A direct numerical simulation code for incompressible turbulent boundary layers

Technical Note: **ETSIA /MF-071**

Mark P. Simens
Javier Jiménez
Sergio Hoyas
Gwenäel Häuet
Samuel Vaux

Technical report FROM:
Computational Fluid Dynamics
School of Aeronautics
Universidad Politécnica Madrid.

February 2007

Summary

A code for the direct numerical simulation of spatially developing turbulent boundary layers has been implemented by the Computational Fluid Dynamics group at the School of Aeronautics in Madrid. The code uses a fractional-step scheme to solve the incompressible three-dimensional Navier–Stokes equations. It is formally second-order accurate, using two-point centred staggered finite differences for the pressure correction substep. The advective derivatives are however discretized using fourth-order compact finite differences in two directions, and Fourier expansions spanwise. Temporal advancement is by a three-step Runge-Kutta. These choices are discussed in the report.

Boundary conditions are fairly general, allowing the simulation of arbitrary imposed pressure gradients, as well as control manipulations at the wall and on the free stream. Turbulent inflow conditions can be generated by recycling a properly-scaled interior plane. The code has been parallelized (MPI) for distributed-memory machines, and has been tested to scale well for large problems, in up to 300 processors.

Here we briefly describe the code implementation and performance, and report on two sets of test cases. One is a zero-pressure-gradient turbulent boundary layer, at $Re_\theta = 600 - 900$. The other is a set of three laminar separation bubbles, induced by a strong adverse pressure gradient, that transition within the separated region, and, as a consequence, reattach. The turbulent attached regions of both test cases are compared with previous simulations, and with experiments. The results are generally satisfactory.

Contents

1 Introduction

Direct numerical simulations of incompressible turbulent boundary layers are more complicated, and more expensive, than those of simple internal flows. This is in part because the boundary conditions are more complex, but also because the computational domain usually has to be longer, to allow enough distance for flow development. Probably because of this, the range of formulations and discretization schemes used for external flows has been wider than for internal ones, where, for example, most large turbulent channels have either been simulated using the formulation in [16] or that in [26].

The inherent non-periodicity of boundary layers in the streamwise direction makes the use of periodic boundary conditions and of spectral methods somewhat artificial, although work-arounds have been implemented in [42, 2, 38]. It is generally preferable to use numerical schemes that allow for the streamwise development of the layer [18, 30, 24, 44, 22].

Various formulations of the Navier-Stokes equation have been used. For example, a vorticity-velocity formulation was chosen in [22, 30]. This avoids, or at least hides, the issues related to pressure and mass conservation, but requires at least 50% more memory than other schemes. The code described here uses a primitive-variables formulation, in which a fractional-step method is used to solve for the pressure and to assure mass conservation. That method was introduced in [6], and was first used for turbulent flow simulations in [15]. Although it is still an area of investigation and of some controversy [8], most of the problems surrounding it [15] have now been solved [28, 9, 26, 3], allowing it to maintain second-order temporal accuracy for the velocity components.

Our code uses second-order staggered centred finite differences for the pressure correction sub-step, and fourth-order-accurate staggered compact finite differences [25] for the advective terms. A large number of boundary-layer simulations has been done using finite-differences schemes [18, 30, 24, 44, 22]. Implementations using second-order central finite difference for all the derivatives were used in [44, 24, 18]. Those schemes can be made to conserve momentum, energy and mass, but their relatively low resolution requires a lot of grid points to achieve a given accuracy. High-order schemes, such as those used in [30, 22] require smaller grids, and ultimately save time. Our choice of schemes with mixed consistency orders is discussed in section §3.2. A critical comparison between high-order finite element methods, B-splines, and compact finite differences can be found in [17]. All of them have their own advantages, including conservation properties, and different degrees of programming complexity. Our choice of fourth-order compact finite differences on a staggered grid [13] does not conserve energy, but is known to be robust if most of the energy-containing scales are resolved [25]. In fact, energy conservation is only relevant if the resolution requirements for direct turbulence simulation are not met. Any numerical scheme that is consistent, and formally stable, is stable in a well-resolved turbulence simulation in which the flow is smooth at the scale of the grid. That implies resolving all the scales larger than approximately the viscous Kolmogorov limit, which is also the requirement for retaining reliable physics.

The organization of this report is as follows. The formulation of the Navier–Stokes problem is presented in section 2, and the spatial and temporal discretization are discussed in section 3.

Since we are interested in simulating separated flows, which are very inhomogeneous streamwise, we retain the ability to use nonuniform grids in the streamwise direction, as well as perpendicular to the wall. This, together with the desire to allow for general boundary conditions that do not automatically guarantee global mass balance, leads us to re-examine the issue of global mass conservation. This important question, together with the treatment of the pressure, is briefly discussed in §3.3.

Section 4 contains the results of testing the code on several moderate-Reynolds-number boundary layers, both with and without pressure gradients. Conclusions are summarized in section 5. An appendix deals with certain numerical details used in the text.

2 Problem Formulation

The equations solved are the incompressible Navier-Stokes equations,

$$\begin{aligned} \nabla\cdot\vec{u} &= 0, \\ \partial_t\vec{u}+\nabla\vec{u}\cdot\vec{u} &= -\nabla p+\frac{1}{\mathrm{Re}}\nabla^2\vec{u}. \end{aligned} \tag{1}$$

The streamwise, spanwise and wall-normal directions and velocities are given respectively by x, z, y, and by u, w, v. The boundary conditions may vary, as the code should be capable of simulating various different spatially developing flows with only minor changes.

