and higher harmonics. This is because a single initial mode generates the harmonics through the nonlinear interaction, and that the MFD can accumulate large amplitudes over long spatial distances. This implies the bGS preconditioner outperforms bJ if the nonlinearity is high, since it also contains off-diagonal terms that capture the interaction of higher harmonics. However, in the case of low or moderate nonlinearity, the bJ method may also provide a good approximation of the inverse, as it contains the terms associated with the base flow (and the mean flow in the case of the implicit NLPSE), which contains most of the weight.

This section is concluded with a few additional remarks regarding the preconditioning of the NLPSE, and the iterative method.

- Note that in the case of the bGS preconditioner, the diagonal blocks are calculated for $j = 0$; however, the off-diagonal blocks are updated during the outer iterations.
- In the present study, we choose the lower triangular part, which includes the interaction of conjugate terms, as more modal interaction is included in the preconditioner this way. Note that the structure of the equations is selected
based on the above argument and the fact that PETSc can only invert the lower-triangular part of a block-structured matrix.

- In the case of the explicit NLPSE, implicit discretization of the diagonal block is known to enhance the robustness of the method, while the equations remain decoupled [4, 30]. Although this was not investigated, the proposed preconditioner method can be applied to this modified explicit NLPSE. Furthermore, it is expected to perform well due to the scaling assumptions discussed above.
- Although the above discussion is for a single mode and its harmonics, the formalism can be naturally extended to narrowband instabilities, where multiple base modes and their first harmonics are considered [78].
- In general, when block preconditioners are applied to very large systems, the inversion of the diagonal block can also be approximated using iterative solution techniques.

2.4.3 Generalization of the preconditioner method: application to the BRE

Finally, we note that the effectiveness of the reused LU factorization as a preconditioner relies only on a simple presumption: the linear systems of equations change only slightly during the successive solutions. This naturally raises the question: is the preconditioning technique more general? Our belief is that the answer is yes. As a test of this hypothesis, the preconditioner method is applied to the BRE; the results are discussed in Section 4.4. The nonlinear systems of equation that arises from the discretization of BRE in each marching step, can be solved with the Newton-Raphson method. This requires repeated application of the Jacobian’s inverse, which is typically calculated with LU factorization. It is proposed that within each marching step, the first Newton-Raphson iteration is done using an LU decomposition; then, in successive iterations, only the Jacobian is updated, and it is solved using the iterative method, preconditioned by the reused LU factorization. Note that a well-known trick that may accelerate the Newton-Raphson method is reusing the same Jacobian for multiple iterations; however, this decreases the order of convergence [80]. The advantage of the proposed technique is that the most time-consuming step, the solution of the equations is accelerated, but the Jacobian is updated - therefore, the second-order convergence of the solution of the nonlinear equations is preserved.

3 Numerical methods

The equations are discretized using the open-source finite element library FreeFem [71]. The parallel version of FreeFem is used, in which the computational mesh is decomposed into multiple overlapping parts, and each part is associated with a single process. For details regarding the implementation of the domain decomposition formalism, see [81, 82]. Unless otherwise stated,
the calculations were performed on 4 processor cores. The equations are discretized using a vectorial finite element space structure that contains all the velocity components and the pressure - in the case of the NLPSE, each block is associated with a single vectorial finite element space. The vectorial finite element space automatically enforces periodicity in the spanwise direction for all its components by built-in FreeFem routines. In most of the calculations, standard Taylor-Hood elements are used - $P_2$ elements for the velocities and $P_1$ elements for the pressure. In Sections 4.3.2 and 4.4, we also investigate the methods using $P_4 - P_2$ elements to test the numerical techniques for a more dense matrix. Higher polynomials are not used, since currently $P_4$ is the highest Lagrangian finite element implemented in FreeFem.

We note that in the case of $P_4$ elements, using the standard periodic boundary condition in FreeFem, we experienced numerical instabilities near the spanwise boundaries, where periodicity is prescribed as the boundary condition. This issue could be resolved by prescribing the appropriate symmetry of the velocities explicitly as boundary conditions. These numerical instabilities are surprising, as Broadhurst and Sherwin [51] also used $P_N - P_{N-2}$ for plane marching PSE and did not experience any instability. Furthermore, in high-order finite element methods for fluid mechanics, the $P_N - P_{N-2}$ finite element combination is known to be stable. An obvious difference between the two implementations is that in [51] studied a channel flow in which all the boundary conditions are homogeneous. It should be addressed in future studies whether this instability is due to the implementation of the periodic boundary conditions in FreeFem or related to the choice of the finite element spaces and the boundary conditions. A final remark regarding this is that augmenting the finite element spaces of the velocity with bubble function is known to stabilize the solution of the incompressible Navier-Stokes equations [83], which implies that this may stabilize the PSE solution. However, unfortunately, in FreeFem, Lagrangian finite elements with bubble functions are only available for $P_1$ and $P_2$ elements, and not higher.

In the parallel version of FreeFem, generally, the left-hand side (LHS) and right-hand-side (RHS) assembly is done using FreeFem routines. At the same time, the solution of the linear and nonlinear equations and eigenvalue problems, and matrix-vector multiplications are handled by PETSc [72, 73], and SLEPc [74]. The FreeFem-PETSc interface provides an amazing numerical tool that offers both convenience and effectiveness, but unfortunately, not all PETSc functions are interfaced. This is because PETSc has been developed much more intensively than FreeFem, and therefore, the development of the interface is user-driven. This limited the optimization of the code to some extent - these limitations are stated explicitly throughout the paper. The distinction between FreeFem and PETSc routines is important since the state vectors have different numbering in FreeFem and PETSc. FreeFem treats the overlapping parts separately, while in PETSc, the variables at the overlapping parts of the computational domain are associated with a single process (Pierre Jolivet, private communication).
In the LPSE, the LHS and RHS assemblies are done in FreeFem using the command varf (VARiational Formulation), which is a high-level command that allows the coding of the matrix terms essentially the same way as written on paper. The generality of this formulation makes it expensive to compute, as functions specified in the variational form (e.g., the base flow in the PSE matrix) are evaluated at the corresponding points. This means that in the LHS assembly, the underlying information that the base flow is a vectorial finite element function is not used. This will be highlighted in the next section. Therefore, in the NLPSE implementation, considerable attention has been paid to minimizing the cost of matrix and RHS assembly to demonstrate the effectiveness of the finite element formalism. Different matrices are created that do not need to be altered during the calculation (mass matrices, differentiation matrices associated with the viscous terms and convective terms, etc.), and these are converted into PETSc numbering. Using these matrices, in the NLPSE calculations, the RHS is always evaluated with successive PETSc matrix-vector and vector-vector multiplications. In the case of the NLPSE, the LHS is assembled from individual block that are created in FreeFem numbering. To reduce the computational cost, in the case of the block Jacobi preconditioner, a 'matrix-free' version was implemented. In this approach, the diagonal blocks of the LHS are formed only once using the command varf to calculate the preconditioner. Then, the action of the LHS is applied by successive multiple matrix-vector and vector-vector multiplications. This approach significantly reduces the time spent with matrix-forming, but the action of the matrix is more expensive to calculate due to the multiple successive multiplications. It could be possible to form a total LHS matrix completely in PETSc numbering which would significantly reduce the computational cost, but this is not possible currently because of the limitations of the FreeFem-PETSc interface. Nevertheless, the numerical experiments will demonstrate that the 'matrix-free' approach could significantly reduce the computational time. Finally, the timing of the different run stages, and the memory consumption is monitored through PETSc routines.