A fractional step method [9, 28] is used to ensure mass conservation, and to efficiently solve for the pressure. The main idea is to write (1) in semi-discrete form,

$$\begin{aligned} D\vec{u}^{n+1} &= 0, \\ \vec{u}^{n+1}-\Delta t\, L\vec{u}^{n+1} &= \Delta t\left(-Gp^{n+1}-N\vec{u}^{n}\right). \end{aligned} \tag{2}$$

The matrices D, G and L contain the finite-differences approximations to the divergence, gradient, and linear implicit terms respectively. The operator N contains the nonlinear advective terms, and whatever part of the viscous terms that is computed explicitly. This system can be written in matrix form as [28],

$$\begin{bmatrix} A & \Delta tG \\ D & 0 \end{bmatrix}\begin{pmatrix} \vec{u}^{n+1} \\ p^{n+1} \end{pmatrix}=\begin{pmatrix} r^n \\ 0 \end{pmatrix}+\begin{pmatrix} b.c.'s \\ b.c.'s \end{pmatrix}, \tag{3}$$

where $A=I-\Delta tL$, and $r^n=-\Delta t\, N\vec{u}^n$ contains the explicit terms. Note that the boundary conditions are imposed at this point, as indicated by the $b.c.'s$.

It was pointed out in [28] that an LU decomposition can be used to factorize the system in (3). Before doing that, for reasons that are explained later, the equations are rewritten as

$$\begin{bmatrix} A & \Delta tG \\ D & 0 \end{bmatrix}\begin{pmatrix} \vec{u}^{n+1} \\ \Delta p^{n+1} \end{pmatrix}=\begin{pmatrix} r^n_m \\ 0 \end{pmatrix}+\begin{pmatrix} b.c.'s \\ b.c.'s \end{pmatrix}, \tag{4}$$

where $r_m^n = r^n - \Delta t G p^n$, and $\Delta p^{n+1} = p^{n+1} - p^n$. The system is then factorized as

$$\left[\begin{array}{cc} A & 0 \\ D & -\Delta t D A^{-1} G \end{array}\right]\left(\begin{array}{c} \vec{u}^* \\ \Delta p^{n+1} \end{array}\right) = \left(\begin{array}{c} r_m^n \\ 0 \end{array}\right) + \left(\begin{array}{c} b.c.'s \\ b.c.'s \end{array}\right), \tag{5}$$

$$\left[\begin{array}{cc} I & \Delta t A^{-1} G \\ 0 & I \end{array}\right]\left(\begin{array}{c} \vec{u}^{n+1} \\ \Delta p^{n+1} \end{array}\right) = \left(\begin{array}{c} \vec{u}^* \\ \Delta p^{n+1} \end{array}\right). \tag{6}$$

This is exact, and could be solved without further modification, but it is only computationally efficient if the time integration is fully explicit, in which case $A = I$. When implicit time integration is used, A^{-1} is a full matrix, and obtaining the solution becomes expensive. This can be avoided by substituting the full inverse of A by its lowest order approximation, $A^{-1} = I$, where the neglected terms are $O(\Delta t)$. This results in,

$$\left[\begin{array}{cc} A & 0 \\ D & -\Delta t D G \end{array}\right]\left(\begin{array}{c} \vec{u}^* \\ \Delta p^{n+1} \end{array}\right) = \left(\begin{array}{c} r_m^n \\ 0 \end{array}\right) + \left(\begin{array}{c} b.c.'s \\ b.c.'s \end{array}\right), \tag{7}$$

$$\left[\begin{array}{cc} I & \Delta t G \\ 0 & I \end{array}\right]\left(\begin{array}{c} \vec{u}^{n+1} \\ \Delta p^{n+1} \end{array}\right) = \left(\begin{array}{c} \vec{u}^* \\ \Delta p^{n+1} \end{array}\right). \tag{8}$$

The reason for using Δp^{n+1} in (4), instead of p^{n+1}, can now be discussed. The error made by neglecting the viscous contribution to A^{-1} is $\Delta t \, L \Delta p^{n+1}$, which is of order $O(\Delta t^2)$ because $\Delta p^{n+1} \sim O(\Delta t)$. If p^{n+1} had been used instead, as in (3), the resulting scheme would have been only first-order accurate. The order of the scheme is thus increased by one by using (4).

The advantage of this matrix interpretation of the fractional-step method is that boundary conditions are imposed before the LU decomposition is applied to (4). In other words, matrices like A, D, etc. do not include the discrete boundary equations anymore. The pressure correction Δp^{n+1} is solved from the discretized equation $-\Delta t D G$, and it is impossible, and unnecessary, to impose boundary conditions on the pressure if velocity boundary conditions have already been used when computing the divergence in (3). In chronological order the following is done

$$\begin{aligned} A\vec{u}^* &= r_m^n + b.c.'s, & (9)\\ \Delta t D G \Delta p^{n+1} &= D\vec{u}^* - b.c.'s, & (10)\\ \vec{u}^{n+1} &= \vec{u}^* - \Delta t G \Delta p^{n+1}, & (11)\\ p^{n+1} &= \Delta p^{n+1} + p^n. & (12) \end{aligned}$$