Next, numerical aspect of the solution of the BRE, BiGlobal stability equations and the PSE are discussed in successive order.

In the case of the BRE marching procedure, the equations were discretized using a second-order backward Euler scheme in the streamwise direction (except for the first step, when the first-order backward Euler method was used). Roughly two hundred grid points were used in the streamwise direction to ensure that this streamwise resolution does not affect the PSE results. Furthermore, this also ensures the slow variation of the solution between successive marching steps, which is the key requirement of the preconditioner method. The stepsize varied in the streamwise direction, following the boundary layer scaling. The nonlinear system of equations resulting from the discretization of BRE is solved using a second-order line search (generalized Newton) method using the SNES library within PETSc. The absolute tolerance was $5 \cdot 10^{-13}$,
Reusing LU factorization for efficient solution of the PSE

The relative tolerance was $5 \cdot 10^{-13}$, or the norm of the change in the solution is less than $10^{-4}$. For the precise definition of the tolerances, the reader is referred to the PETSc manual [73]. As explained in Section 2.4, the preconditioner method was also tested for the BRE. In the case when an iterative linear solver is requested, in the first five downstream marching steps, where initial transients in the solution may be expected, LU factorization is used to invert the Jacobian. Then, in each marching step, the LU factorization of the Jacobian is calculated for the first line search (Newton-Ralphson) iteration, and this LU factorization was reused in the following linear equation solution within that marching step.

The numerical aspects of the BiGlobal stability equations are covered briefly. The generalized eigenvalue problem arising from the BiGlobal stability equations is solved with the standard Krylov-Schur algorithm combined with the shift invert technique, similarly to [39]. This is done using the SLEPc PEP library [84], which requires only the coefficient matrices of the polynomial eigenvalue problem, and handles the rest of the operations. A Krylov subspace of dimension ten was used to find the eigenmode, which serves as the initial condition of the PSE solution.

In the case of the PSE, the streamwise discretization is the same as in the case of the BRE: first and second second order backward Euler method in the first step and the rest, respectively. The streamwise PSE grid is non-equidistant, and the grid points are clustered near the leading edge so that they follow the boundary layer scaling; the exact number of grid points, along with the physical parameters, is specified in the next section. In the PSE equations, the wavenumber was updated according to the formula [5, 79]:

$$\alpha_{n,j+1} = \alpha_{n,j} - \frac{i}{\int \int \mathbf{q}_{*}^n \frac{\partial \mathbf{q}_n}{\partial x} dydz}{\int \int \mathbf{q}_{*}^n \mathbf{q}_n dydz},$$

(19)

where $\alpha_{n,j}$ is the streamwise wavenumber of the $n$th mode at the $j$th iteration. The iteration continues until the relative difference in $\alpha_{n,j}$ between successive iterations is $10^{-8}$, and the maximum number of iterations is set as 30 to limit the computational time when using a low number of processes.

In the NLPSE solution, it is customary to track only those modes among the requested ones with a sufficiently high amplitude (e.g., [24]). This can substantially decrease the size of the problem and accelerate the computation. In the present implementation, all the modes are tracked during the computation. This method allows more controlled monitoring of the computational time and resources with respect to the size of the system, although this implementation was motivated by coding considerations (related to the creation, updating, and destruction of PETSc matrix objects) and may increase the computational time. As mentioned in Section 2.4.2, in the solution of the explicit NLPSE method, all the modes are combined into a block-diagonal matrix, which is preconditioned by a bJ preconditioner. Recalculation of the bJ preconditioner in every PSE iteration corresponds to using a direct solver.
Direct solution of the linear systems of equations was performed using the sparse multifrontal LU solver MUMPS [85, 86]. A sparse direct solver consists of four steps [67]: (i) preordering to reduce fill-in (the nonzero elements of the factorization), (ii) symbolic factorization (the factorization is computed only symbolically, without the numerical values), (iii) numerical factorization, during which the actual LU decomposition is calculated, and (iv) forward and backward substitution. As mentioned in the previous paragraph, since all the modes are tracked through the whole calculation, the matrix sizes are exactly the same during the NLPSE calculation; this is trivial in the case of the LPSE. Furthermore, the sparsity pattern of the individual block matrices associated with the PSE modes remains the same. This means that the symbolic factorization is the same during the whole PSE solution, meaning step (ii) is required to be calculated only once. Since the appropriate options cannot be set in every case due to the present limitations of the FreeFem-PETSc interface, this factorization is recalculated every time. However, the time of the symbolic factorization can be very precisely monitored with PETSc, and it was found that it takes about 10% of the system solution. Although this is substantial, it will be shown that this is small compared to the computational cost reduction achieved by the proposed preconditioning technique. Furthermore, the standard options regarding step (i), the preordering, were not changed. Finally, we note that only step (iv) is executed when the preconditioner is applied.

The flexible Generalized Mean Residual methods (fGMRES) [87] is applied as the iterative solver. Although other solution techniques are available, and no other method was tested, this method was successfully used for similar flow stability problems [68–70]. The tolerances of the iterative solver are displayed in Table 1. The tolerances were tuned based on trial-and-error approach. Note that the only large difference is in the relative solver of the BRE. The reason for this is that as the nonlinear systems of equations are solved iteratively, the previous solution is used as an initial condition for the inner linear iterative solver. This option enables converging the inner linear and outer nonlinear solvers together, i.e., initially, a less accurate solution of the linear system can also yield a sufficiently large step in the right direction. Note, however, that for the BRE equations, the first nonlinear iteration step is always taken with the exact solution of the system; then, the rest of the linear systems are solved using the previous LU factorization as a preconditioner.

<table>
<thead>
<tr>
<th>Tolerance</th>
<th>BRE</th>
<th>PSE</th>
<th>NLPSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute</td>
<td>$10^{-12}$</td>
<td>$10^{-12}$</td>
<td>$10^{-11}$</td>
</tr>
<tr>
<td>Relative</td>
<td>$10^{-5}$</td>
<td>$10^{-10}$</td>
<td>$10^{-12}$</td>
</tr>
</tbody>
</table>

Table 1: The tolerance of the linear solver fGMRES. For the precise definition of the tolerances, the reader is referred to the PETSc manual [73].
The computational mesh was constructed using Delaunay triangulation using the BAMG program \cite{88}. A typical mesh is displayed in Fig. 1. The mesh consists of four regions, and the mesh size is controlled by specifying the number of elements at the edges of the mesh segments. The parameters of the meshes used in the study are specified in the following sections. The implementation of the different solvers was verified by reproducing the results of Siconolfi et al. \cite{41} and Martín and Paredes \cite{42} and Paredes et al. \cite{52}.