Up to this point the equations are given as if Euler time integration was used. Instead a Runge-Kutta scheme is applied. A popular time-stepping scheme for fractional-step methods is

$$\begin{aligned} (I - \beta_l \Delta t \, L)\, \vec{u}_*^{l+1} &= -(\alpha_l + \beta_l)\, G p^l - \gamma_l N u^l - \zeta_l N u^{l-1} + \alpha_l \Delta t \, L u^l + b.c.'s, & (13)\\ (\alpha_l + \beta_l)\, \Delta t D G \Delta p^{l+1} &= D\vec{u}_*^{l+1} - b.c.'s, & (14)\\ \vec{u}^{l+1} &= \vec{u}_*^{l+1} - (\alpha_l + \beta_l)\, \Delta t G \Delta p^{l+1}, & (15)\\ p^{l+1} &= \Delta p^{l+1} + p^l, & (16) \end{aligned}$$

where $l = n + k/3$, and $k = [0, 3]$. The coefficients are given in [41] as,

$$\begin{array}{lll} \zeta_1 = 8/15, & \zeta_2 = 5/12, & \zeta_3 = 3/4, \\ \gamma_1 = 0, & \gamma_2 = -17/60, & \gamma_3 = -5/12, \\ \alpha_1 = 4/15, & \alpha_2 = 1/15, & \alpha_3 = 1/6, \\ \beta_1 = 4/15, & \beta_2 = 1/15, & \beta_3 = 1/6. \end{array} \tag{17}$$

It remains to be determined whether all the viscous terms need to be integrated implicitly, which complicates the computation, or whether some of them could be treated explicitly. Previous studies suggest that the latter might be possible [1].

Historically, the viscous terms in all three directions have been treated implicitly because an explicit stable simulation requires that

$$\Delta t < \min\left(\frac{\alpha\Delta x^2}{\nu}, \frac{\alpha\Delta z^2}{\nu}, \frac{\alpha\Delta y^2}{\nu}; \frac{\beta\Delta x}{\max|u|}, \frac{\beta\Delta z}{\max|w|}, \frac{\beta\Delta y}{\max|v|}\right), \tag{18}$$

where α and β are constants that depend on the largest eigenvalues of the first- and second-derivative operators, and $\Delta x, \Delta y, \Delta z$ are the grid sizes. In boundary-layer flows, the advective limit is frequently set by Δx and by the free-stream velocity U_∞, while the transverse velocities are lower. One can then approximate (18) by,

$$\Delta t < \min\left(\frac{\beta\Delta x}{U_\infty}; \frac{\alpha\Delta z^2}{\nu}, \frac{\alpha\Delta y^2}{\nu}, \frac{\alpha\Delta x^2}{\nu}\right), \tag{19}$$

because the transverse convective time-step restrictions are not important. Note that this is not always the case, specially in separated and in free-shear flows, where the transverse velocities may be of order U_∞. This however does not invalidate the following analysis, since it just tightens the convective time limitation.

The factors Δx^2, Δy^2, and Δz^2 of the viscous limit seem to lead to a severe time-step restriction. To see whether that is the case, we now compare the convective and viscous time step limitations. For the convective limit to dominate, it is sufficient that

$$\frac{\beta\Delta x}{U_\infty} \leq \frac{\alpha}{\nu}\min\left(\Delta x^2, \Delta y^2, \Delta z^2\right). \tag{20}$$

If this inequality is satisfied, the viscous terms can be treated explicitly. After multiplying both sides by $u_\tau^2/(\nu\beta)$, the inequality becomes

$$\frac{u_\tau}{U_\infty}\Delta x^+ \leq \frac{\alpha}{\beta}\min\left({\Delta x^+}^2, {\Delta y^+}^2, {\Delta z^+}^2\right), \tag{21}$$

where $\Delta x^+ = u_\tau \Delta x/\nu$ is expressed in 'wall units', defined in terms of the shear stress at the wall, u_τ^2, and of the kinematic viscosity ν. This adimensionalization is often the right one for wall-bounded turbulent flows, and grids for those cases are usually chosen of the order of [16],

$$\Delta x^+ \approx 8, \quad \Delta z^+ \approx 4, \quad \Delta y^+ \sim 0.3. \tag{22}$$

The relations $u_\tau/U_\infty \approx 1/20 \ll 1$, and $\alpha/\beta \sim 1$, indicate that the viscous time-step limit is only severe in the y-direction. That is therefore the only direction treated implicitly in our code, while the viscous terms in the x and z-direction are done explicitly.

3 The spatial discretization

The sketch in figure 1 gives detailed information about the positions of the different variables in the numerical domain. The derivatives in the x and y-directions are calculated using compact

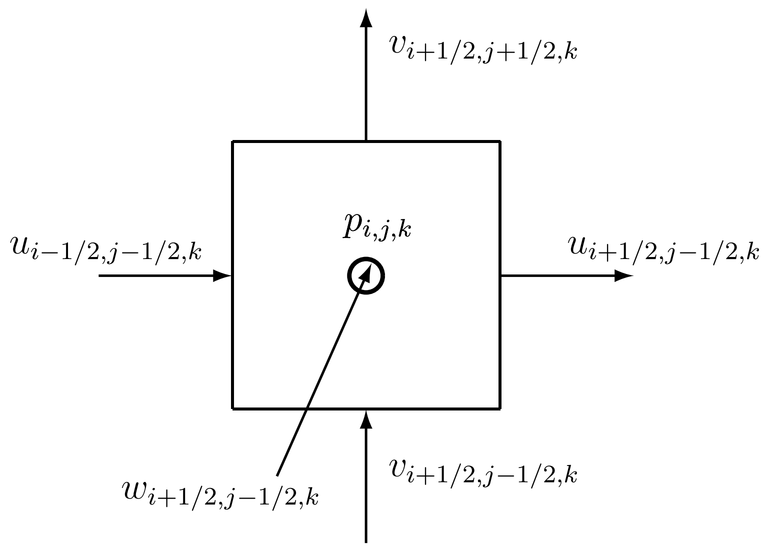


Figure 1: Location of the flow variables in the computational grid. Note that the grid is not staggered in the z direction, so that all the variables are in the same Fourier mode or collocation point.