To study the performance of the preconditioning technique, we examine the different stages (solution of the linear systems of equations, LHS and RHS assembly, and the remaining operations (rest)) of the operation for the whole PSE run. This is chosen since the number of iterations to find the wavenumber determines how many times the LU factorization can be reused as a preconditioner. In a similar fashion, when the memory requirement is compared between the different methods, only the peak memory consumption is examined. All the calculations were run on a personal computer with an AMD Ryzen 9 3900X 12-Core Processor and 32 Gb of RAM. Both FreeFem and PETSc were compiled from the source code, using a gcc compiler with standard options.

![Fig. 1: The mesh configuration used in sec. 4.3.1. (a) full computational domain, (b) the lower part of region of the computational domain zoomed-in. The red lines in Fig. (b) denote the boundary layer thickness (the wall normal location, where the streamwise velocity is 99% of the free-stream velocity) upstream and downstream ends of the computational domain.](image)
4 Results

4.1 Problem setup

We consider an incompressible zero pressure gradient flat plate boundary layer disturbed by idealized vortices. The numerical experiments of the proposed computational methods are performed on the same configuration proposed by Siconolfi et al. [41] and was also studied by [42]. At a distance \( \hat{x}_0 \) from the leading edge, counter-rotating Rankine vortices are introduced into the Blasius boundary layer, which generate streamwise streaks through the lift-up mechanism as the boundary layer evolves downstream [75]. The streamwise velocity in the free stream is \( \hat{U}_0 = 7.7 \) m/s, and the kinematic viscosity is \( \hat{\nu} = 1.46 \cdot 10^{-5} \) m\(^2\)/s; all quantities in the equations and the results are nondimensionalized with the boundary layer scale \( \hat{\delta}_0 = \sqrt{\hat{L}_0 \hat{\nu} / \hat{U}_0} \) at the location \( \hat{L}_0 = 52 \) mm, where the vortices are introduced into the boundary layer. The key parameters of the configuration are displayed in Fig. 2a. The distance between the counter-rotating vortices is \( \hat{l}_v = 3.12 \) mm, the vertical distance between the vortex center and the plate is \( \hat{h}_v = 3.9 \) mm, and the spanwise periodicity is \( \hat{\Lambda} = 13 \) mm. Rankine vortices are used as idealized vortices:

\[
\hat{v}_\theta = \frac{\hat{\Gamma}}{2\pi} \begin{cases} \hat{r}/\hat{r}_0^2, & \text{if } \hat{r} < \hat{r}_0, \\ 1/\hat{r}, & \text{if } \hat{r} > \hat{r}_0, \end{cases}
\]

where the circulation of the vortices is denoted with \( \hat{\Gamma} \), and the vortex center radius is denoted with \( \hat{r}_0 \). In the study, we use \( \hat{r}_0 = 1.3 \) mm and \( \hat{\Gamma} = 2 \cdot 10^{-4} \) m\(^2\)/s. Note that this value is lower than \( \hat{\Gamma} = 9 \cdot 10^{-4} \) m\(^2\)/s used in [41]. The reason for the choice of the value of \( \hat{\Gamma} \) is that a strongly three-dimensional boundary layer is generated, but the stabilization of the TS waves is only mild. Therefore, so high amplitude nonlinear TS waves can be generated to test the proposed method in a region where the PSE solution is problematic. The streak evolution is depicted in Fig. 2b, where the streak amplitude is calculated using both the difference-based [41] and integral based [89] amplitude definitions:

\[
A_{ST}^{\text{int}}(x) = \frac{1}{\hat{U}_0} \int_{-0.5}^{0.5} \int_0^{\eta_1} |U(x, \eta, \zeta) - U(x, \eta)| \, d\eta d\zeta, \quad (21)
\]

\[
A_{ST}^{\text{diff}}(x) = \frac{1}{2} \max_y \left[ \max_z \{U(x, y, z)\} - \min_z \{U(x, y, z)\} \right]. \quad (22)
\]

In the equation above, \( \eta = \hat{y}/\hat{\delta}(x) \) is the wall-normal coordinate scaled with the local boundary layer scale, and \( \eta_1 \) is a sufficiently high wall-normal location where the boundary layer modulation vanishes, and \( \zeta = \hat{z}/\hat{\Lambda} \). The difference-based streak amplitude is around \( 0.04 - 0.08 \) in the \( \text{Re} \in [500 - 900] \) range, where TS waves are unstable at the examined frequency. This difference-based amplitude range was found to stabilize the flat-plate boundary layer in the experimental study of Fransson et al. [90]. The integral amplitude range is
around $0.07 - 0.14$; in the study of Shahinfar et al. [89] in the case of configuration C09 this amplitude range was found to mildly stabilize the boundary layer. Therefore, this case is expected to include aspects that make the plane marching PSE challenging - full three-dimensionality and large instability amplitudes due to the weak stabilization in the NLPSE.

(a) Parameters of the idealized vortices. (b) Streak amplitude evolution as function of the Reynolds number as the nondimensional streamwise coordinate. Blue: integral amplitude; Red: difference based amplitude. The downstream end of the domain in the PSE calculations is displayed with the dashed vertical line, and the streak evolution is displayed in a longer streamwise domain.

The end of the calculation is at $\hat{x}_1 = 1.43823$ m, which corresponds to a $\hat{\delta}_1/\hat{\delta}_0 \approx 5.3$ growth in the boundary layer thickness. This requires a large region of the computational mesh to be refined; therefore, the large system size renders the computation numerically challenging to assess the capability of the proposed method. The change in the region of interest is also illustrated in Fig. 1b by the red lines, which show the boundary layer thickness $\hat{\delta}_{99}$ at $\hat{x}_0$ and $\hat{x}_1$.

The mesh used in this and the following section is displayed in Fig. 1. It consists of 9981 triangles and 5129 vertices, which results in a total number of 65286 unknowns for each velocity-pressure field. The mesh is sufficiently fine so that further mesh refinement did not increase the accuracy of the solution. More precisely, even in the case of the NLPSE with the highest nonlinearity, refining the mesh changed the mode amplitudes by less than 1%. Although this mesh may be considered overrefined, in general, overrefined meshes are needed since, within parameter studies, large unexpected variations in a disturbance
function may arise. Furthermore, when multiple frequencies are considered, the spatial support of the different frequency shape functions covers a large domain, meaning a suitable mesh should be fine to capture this behavior.

The instability waves are initiated at nondimensional frequency $F = \hat{\omega}/\hat{U}_0^2 \cdot 10^6 = 86$ at streamwise location $\hat{x} = 0.3035$ or equivalently, $Re = 400$ - a case which was also studied by [2, 52]. In the streamwise distance, the final location is reached in 50 marching steps. In the case of the LPSE, different growth rates can be defined based on different quantities [5]; here, the energy-based growth rate is used:

$$
\sigma_E = -\alpha_i + \frac{\partial}{\partial x} \ln \left( \sqrt{E} \right), \quad E = \int_{z_{\min}}^{z_{\max}} \int_{0}^{y_{\max}} (uu^* + vv^* + ww^*) \, dy \, dz, \quad (23)
$$

where $E$ is the nondimensional kinetic energy of the instabilities. In the case of the NLPSE, mode-by-mode amplitude measure similar to Haynes and Reed [24] is adopted:

$$
u_{RMS,max} = \max_y \left\{ \frac{1}{\Lambda} \int_{z_{\min}}^{z_{\max}} \left| A_n u_n(x, y, z) \right| \, dz \right\}, \quad \begin{cases} a = 1, & \text{if } n = 0, \\ a = \sqrt{2}, & \text{if } n \neq 0. \end{cases} \quad (24)
$$

where $A_n = \exp(i \int_{z_{\min}}^{z_{max}} \alpha_n(\xi) \, d\xi)$. This, in the two-dimensional case, reduces to the amplitude definition used by Paredes et al. [52].

### 4.2 Linear PSE

In the case of the LPSE, we aim at demonstrating that the proposed preconditioned iterative method is more efficient than LU factorization, and it also outperforms other preconditioners.

As reference, off-the-shelf preconditioners are used, and it is argued that a significant speedup cannot be expected from other state-of-the-art preconditioners such as the augmented Lagrangian preconditioner [68]. Two preconditioners are compared to the reused LU factorization: the additive Schwarz method (ASM) with LU factorization as a subdomain solver and the bJ preconditioner. Both of these are readily available in PETSc. The additive Schwarz method was used as a preconditioner for the velocity blocks in [68, 69], and LU factorization with MUMPS was chosen as a subdomain solver, similarly to the previously mentioned studies. In the bJ preconditioner, PETSc automatically selects the blocks, unlike in the preconditioner described in Sec. 2.4 where the blocks of the system correspond to different harmonics. (It is noted that selecting the block corresponding to the velocities and the pressure would be a more natural choice, as it aligns with the structure of the equations. However, this requires circumventing the singularity, which is associated with the last diagonal block being zero due to the incompressibility [68].)

First, briefly the wavenumber and the growth rate are examined to display the degree of modification in the boundary layer. Fig. 3a displays the evolution of the real part of the streamwise wavenumber and the growth rate of the
disturbances based on the kinetic energy of the perturbations. It is apparent from the data that only a mild moderation of the instabilities can be observed, as expected from the low streak amplitudes. Although it is not shown, all the different solvers yielded the same accuracy; e.g., the difference between the wavenumbers was $O(10^{-10})$. This is expected because of the tolerances of the solvers.

![Graphs](image)

**Fig. 3**: Evolution of the (a) real part of the streamwise wavenumber, (b) growth rate defined with Eq. (23) as functions of the Reynolds number.

The total computational time is displayed in Fig. 4a, where the blue color denotes the time spent with the solution of the linear system of equations. Red and yellow colors denote the LHS and RHS assembly, and purple denotes the rest of the computation (e.g., I/O operations, updating the wavenumber using Eq. (19) (which required the calculation of two integrals), etc.). LU, LUPC, denote LU factorization (direct solver), reusing LU factorization as a preconditioner, respectively; the other two notations were already introduced. IC0 and IC1 denote whether the iterative solver was initialized with zero initial condition or from a previous solution, respectively. Note that the number of PSE iterations is not shown, except for the first marching step, where four iterations were required, three iterations were sufficient to determine the wavenumber that satisfies the auxiliary condition.

Examining LU factorization as a reference, it is apparent that most of the time is spent with the solution of the linear system of equations (approximately 67%, two-thirds of the computation), while RHS assembly along with auxiliary operations is marginal (less than 5%). Interestingly, the LHS assembly takes a considerable time. This is due to the fact that, as explained in Section 3, the LHS assembly requires repeated interpolation of the base flow due to the convenient but inefficient FreeFem varf routine. The computational time of the LHS will be analyzed in more detail in the next section. Moving our attention
Reusing LU factorization for efficient solution of the PSE

Fig. 4: Total solution time (a) and mean number of iterations to solve the linear systems of equations (b) in the LPSE solution. The abbreviations are the following: LU: LU factorization; LUPC: LU factorization reused as a preconditioner, ASM: additive Schwarz method, bJ: block Jacobi preconditioner. IC0 and IC1 denote whether iterative solver was initialized with zero initial condition, or from a previous initial condition.

to the different solvers, it is apparent that every operation takes about the same time; only the solution time changes, as expected. It stands out from the results that the proposed method (LUPC) requires 60% less time to solve the linear systems of equations. Using this preconditioner, starting the iterative solver from the previous initial condition has a negligible effect on the total computational time. Since the number of PSE iterations is the same in each case, it can be estimated in this case how much faster the preconditioned solver is than LU factorization. These calculations show that a system solution using LU factorization takes approximately 1.1396 s, while with zero or nonzero initial condition the iterative solver takes 0.0890 s and 0.0515 s, respectively - a speedup by roughly a factor of 13 and 22! This means that in the case of the LUPC method, the most time-consuming step of the solution of the linear systems of equations is setting up the preconditioner - calculating a single LU factorization. This is the reason why although the initial condition has a significant effect on the performance of the iterative method, it does not have a considerable effect on the total computational time. Conversely, the iterative method with the off-the-shelf preconditioners (ASM and bJ) more than doubles the computational time compared to the LU factorization. In the case of these preconditioners, the initial condition has a more significant effect on the computational time.

The reason for the behavior of the different preconditioners can be interpreted by examining the average number of iterations per linear solution, which is displayed in Fig. 4b. In the case of LUPC, as the preconditioner provides an excellent approximation of the inverse, therefore the solver converges only
in a few iterations, 2 – 5 iterations. In contrast, the off-the-shelf preconditioners require 100-200 iterations, as they approximate the inverse of the system matrix much more poorly. These observations clearly display the power of an effective preconditioner.

Finally, the use of more efficient preconditioners must be addressed, which we did not investigate in the present study due to the cumbersomeness of their implementation. We compare the present result with those by Moulin et al. [68], who proposed the augmented Lagrangian preconditioner, which may be considered state-of-the-art. Although they applied it to the linearized Navier-Stokes (LNS) equations, we believe the comparison with their result for a 2D wake flow geometry discretized with $\approx 14500$ triangles and Taylor-Hood finite elements can yield relevant conclusions. The two reasons for this are (i) the PSE and LNS systems are very similar, and (ii) the sizes of the two problems are roughly the same. In [68] the lowest number of iterations they reported is 56. Even if their preconditioner is really cheap to evaluate, it is not expected to outperform either LU factorization or the LUPC preconditioner as a preconditioner based on the results of the standard preconditioners. Furthermore, if we consider that in [68] a relative tolerance of $10^{-6}$ was used, which is much higher than that used in the present study, we can infer that it is extremely unlikely that any previously proposed preconditioner can beat reusing LU factorization for the PSE system.