finite-difference schemes of fourth-order accuracy, while the derivatives in the z direction are calculated in Fourier space. The x and y derivatives are obtained for non-uniform grids, and can be written in a general form as,

$$\alpha_j^I \bar{f}_{j-1} + \bar{f}_j + \beta_j^I \bar{f}_{j+1} = a_j^I f_{j-1/2} + b_j^I f_{j+1/2}, \tag{23}$$

$$\alpha_j^D f'_{j-1/2} + f'_{j+1/2} + \beta_j^D f'_{j+3/2} = a_j^D \bar{f}_j + b_j^D \bar{f}_{j+1}, \tag{24}$$

$$\alpha_j^V f''_{j-1/2} + f''_{j+1/2} + \beta_j^V f''_{j+3/2} = a_j^V f_{j-1/2} + c_j^V f_{j+1/2} + b_j^V f_{j+3/2}, \tag{25}$$

where f' and f'' are the first and second derivatives of f, and $\bar{f}$ is its interpolated value at some intermediate point. For uniform grids, the coefficients in (23–25) are given in reference [25]. The coefficients for nonuniform grids are obtained using Taylor expansions to ensure maximum order of accuracy, and can be found in [31]. It is important to note that, on a staggered non-uniform grid, the coefficients are not the same for u and v. On the other hand, the coefficients for the first derivative and for the interpolation in the x-direction of w are equal to the ones for v. Similarly, the coefficients for the first derivative and for the interpolation of u and w in the y direction are the same. So are the coefficients of the second derivatives of v and w in the y direction, and those of u and w along x.

3.1 Fourth-order schemes and the Poisson matrix

Various ways of constructing the Poisson matrix DG were tested during the development of the code. The most obvious choice is to use the same compact finite-differences schemes as for the advective derivatives. In that case the divergence is calculated using

$$A_D \vec{u}' = B_D \vec{u}, \tag{26}$$

and the pressure gradient is calculated using

$$A_G p' = B_G p, \tag{27}$$

where A_G, A_D, B_D, B_G contain the compact finite difference operators. The discretized Poisson equation then becomes

$$A_D^{-1} B_D A_G^{-1} B_G p = A_D^{-1} B_D \vec{u} \quad \Rightarrow \quad B_D A_G^{-1} B_G p = B_D \vec{u}. \tag{28}$$

The matrix $B_D A_G^{-1} B_G$ is full, and discretizing DG in this manner is prohibitively expensive. Note that this problem has similitudes with the fractional step method described in (4). The only efficient solution is to require that A_G^{-1} should not be a full matrix.

This led to the next solution to be tried, which was to combine a compact finite difference scheme for the divergence, and an explicit one for the gradient. The divergence was calculated as in (26), while for the gradient we let $A_G = I$. This was implemented using a multigrid solver that is described in detail in [31]. Although the solver worked reasonably well, the results were unsatisfactory, and the code had a tendency to blow-up at the boundaries during start-up. This was probably due to the lack of energy conservation, which cannot be guaranteed once the differentiation schemes for the nonlinear terms and for the pressure are different. However, since it will be seen below that the Poisson substep is not the limiting one for the accuracy of the code, this option was discarded without further investigation.

The next solution was therefore to let $A_G = A_D = I$, so that both differentiation schemes were explicit and comparable. Within this choice, both second- or fourth-order stencils for the derivatives were considered.

Both options have disadvantages when compared with compact finite differences. In the case of fourth-order explicit derivatives, resolution is lost, besides paying the fairly high price of wider stencils. In second-order schemes, both consistency and resolution are lost. In our code, the decision was to use second-order stencils for the pressure, essentially because it turns out that, for reasonable resolutions, the dominant error comes from the first derivatives of the advective terms. This is discussed in detail in section §3.2.

The divergence of the velocity is thus calculated as

$$D\widehat{\vec{u}}_{i,j,k_z} = \frac{\widehat{u}_{i+1/2,j,k_z} - \widehat{u}_{i-1/2,j,k_z}}{\Delta x} + \frac{\widehat{v}_{i,j+1/2,k_z} - \widehat{v}_{i,j-1/2,k_z}}{\Delta y} + ik_z \widehat{w}_{i,j,k_z}, \tag{29}$$

where k_z is the spanwise wavenumber and the carat is used to indicate that the variables are in spectral space along the z-direction. The pressure gradient is

$$G_x \widehat{p}_{i-1/2,j,k_z} = \frac{\widehat{p}_{i,j,k_z} - \widehat{p}_{i-1,j,k_z}}{\Delta x}, \text{ etc., and} \quad G_z \widehat{p}_{i,j,k_z} = \mathrm{i} k_z \widehat{p}_{i,j,k_z}. \tag{30}$$

The resulting Poisson operator is tridiagonal in the first two directions, and is expensive to solve directly. Two solution schemes have been implemented.

The first one is a direct method, consisting in expanding $D\widehat{\vec{u}}^*$ and $\widehat{p}$ as cosine series in the x direction [15]. The variables are already in spectral space along z, and the matrix reduces to a set of systems of the type

$$(\partial_{yy} - K^2)\widehat{p} = D\widehat{\vec{u}}^*, \tag{31}$$

where K is some modified wavenumber. Those systems can easily be treated by a tridiagonal matrix solver. An important disadvantage of this method is its inability to treat correctly non-periodic schemes higher than second order. The reason is basically that the spectral cosine expansion of the divergence assumes that the derivatives at both ends vanish, and, if that is not true, it incurs in second-order errors. Another way of expressing the same idea is that high-order schemes in non-periodic domains need boundary stencils. Those have in general a different modified wavenumber than the interior scheme, and no unique modified wavenumber can be used in (31). A second disadvantage is that only grids that are uniform along x can be accommodated by the Fourier expansion.