Summarizing the investigation of the LPSE system, we can draw the conclusion that the proposed preconditioner outperforms all other tested solution methods. Furthermore, it can be inferred that in the case of the NLPSE method, an even larger speedup may be expected because of the proposed preconditioner dramatically decreased the solution of a single linear system. In the LPSE, the number of PSE iterations was low, so relatively few LU factorizations were spared. However, in the case of the NLPSE method, typically a larger number of iterations is required, which implies a more significant speedup can be achieved.

### 4.3 Nonlinear PSE

In this section, the LUPC preconditioner is investigated for the NLPSE. First, in Section 4.3.1, we investigate how the NLPSE implementation scales with the number of processes. Then, in Section 4.3.2, we investigate, using four processes, how the implementation scales with the mesh size. In these studies, initially, a single base mode is the initial condition, which generates higher harmonics through nonlinear interaction. Two cases are considered: case (0) has an initial amplitude $A_0 = 0.002$ as defined in Eq. (24), and two higher harmonics (plus the MFD). In case (1), the initial amplitude is $A_1 = 0.01$, and the higher frequency modes are considered up to the sixth harmonic. These two cases are set up to assess the performance of the preconditioner method with a different number of modes and level of nonlinearity. As pinned down in the introduction, we aim at setting up test cases to evaluate the effectiveness of the preconditioning method, not at accurately modeling a transition scenario
to learn more about the physical phenomena. Since, as it was noted in the previous section, using the previous solution did not have a large effect on the total computational time, we initialized the iterative solvers with zero initial condition. This choice was also motivated that not all PETSc routines were interfaced with FreeFem, therefore this did not require additional code development the implementation.

The different cases are denoted as follows. Regarding the treatment of the nonlinear terms, ex. stands for explicit, im. for implicit discretization. The notations bJ and bGS have already been introduced. Finally, MF stands for matrix-free implementation. In this case, the matrices (in the implicit case, the diagonal blocks) are formed only a single time using the standard FreeFem commands in each marching step to calculate the preconditioner. Then, in successive PSE iterations, the action of the updated systems is calculated using successive matrix-vector and vector-vector multiplications to reduce the computational cost, as discussed in Section 3.

The evolution of the mode amplitudes is displayed in Fig. 5. In both cases, since the initial mode amplitude, and therefore the nonlinear interaction is significant, high harmonics amplitudes build up. In case (0), the fundamental mode (ω) amplitude of 0.015 and MFD amplitude of 0.026 is reached, while in case (1), these values are 0.05 and 0.15, respectively, which can be considered very high within the PSE framework. As the MFD builds up, after some time, the higher mode amplitudes saturate, and they start to decay. In Fig. 5b, at the amplitude plateau, it is likely that turbulence would develop, but this cannot be captured in the NLPSE equations due to the finite number of discrete modes. Furthermore, in Fig. 5b, the vertical dashed line denotes the location at which computations using explicit treatment of the nonlinear term blow up. This is because the explicit treatment of the nonlinear terms is significantly less robust than the implicit one, especially without using additional stabilization techniques discussed in the introduction. Therefore, the explicit run in case (1) is conducted on a shorter streamwise domain.

4.3.1 Scaling with the number of processes and general observations

First, the explicit case is examined. In Fig. 6, the computational time is displayed using the same colors as in Fig 4a, which will be continued throughout this paper. The first row shows the lower amplitudes (case (0)), and the second row shows the larger amplitudes (case (1)). What stands out from examining the change of the computational time with the number of processes is that the scaling flattens out when using more than six processes. This is because the size of the computational mesh is not large; therefore, communication between the processes is substantial compared to the time spent on the actual calculations. This implies that it may be more efficient to assign each mode to a single process instead of distributing the variables between the processes based on mesh regions. The next notable thing is that, once again, the solution of the linear system and the LHS assembly takes most of the computational time.
Comparing the direct and iterative method, the direct solution takes 5-7 times as much time as the iterative solver, which yields roughly a 50% reduction in the total solution time. Comparing the matrix-forming and matrix-free iterative methods, although the LHS assembly time is drastically reduced, the solution time is increased quite substantially. The reason for this is that evaluation of the system matrix is more expensive as its action is calculated using multiple successive operations - a limitation of the implementation due to the incompleteness of the FreeFem-PETSc interface. Nevertheless, even with these limitations, an additional factor of two speed-up could be achieved using the matrix-free formulation.

Next, the implicit case is investigated. Here, since the computational time difference varies significantly between the different methods, a logarithmic scale is used for the $y$ axis. First, the total time is examined; then, the equation solution and LHS assembly are monitored. The time spent on the RHS assembly and additional operations are marginal compared to the other computational tasks; therefore these are not shown. The total computational time for the implicit case is displayed in Fig. 7. Once again, when examining the dependence of the computational time on the number of processes, it is apparent that the scaling flattens out when using more than fix processes. The computational time is roughly the same when using 8 and 10 processes in the case of the iterative solvers. This is not investigated further; it is hypothesized that this is related to the partitioning of the mesh. When comparing the direct solvers with the iterative ones, the runs using the iterative solver took roughly
Fig. 6: Computational time of the NLPSE cases with explicit treatment of the nonlinear terms. The first row is case (0), while the second row is case (1). The coding of the colors is the same as in Fig. 4a.

one-tenth of the ones using LU factorization in case (0), and one-twentieth in case (1), which can be considered an outstanding speedup. In the case of the matrix-free approach, which intends to reduce the computational time associated with the LHS assembly, a factor of twenty and thirty reductions are the computational time compared to the direct solver in cases (0) and (1), respectively. Interestingly, there is no significant difference between the bJ and bGS preconditioners - this is likely because although the bGS preconditioner provides a better approximation of the inverse, it is also more expensive to evaluate. Because of the convergence issues of the explicit NLPSE in case (1), the performance of explicit and implicit methods is compared only in case (0). In case (0), the total solution time using implicit bJ and bGS preconditioners was roughly 1.6-1.7 times the explicit bJ case; comparing the matrix-free cases, this ratio was 1.7-1.9. This ratio was lower when a higher number of processes was considered. This implies that explicit treatment of the nonlinear terms is favored; however, case (1), where the explicit runs diverge with the higher amplitudes, displays that the robustness of the implicit method may be beneficial.
Next, the computational time associated with the separate parts of the computation is examined to gain more insight into the performance of the methods. Fig. 8 displays these values for case (0): the first and second row shows the equation solution time LHS assembly time, respectively. Case (1) is not shown, as similar trends can be observed in the two cases. Similar observations can be made as in the explicit case. The solution time is significantly lower for the iterative cases than for the direct solver, and the solution time for the matrix-free case is increased multiplication with the matrix becomes more expensive to evaluate. Quantitatively comparing the solution times, a factor of 30-60 and 75-95 reduction was observed in cases (0) and (1) (not shown), respectively. Interestingly, a more significant solution time reduction is associated with a lower number of processes, which implies the LU factorization scales relatively better than the iterative method.

Continuing the evaluation of the preconditioners, we compare the number of iterations it took to solve the linear systems of equations. The number of iterations are averaged as each marching step, so the effect on nonlinearity can be examined on the preconditioner, which is displayed in Fig. 9. Note that this number was the same, regardless of the number of processes used, as expected.
Fig. 8: Linear equation solution time (first row) and LHS assembly time (second row) of the NLPSE cases with implicit treatment of the nonlinear terms, case (0). Note that the y axis is logarithmic.

Focusing first on case (0) (Fig. 9a), we see that as we move downstream, initially, the different methods perform the same way, but as we advance, and the mode amplitude increases (see Fig. 5a), the behavior of the different methods diverges. What stands out from the figure is that the explicit case requires a lower number of iterations. This is due to the fact that in the explicit case the LHS does not contain any of the nonlinearities: the LHS changes only through the change in the wavenumber, which is quite low. In the case of the implicit method, since the nonlinear interaction is captured in the system matrix, the number of iterations required to solve the system follows the level of nonlinearity: the higher the nonlinear interaction, the more iterations are required to solve the system. In the implicit case, it can also be observed that the bGS method requires a significantly lower number of iterations than the bJ method, as the preconditioner carries most of the nonlinearities contained in the lower triangular part of the LHS (see Eq. (18)). Although this would imply that the bGS preconditioner yields a lower computational time than the bJ preconditioner, this is counterbalanced by the fact that the bGS preconditioner is more expensive to evaluate. This is the reason why the two methods yield the same computational time. Although these observations were made examining
Fig. 9a, the same trends can be observed in Fig. 9b, which support these findings.

![Fig. 9](image)

**Fig. 9:** Average number of iteration per marching step as a function of the streamwise distance. (a): case 0; (b): case 1.

Monitoring the effect of the preconditioner on the solution time more precisely, the time of each marching step is examined as a function of the streamwise distance. This is displayed in Fig. 10a and 10b for cases (0) and (1), respectively. For clarity, the results are only plotted in the case of four processes, but similar trends can be observed in the cases when a different number of cores was used. Here, the total time of a timestep is monitored using FreeFem routines, which are although slightly less reliable than PETSc timers, provide more flexibility. The direct solution of the implicit system is omitted for convenience, as it is significantly higher than all the other methods. Examining case (0), the trends are quite obvious: for the whole computation, iterative solvers outperform direct solvers; explicit discretization of the nonlinear terms provides lower computational time than the implicit treatment; bJ and bGS preconditioners perform roughly the same in the implicit case; the efficient matrix-free implementations beat their counterparts that utilize matrix-forming. However, in case (1), although in the first part of the computations, these trends are the same, in the region of very high nonlinearity, different trends can be observed. In the interval \((\text{Re}_\delta \approx [650, 750])\), where the primary mode amplitude is larger than 2%, and the explicit solution diverged, the trends are reversed. The bGS preconditioner provides the best performance, followed by the bJ methods, and the otherwise efficient matrix-free bJ implementation is the worst. The reason for this can be interpreted with the mode amplitudes displayed in Fig. 5b and the average number of GMRES iteration show in Fig. 9b. In the region of very high nonlinearity, the bGS provides a better approximation of the inverse than the bJ, and although the bGS
preconditioner is more expensive to evaluate, because of the reduction in the
iteration count, bGS proves to be better. Furthermore, as in the matrix-free
implementation, multiplication with the LHS is more expensive than with the
matrix forming approach. Therefore, because of the high iteration count, the
performance of this method deteriorated. However, this is valid only for the
short spatial interval, and the performance of the bGS is only slightly better
than that of the bJ. These observations suggest that the bGS preconditioner
can only be efficient in a very high degree of nonlinearity.

![Fig. 10](image)

**Fig. 10**: Time of the marching steps as a function of the streamwise distance
in the case of four processes. (a): case 0; (b): case 1.

Next, we briefly examine the number of PSE iterations, which is shown in
Fig. 11. This, similarly to the number of GMRES iterations, was the same for
all the runs with different numbers of processes. Unsurprisingly, the direct and
iterative methods require the same number of PSE iterations; the difference is
only between the explicit and implicit ones. Case (0) can be seen in Fig. 11a. In
the first portion of the computation, there is only a small difference between the
number of iterations. Conversely, large differences can be found between the
number of iterations at large amplitudes. It is likely that the implicit method
outperforms the explicit one in this high-amplitude region; however, in a larger
portion of the computation, the explicit method is more efficient as it is less
expansive to evaluate, and the number of PSE iterations is the same. Case (1)
is presented in Fig. 11b. Here, essentially the same conclusions can be drawn
as in case (0). There is one key difference, however, compared to case (0): when
very large disturbance amplitudes are present, the maximum number of PSE
iterations is reached. It was checked, and in these cases, the difference between
the wavenumbers in successive iterations was less than $10^{-5}$, which is less than
1%. Therefore, this practical bound, which was set to limit the computational
time, did not influence the results.
Reusing LU factorization for efficient solution of the PSE

Finally, the maximum memory consumption is examined. This is displayed in Fig. 12a and Fig. 12b for case (0) and (1), respectively, where for each case, the columns are for a different number of processes in ascending order. Interestingly, a monotonic increase is observed in memory consumption with the increase in the number of processes; however, as the increase in memory is not significant in the case of the proposed iterative methods, this behavior was not investigated further. As expected, all the different methods require roughly the same memory except for the direct implicit case. This is because only the diagonal blocks of the matrices need to be LU factored in these cases. The small differences between these cases are associated with the storage of auxiliary matrices, off-diagonal matrix blocks in the implicit case, etc. In case (1) using the fully implicit discretization, at a large number of processes, the computer ran out of RAM, and likely, the SWAP memory was used (data was copied to the HDD). This definitely has a negative effect on the performance of the solver and maybe the reason for the slightly worse scaling in case (1) than in (0) (see Figs. 6 and Fig. 7). Nevertheless, it is also apparent that regardless of this deteriorating effect, the proposed preconditioners vastly outperform the direct solver.

4.3.2 Scaling: mesh size and sparsity

In this section, only the solution time of the linear systems of equations is measured. The LUPC preconditioner is compared with the direct solver for different mesh sizes. This calculation was carried out only for case (0) with the lower mode amplitude and fewer modes. Additionally, this calculation was performed not only using $P_2 - P_1$, but also with $P_4 - P_2$ elements. The parameters of the different meshes used in this study are presented in Table 2. The
mesh sizes were identified to yield the prescribed degrees of freedom in column 1. Note that these values could only be approximately set, since the same mesh structure was preserved: tuning a single scaling factor allows only a coarse control of the number of triangles.