The second method is a multigrid solver that can be used for arbitrary Cartesian grids, and that could, in principle, be adapted to derivatives of any order. It has only been implemented for the second-order pair (29–30). There is no previous Fourier expansion of the divergence, except along z, and the numerical equation to be solved is

$$(\partial_{xx} + \partial_{yy} - k_z^2)\widehat{p} = D\widehat{\vec{u}}^{\,*}. \tag{32}$$

The solver was designed using a V-cycle, applying line-Gauss iteration in the y direction. Around ten V-cycles are needed to converge to round-off precision. In every cycle the full matrix is iterated twice; once for the down-stroke and a second time for the upstroke. The multigrid was developed at an early stage of the code development process, and it proved to be slower than the Fourier decomposition method.

A crude operation count suggests that this was not solely the effect of poor coding, although it is true that the publicly available Fourier transforms used for comparison are usually very well written. It was found that the publicly available multigrid codes did not fit our needs. Our multigrid was coded from scratch, and was abandoned fairly early for the cosine method. It is conceivable that it could be recoded to be more efficient, and it may prove necessary to do so in the future, since it remains the only solver able to cope with non-uniform grids in the x direction. All the results presented below were obtained with a second-order Poisson matrix, using cosine expansions.

3.2 Discretization errors

The errors of finite-differences approximations have to be discussed both in terms of consistency, i.e, their behaviour when $\widetilde{k} = kh \ll 1$, and of resolution, when $\widetilde{k} = O(1)$. Here we use k for a generic dimensional wavenumber, and h for the grid spacing. We have seen above that our code uses fourth-order finite differences for the advective terms, but only second-order ones for the pressure correction. This is difficult to justify at the consistency level, but becomes reasonable when the resolution limit is considered. The argument is similar to the one used in [41] to justify the use of a three-step Runge–Kutta, even if the formal accuracy is limited to second order by the fractional-step splitting. The magnitude of the errors for the advection schemes increases faster in the resolution limit than those for the pressure correction step, and balancing the errors for $\widetilde{k} \approx O(\pi/2)$ requires a higher-order formula for the former than for the later.

This is clear in figure 2, which shows the relative errors of both second-order FD and fourth-order CFD schemes as functions of the spatial wavenumber $\widetilde{k}$, compared with those of the

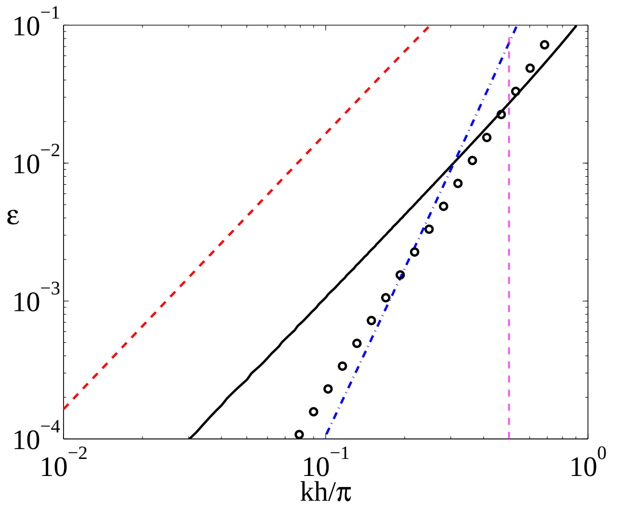


Figure 2: Relative error due to the spatial discretization, versus the numerical wavenumber. ---- , second-order first derivative; —·— , fourth-order compact first derivative; ——— , second-order pressure projection. For details, see appendix A. The circles are the temporal discretization error of a three-step Runge-Kutta adjusted to $CFL = \sqrt{3}$ at the grid spacing. The vertical dashed line is $\widetilde{k} = \pi/2$.

second-order pressure correction. The details of the error estimation can be found in appendix A. If we for example take $\widetilde{k} = \pi/2$ as the design point for the code, and require that the errors of both substeps should be comparable, the figure implies that the order of the operators for the advection terms should be higher than those used for the pressure correction. Choosing such a design point implies that the grid spacing along the directions using finite differences should be 2 or 3 times finer than the one along spectral directions, since the latter are essentially accurate up to $\widetilde{k} = \pi$. The expected discretization errors would then be of the order of a few percent. Using a higher-order scheme for the pressure substep would of course make it more accurate, but the overall performance of the code would be limited by the errors in the advective step, unless a much finer grid was chosen for it.

Figure 2 also includes the relative temporal error to be expected from a three-step Runga-Kutta for which the time step has been adjusted to the stability limit at the grid resolution. The error for a given wavenumber decreases as $\widetilde{k}^3$, and is comparable to the spatial errors at the previously discussed design point.

3.3 Pressure, mass conservation and boundary conditions

We discuss in this section the relation between the computation of the pressure, the conservation of mass, and boundary conditions. The latter can be imposed either in the velocity equation (13) or in the pressure equation (14), but generally not in both. We have found in the literature no example in which the pressure is enforced directly. The desired pressure distribution is usually obtained by prescribing the velocities at the boundary [2, 24]. However, while that is relatively easy to do when the desired boundary conditions are static, it becomes more complicated when they are allowed to change in time, such as in control applications involving arbitrary mass injection. In those cases, for example, there is no automatic guarantee that the sum of all the mass fluxes across the boundary vanishes at all times, as should be the case for incompressible flow.