The results are shown in Fig. 13. The first and second rows display the data for the $P_2 - P_1$ and $P_4 - P_2$ elements, respectively. It is apparent that the direct and iterative solvers follow the same trend when varying the mesh size. However, the curves of the two solvers slightly diverge: the iterative solver scales better than the direct method. More precisely, the ratio of the direct and iterative solution varies roughly from 4.5 to 7.5 in the explicit and 20 to 60 in the implicit case using $P_2 - P_1$ elements. For the $P_4 - P_2$ elements, these value pairs are 5 – 7 and 15 – 45, respectively. Although this shows that a larger speedup can be achieved using more sparse matrices, the difference between the speedups is not substantial, and for higher order elements also an outstanding computational speedup is achieved. For the sake of completeness the matrix-free implementation is also included, which shows similar scaling
Table 2: The parameters of the meshes in the scaling study. nVert., nTri., and nDOF states for the number of vertices, triangles and degrees of freedom associated with the mesh, respectively.

<table>
<thead>
<tr>
<th>Approx. nDOF</th>
<th>$P_2 - P_1$</th>
<th></th>
<th>$P_4 - P_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nVert.</td>
<td>nTri.</td>
<td>nDOF</td>
</tr>
<tr>
<td>$10^4$</td>
<td>823</td>
<td>1528</td>
<td>10093</td>
</tr>
<tr>
<td>$1.5 \cdot 10^4$</td>
<td>1230</td>
<td>2320</td>
<td>15276</td>
</tr>
<tr>
<td>$2 \cdot 10^4$</td>
<td>1620</td>
<td>3082</td>
<td>20257</td>
</tr>
<tr>
<td>$3 \cdot 10^4$</td>
<td>2430</td>
<td>4663</td>
<td>30593</td>
</tr>
<tr>
<td>$4 \cdot 10^4$</td>
<td>3187</td>
<td>6156</td>
<td>40336</td>
</tr>
<tr>
<td>$6 \cdot 10^4$</td>
<td>4692</td>
<td>9125</td>
<td>59694</td>
</tr>
<tr>
<td>$8 \cdot 10^4$</td>
<td>6315</td>
<td>12320</td>
<td>80542</td>
</tr>
<tr>
<td>$1.2 \cdot 10^5$</td>
<td>9276</td>
<td>18184</td>
<td>118749</td>
</tr>
</tbody>
</table>

to the iterative method with matrix-forming. These findings imply that the proposed preconditioner should generalize well to compressible flow, where the structure of the matrices is more dense. As a final remark, we note that in the case of $P_4 - P_2$ and the smallest mesh, the explicit NLPSE solution diverged, and no results could be obtained except for the matrix-free implementation (bottom-left figure). No efforts were made to investigate this behavior; however, it is hypothesised this is related to the numerical instabilities mentioned related to high order finite element mentioned in Section 3.

4.4 Generalization: test on the Boundary Region Equations

Next, the generalization of the preconditioner method is tested on the BRE. The same streamwise domain is used in the solution as for the PSE, and the initial solution is marched downstream in 200 non-equidistant steps that follow the boundary layer growth. In this investigation, only the time of the linear equation solution is monitored. Since the solution of the linear equations is handled by the nonlinear equation solver SNES routine in PETSc, and the present FreeFem-PETSc interface also limits access to the internal function calls, the linear equation solution time could be only approximated. It was estimated as the time of the nonlinear equation solver minus the time spent forming the Jacobian and the error calculation - these are the two other more expensive operations. Note that this estimate overpredicts the linear equation solution time, as it contains additional operations (e.g., vector norming and solution update in the calculated direction). The results are shown in Fig. 14. The blue and red lines denote the LU factorization and the preconditioned iterative method, and the left and right figures displaying the $P_2 - P_1$ and $P_4 - P_2$ results, respectively. An approximately linear relationship can be found in the case of both discretization and solution time of the LU factorization is roughly twice the solution time of the iterative method. Note that here,
Fig. 13: Time of the linear equation solution as a function of the number of unknowns in the case of the NLPSE, case (0). The first row displays the $P_2 - P_1$, the second row the $P_4 - P_2$ discretization. The first and second columns display the explicit and implicit discretization, respectively.

real arithmetic is used, which is considerably cheaper than operations with the complex numbers in the case of the PSE, which infers that the method generalizes to computationally less cumbersome tasks, e.g., smaller problems. All the above findings suggest the generality of the LUPC method.

5 Summary and concluding remarks

In this paper, it was investigated whether the plane marching PSE solution can be accelerated by a simple iterative technique. In general, in each marching step, the linear systems of equations arising from the discretization of the governing equations are solved repeatedly with LU factorization. As an alternative, the LU factorization was only calculated a single time, and it was used as a preconditioner for an iterative solver for the following linear equation solutions. The technique’s effectiveness is argued to stem from the slow variation of the problem parameters in the marching direction. The proposed method is tested with numerical experiments of the incompressible PSE formulation that were conducted using the parallel implementation of the FreeFem-PETSc
Reusing LU factorization for efficient solution of the PSE

In the case of the LPSE, it was shown that the preconditioned iterative solver takes roughly one-tenth the time of the LU factorization, once the factorization is available. Moreover, it was also shown that the iterative method strongly outperforms off-the-shelf preconditioners, and it was argued that is also outperforms state-of-the-art preconditioners. In the NLPSE, both explicit and implicit treatment of the nonlinear terms was investigated for two cases with different numbers of modes and amplitudes, and therefore, different level of nonlinearity. It was found that a factor of 5-7 and 30-95 acceleration was achieved in the explicit case and implicit cases, respectively. In the implicit case, the bJ and bGS preconditioners yielded the same solution time, and the results suggest that for the low-moderate level of nonlinearities, the bJ, while for high levels of nonlinearity, the bGS method is preferable. Although the computational cost of the implicit formulation remained higher than of the explicit one, the very fact that the computational cost of the two techniques is comparable can be considered an outstanding achievement. This truly renders the suggestion of Herbert [5] - that the implicit formulation can be favorable - relevant. Moreover, it was shown that the method works not only for $P_2 - P_1$ but also for $P_4 - P_2$ finite element spaces, suggesting that the method generalizes for high order finite difference discretization, which is a popular choice for PSE implementations. Finally, using the fact that the key underlying assumption for the preconditioning technique was the slow variation in the marching direction, it was shown that the solution of the BRE could also be accelerated using the proposed method.

The present study focused on accelerating the solution of the linear systems of equations. Therefore, in most of the numerical experiments, the matrix-forming was done by using convenient but ineffective FreeFem routines. As a
remedy, constrained by the present limitation of the otherwise very efficient and amazing) FreeFem-PETSc interface, a simple matrix-free approach, was formulated, which could significantly accelerate the runtime. However, it is very likely that with access to the complete PETSc interface, a much more efficient implementation of the matrix forming may be possible. The above results indicate that combining the plane marching PSE with the finite element discretization can be a very efficient choice, maybe even superior to finite difference discretization. However, a thorough comparison of the two methods in a detailed benchmark study of the PSE (and stability equations, in general), should consider the following aspects:

- **Spatial discretization**: in semi-infinite domains such as boundary layer flows, a large far field is required for the solution to satisfy the boundary conditions to usually prescribed at infinity. Although there are different mappings for finite difference/spectral collocation methods which allow the points to be clustered in the near field, the spanwise discretization cannot be controlled in the far field. However, the finite element method allows a very flexible mesh control (see Fig. 1a).

- **Sparsity**: question naturally arises, whether low-order discretization schemes are preferable that result in a larger number of unknowns but more sparse matrices or higher order schemes with smaller but more systems should be favored. This was investigated by Paredes et al. [64] presented a very precise, detailed study examining this aspect in the context of flow stability problems. They were interested in the comparison of finite difference and spectral methods, and they did not reduce the discretization order below eight. However, both the finding of the present study, and the fact that 4-6th order finite difference method is the most common in PSE studies, motivate the detailed assessment of low order discretization methods.

- **Solver**: both direct and iterative solvers should be considered in the benchmark studies.

- **LHS and RHS assembly**: The present study demonstrates how suboptimal matrix assembly can increase the computational time. The LHS and RHS assembly can be even more cumbersome in the compressible formulation of NLPSE, especially if the additional terms arising from the curvilinear coordinate system are considered [32].

Furthermore, the origin of the numerical instabilities associated with the spanwise periodic boundary condition reported in Sec. 3 should also be investigated.

Despite the above outlined several aspects and questions that may improve the performance of the proposed preconditioner method, it can be considered very effective and successful. Therefore, due to the generality of the basic assumptions that guarantee the success of the method, although an incompressible flow was considered, the results indicate the reused LU factorization preconditioning could be extended to compressible flows. The method should
also generalize to the different modified versions of the PSE method. The stabilization of the explicit NLPSE by implicitly discretizing the MFD terms \([4, 30]\) is also possible within this formalism. The preconditioner method should be very efficient in the explicit NLPSE when the equations are stabilized by relaxation of the updated nonlinear terms, since in this case, the convergence is quite slow, and a single LU factorization can be reused several times as a preconditioner. The effectiveness of the preconditioner method is questionable to the quasi-3D PSE formalism, where the number of unknowns is much smaller than in the plane marching PSE. As the quasi-3D PSE formalism is very popular, and the findings of the present study are encouraging, the combination of the two should be studied.

The findings of the present study strongly question the paradigm in the boundary layer stability community that LU factorization is the most efficient method to solve moderate-size sparse linear systems of equations. This was highlighted by the fact the solution of the BRE could also be accelerated using the LUPC technique. We hypothesize that based on (i) the structure of the model equations and (ii) physical assumptions, efficient iterative methods can be constructed that are superior to direct solvers. In the PSE formalism, when chemical reactions are included among the modeling equations, it is possible that efficient preconditioners can be constructed based on the block structure of the equations. Another problem for which economic preconditioners may be constructed is the one-way Navier Stokes (OWNS) equations which were proposed by Towne and coworkers \([17–19]\). The OWNS solves two major shortcomings of the PSE formalism, as they are both well-posed and include the interaction between different instability modes. OWNS is constructed by augmenting the system with additional variables. The hierarchical structure of the additional variables suggests that a lower/upper triangular preconditioner may be efficient for the original OWNS. Furthermore, possibly, a hybrid upper-lower triangular preconditioner could be applied to the more recent projection-based OWNS formulations (see eq. C03 in \([18]\)). As mentioned in the introduction, simple block preconditioners have already been proven effective for harmonic balance-type flow stability problems \([69, 70]\). The combination of these two methods is likely to be efficient in boundary layer stability theory, and applicable to e.g. parabolized Floquet equations \([91]\) or the adaptive linearized Navier-Stokes equations \([92, 93]\). Moreover, the efficiency of block preconditioners may be improved with physics-based arguments. Moulin et al. \([68]\) used the augmented Lagrangian preconditioner for inversion of the linearized Navier-Stokes (LNS) equations. In their formulation, they invert the lower-triangular part of the LNS operator since this is the available option in PETSc, the library used in their calculations. However, in many convection-dominated flows, such as boundary layers, the streamwise velocity component is dominant. This implies that the preconditioner might be more efficient if it approximates the inverse of the upper-triangular part of the LNS equations, as it carries more weight than the lower one. If the above assumption holds, it could also be applicable to compressible flows, for which iterative solvers
received much less attention. Finally, in a more broad context, the block structure of the equations arising from different types of flow stability problems, e.g., energy stability calculation using the Reynolds-Orr equation [9] (see also Eq. (14) in [94]) also implies that efficient preconditioning techniques may be developed for three-dimensional flow problems.

As a concluding remark, we briefly return to the fact that the matrix assembly took a substantial amount of computational time. A significant factor of 2 speedup could be achieved in the total computational time by a small amount of code optimization, as the slowness was because of the use of convenient but inefficient program utilities. This highlights the seldom-mentioned but paramount importance of efficient coding. For example, the proposed preconditioning technique might not be applicable in Matlab using standard functionalities since the default sparse direct solver is the UMFPACK package [95] written in C, while the built-in GMRES algorithm is written in Matlab. Efficient coding itself can be a source of performance optimization. In the context of low-order ordinary differential equations, when a large number of equations is solved in parallel (in the case of parameter studies), it was pointed out the order-of-magnitude speedup could be achieved by identifying bottlenecks in the computation [96, 97]. Finally, GPUs, which are becoming more and more widespread in computational modeling, their utilization could be a promising future direction to accelerate boundary layer stability calculations as the open-source libraries developed by the scientific community continue to evolve.

Acknowledgments. The authors would like to thanks Pierre Jolivet for suggesting reusing LU factorization as a preconditioner, and his help with the parallel capabilities of FreeFem. The fruitful discussions with Péter Tamás Nagy are gratefully acknowledged.

Declarations

- **Ethics approval**: not applicable.
- **Conflict of interest**: The authors declare that they have no conflict of interest.
- **Authors’ contributions**: A. Sz. developed the methodology and software, prepared the figures, wrote the initial draft, and also took part in the revision of the manuscript. P. GY. was part of the conceptualization and supervision of the work, took part in the revision of the manuscript, and was responsible for the funding acquisition.
- **Funding**: This research was performed within the framework of NKFI Hungary 142675 project and grant No. CELSA/21/024. András Szabó was supported by the UNKP-22-3-II-BME-112 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.
- **Availability of data and materials**: not applicable.
References


Reusing LU factorization for efficient solution of the PSE

S0045-7930(96)00044-8


Reusing LU factorization for efficient solution of the PSE