Simulations generally try to avoid these problems. In [24], for example, a pressure gradient was imposed by a distribution of transpiration velocities with approximately zero net mass flux, and in [44] mass conservation was ensured by unspecified adjustments to the velocities at the exit plane of the numerical domain.

In many of those cases, it would have been more convenient to impose the desired pressure directly, but, although the newer treatments of the fractional step method [28, 9] provide a mechanism to do so through the boundary conditions in equation (14), the problem is far from trivial and has received little attention. Following this unfortunate tradition, we only deal in this section with the velocity boundary conditions used for the test cases discussed later.

The background of the problem is the separation into two parts of the fractional-step method. The advective step (13) creates a velocity field, u^*, which is not solenoidal and on which we impose boundary conditions that we would like to be those of the final velocity. The pressure correction step (15) attempts to enforce continuity without changing those boundary velocities. That is impossible in general, and recipes have been devised to modify the conditions on u^* so that the final boundary velocities are close to the desired ones [15]. The improvement that they afford is comparable to that of the formulation in terms of the pressure increment, and both techniques are not usually applied together. Even if they were, the boundary corrections proposed up to now only modify the tangential boundary velocities, and assume that the normal ones are not changed by (15). This is formally equivalent to requiring that the pressure satisfies

$$\frac{\partial(\Delta p)}{\partial n} = 0, \tag{33}$$

at the domain boundary $\partial\Omega$, where n is the outer normal. The problem is that the Poisson operator with this boundary condition is singular, because it is invariant to the addition of a constant pressure, and that the discretized version DG usually inherits both that invariance and the singularity. Singular operators have two properties. Their solutions are not unique, and the inhomogeneous problem can only be solved for some particular class of right-hand sides. The first problem is easily solved, for example by arbitrarily fixing the pressure at one point. The second one requires that the divergence of u^* satisfies a compatibility condition that is essentially equivalent to global mass conservation. In the continuum case the Poisson problem is,

$$\nabla^2(\Delta p) = \nabla \cdot u^*. \tag{34}$$

Integrating over the domain Ω, and applying Gauss' theorem, leads to

$$\int_\Omega \nabla^2(\Delta p)\,\mathrm{d}\Omega = \int_{\partial\Omega} \partial_n(\Delta p)\,\mathrm{d}\partial\Omega = \int_\Omega \nabla\cdot u^*\,\mathrm{d}\Omega = \int_{\partial\Omega} u_n\,\mathrm{d}\partial\Omega = 0. \tag{35}$$

The discrete equivalent is

$$\psi^T DG\Delta p^{n+1} = \psi^T D u^* = \phi^T u^* = 0, \tag{36}$$

where the T superscript denotes transposition, ψ is the null vector of the DG operator, and $\phi = D^T\psi$. The three equations in (36) correspond to the last three equalities in (35).

Physically, ψ is the set of weights for each grid point in the volume integral $\int \mathrm{d}\Omega$, while ϕ is the set of weights in the boundary integral of the fluxes. The latter is typically non-zero only for points near or at the boundary. For a second-order discretization in a uniform grid, ψ is unity everywhere, but for non-uniform grids, or for more complicated operators, it has to be obtained from the solution of $(DG)^T\psi = 0$. Similarly, in the simple second-order uniform case, the boundary vector ϕ is unity for the normal velocities at the boundary, and vanishes otherwise.

Equation (36) provides a tool for generating the compatible boundary conditions that have to be satisfied by u^* before it is used in the pressure equation (14). In open systems, such as boundary layers, one usually knows the velocity over part of the boundary, and tries to estimate it over the rest. In a typical example, one prescribes the velocity at the inflow and at the lateral boundaries, and estimates the outflow using some advective boundary condition designed to minimize numerical reflections. This first guess of the outflow would be, for example,

$$\partial_t \vec{u}^* = -U_c \partial_x \vec{u}^*, \quad \text{at} \quad x = L_x, \tag{37}$$

with some upwinded discretization for ∂_x. Such outflow conditions may not guarantee continuity, and have to be corrected. Assume that we decide to modify our outflow plane by some small uniform increment. In our code, this is applied over the whole outflow line of the $k_z = 0$ Fourier mode of the u component. This means that, for this mode,

$$\partial_t u_0^{**} = -U_c \partial_x u_0^* + \alpha, \quad \text{at} \quad x = L_x. \tag{38}$$

In discrete terms

$$u_0^{**} = u_0^* + \alpha \mathbf{1}_s, \tag{39}$$

where $\mathbf{1}_s$ is one in the last column, and zero elsewhere. Other wall-normal distributions for the outflow correction can be trivially accommodated. Introducing (39) in (36), we obtain the magnitude of the required correction,

$$\alpha = -\frac{\psi^T D u_0^*}{\psi^T D \mathbf{1}_s} = -\frac{\phi^T u_0^*}{\phi^T \mathbf{1}_s}. \tag{40}$$

Note that the denominator in this equation is a number that only needs to be computed once for each simulation. Other components do not contribute to continuity at the outflow, and the Poisson operator for the higher Fourier modes is not singular.

This outflow correction could be interpreted as the decomposition of the pressure Δp into a part due to u^*, which does not satisfy the Neumann condition at the exit, and a second component, due to $\alpha(I - \Delta t\, L)^{-1}\mathbf{1}_s$, which corrects the boundary. This component is a real physical effect, and corresponds to the pressure gradient needed to evacuate through the downstream boundary the extra fluid injected by the unbalanced inflows elsewhere. It is the principal limitation of velocity boundary conditions, which are effectively equivalent to solving the boundary layer inside a channel of finite height, although with an upper free-slip wall. They are therefore not subject to viscous effects at the upper boundary, but they suffer from the same blocking problems as wind tunnels of limited height.

It is important to consider at which moment to apply the outflow correction. Of the two expressions in equation (40), the first one appears to depend on the whole u_0^* field, but the

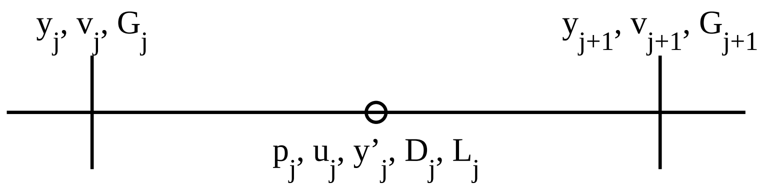

y_j, v_j, G_j
y_{j+1}, v_{j+1}, G_{j+1}
p_j, u_j, y'_j, D_j, L_j

Figure 3: Sketch of the wall-normal grid. The u_j are located in $y'_j = (y_{j+1} + y_j)/2$. The range of indices for the different variables are, y, $v \in (1 \dots N+1)$; p, u, D, $L = DG \in (1 \dots N)$; $G \in (2 \dots N)$.

second one shows that it is essentially a property of the boundary. If we choose to use the first expression, the correction can only be applied after the evolution step (13), because u_0^* is unknown until then. On the other hand, the boundary conditions are known a priori, and the correction can be applied before (13). The latter is preferable, because it guarantees that all the boundary conditions are satisfied after the time step. Otherwise, the extra pressure introduces spurious slip velocities of $O(\alpha\Delta t)$ at the boundaries. Those velocities are potentially $O(\Delta t)$ if the boundary conditions change discontinuously, but, in practice, boundary conditions usually evolve smoothly in time, $\alpha = O(\Delta t)$, and the slip velocities are of the same quadratic order as those generated by the rest of the method.

In the particular case in which the Poisson equation is solved by cosine expansions, the problem is further simplified, because a lot of the work can be done analytically. In that case, the only singular equation is (31) with $k_z = k_x = 0$, which reduces to

$$\partial_{yy}\widehat{p}_{00} = \partial_y \widehat{v}_{00} + \frac{\widehat{u}_0(L_x,\, y) - \widehat{u}_0(0,\, y)}{L_x}, \tag{41}$$

where $\widehat{p}_{00}$ is the mean value of $\widehat{p}_0$, taken over x. Note that the second part of the right-hand side is actually the $k_x = 0$ discrete Fourier mode of $D_x\widehat{u}_0$, and that (41) is only true if the divergence is computed using the second-order formula (29). Higher-order formulas give rise to slightly more complicated boundary terms, but we have seen that cosine expansions should not be used for them.

It is tempting to integrate this equation once,

$$\partial_y \widehat{p}_{00} = \widehat{v}_{00}(y) - \widehat{v}_{00}(0) + L_x^{-1} \int_0^y [\widehat{u}_0(L_x,\, y) - \widehat{u}_0(0,\, y)]\, \mathrm{d}y, \tag{42}$$

to reduce the influence of $\widehat{v}_{00}$ to the boundaries, and to require that the right-hand side should vanish at $y = L_y$, but this has to done consistently with the numerical operators. This case provides a simple example of the theory just developed, which can be solved analytically and used directly in the code. Assume that the divergence and gradient operators are those (29–30), and that the vertical grid is the one in figure 3. Introduce the notation $h_j = y_{j+1} - y_j$. The divergence and gradient operators become,

$$D_j v = \frac{v_{j+1} - v_j}{h_j}, \tag{43}$$

$$G_j p = 2\frac{p_j - p_{j-1}}{h_j + h_{j-1}}. \tag{44}$$

The Laplacian is

$$L_j p = \frac{2p_{j+1}}{h_j(h_{j+1}+h_j)} - \frac{2p_j(h_{j+1}+2h_j+h_{j-1})}{h_j(h_{j+1}+h_j)(h_j+h_{j-1})} + \frac{2p_{j-1}}{h_j(h_j+h_{j-1})}, \tag{45}$$

for $j = 2 \ldots N-1$, and

$$L_1 p = \frac{2(p_2 - p_1)}{h_1(h_2+h_1)}, \qquad L_N p = \frac{2(p_{N-1}-p_N)}{h_N(h_N+h_{N-1})}. \tag{46}$$

The index ranges for the different operators are given in the caption of figure 3.

The null vector ψ satisfies the recursion relation $\psi^T L = 0$,

$$\frac{\psi_{j-1}}{h_{j-1}}\,\frac{1}{h_j+h_{j-1}} - \frac{\psi_j}{h_j}\,\frac{h_{j+1}+2h_j+h_{j-1}}{(h_{j+1}+h_j)(h_j+h_{j-1})} + \frac{\psi_{j+1}}{h_{j+1}}\,\frac{1}{h_{j+1}+h_j} = 0 \tag{47}$$

which is solved by $\psi_j = h_j$, $j = 1 \ldots N$. For the boundary weights, $\phi_T = \psi_T D$, we obtain

$$\phi_1 = -1, \quad \phi_{N+1} = 1, \quad \text{and} \quad \phi_j = 0 \quad \text{otherwise.} \tag{48}$$

The orthogonality condition for the right-hand side of (41) is then

$$C \equiv \widehat{v}_{00;\,N+1} - \widehat{v}_{00;\,1} + L_x^{-1}\sum_{j=1}^{N} h_j\,[\widehat{u}_0(L_x)_j - \widehat{u}_0(0)_j] = 0, \tag{49}$$

which has an obvious integral interpretation. As in the general case, we can now choose which boundary condition to change. If we for example decide to adjust the outflow by a constant increment α, we obtain that $C + \alpha L_x^{-1}\sum h_j = 0$. The general theory thus recovers the intuitive result that the integral of all the boundary normal velocities should vanish, although it gives some information on how to integrate in y that is not immediately obvious. In more general cases, intuition is not always enough, but the general procedure can still be applied, at least numerically. The boundary vector, for example, determines which are the natural boundary conditions to impose for the normal velocities, which in higher-order schemes usually involve more than the boundary point itself.

3.4 Parallelization of the code

Parallelization is required to allow the code to simulate turbulent boundary layers at high Reynolds numbers. Our code is designed for large distributed-memory machines, using the MPI message passing library.

Two approaches can be followed. The first one is to develop distributed matrix solvers or Fourier transforms. The second one is the method used for example in [14], and described in [12], in which the database is transposed between serially processing each coordinate direction.

The comparative advantages of both approaches were analysed early in the development of parallel computing. For example, the different implementations of the fast Fourier transform, and their application to pseudospectral methods, were examined by [7, 27]. Both concluded that the operation and communication counts for the distributed and transpose methods are asymptotically similar, and that the choice depends on the particular architecture of the machine being considered. A good example is the comparative study in [43] of identical pseudospectral simulations in two similar machines, one a hypercube and the other with mesh internode connectivity. The code used a transpose method for the FFT, originally optimized for the hypercube, and the performance degraded substantially in the supposedly faster mesh machine. The degradation persisted even after the transpose had been optimized for the mesh. Communication networks have improved since then, while local memory hierarchies have become more complicated, and the transpose method is nowadays usually preferred. The MPI parallel multidimensional transforms of the fftw library, for example, use the transpose method.

The transpose approach has the advantage of being easy to code and of keeping latency to a minimum. Suppose that the equations are discretized on $n_x \times n_y \times n_z$ points, and that the parallel calculation is done on n_p processors. The transpose method first partitions the data across the z-direction over the processors available. At this point every processor contains arrays of size $n_x \times n_y \times (n_z/n_p)$. Each processor contains a number of full $x-y$ planes, and the derivatives in the x and y directions can be calculated as if we were working in a serial processor. The Fourier transforms along z cannot be done locally at this point. They are made possible by transposing the data to a partition across x, which means that every processor now contains arrays of size $(n_x/n_p) \times n_y \times n_z$. At this point, operations along z can be done locally, but it is impossible to calculate the x-derivatives, due to the global nature of the compact finite differences and of the cosine transforms. It is clear that large amounts of data are sent, stressing the necessity of a fast communication network.

For the number of grid points typically expected in our simulations ($10^8 - 10^{10}$ Mpoints), and the number of nodes available (100–1000), each node is able to accommodate at least one grid plane, so that only two transposes per variable are needed per time substep. The choice of which directions to transpose is important. In boundary layers, the longest dimensions are typically x and z, so that slicing across them allows the largest number of nodes to be used. Slicing in $x-z$ planes would actually be slightly more convenient, since the two FFT directions would be in-node, and parallel algorithms with reasonable communication counts are available for matrix computations [23], but that choice would have limited the number of nodes to the order of 200–500 before a new distributed direction had to be included. Moreover, many of the parallel matrix algorithms lose their advantage when only one plane is allocated to each processor. As a consequence our code switches between $z-y$ and $x-y$ slices. In the latter, each processor contains only the real or the imaginary part of a complex Fourier mode, which need to be switched at some point between adjacent processors. This one-to-one communication is however much faster than the all-to-all communication involved in the transposition.

The parallel performance of the code depends on the particular machine, and on the node count. While, for example, the communication load is of the order of 15% in our 128-node departmental Xeon cluster, it can reach 45% in 512 nodes of the Barcelona supercomputing centre. The difference reflects in part the balance of processor and communication speeds in both machines, but also software differences such as the availability in the former, but not in the latter, of asynchronous transfers running concurrently with local processor instructions.

4 Test cases

The code was tested on two sets of cases. The first one is a zero-pressure-gradient (ZPG) boundary layer with a turbulent inlet, intended to test the inflow conditions and the basic performance of the code.

The second one consists of three different adverse-pressure-gradient boundary layers with separation. The inflows are laminar, and the pressure gradient separates the layers, which later transition and reattach. In the three cases the pressure gradient remains adverse until the end of the simulation box, except for a short adaptation region at the outlet for numerical purposes. The pressure gradients are however different, and so are their effects. While the reattached layer with the weakest gradient is not very different from the ZPG case, those with the strongest ones tend to separate again, and would have probably separated in a longer box.

Both sets of results are compared with published experimental and numerical data.

4.1 Zero Pressure Gradient

This case was designed to span Reynolds numbers in the neighbourhood of $Re_\theta = 700$, for which there are several previous numerical and experimental cases.

The turbulent inflow is generated from an implementation of the method in [20], in which the instantaneous flow field in a reference plane, $x = x_{ref}$, is copied at each time step to the inlet at x_{in}, after some re-scaling to account for the growth of the layer in the intervening space. The simulation was initialized using a pre-stored flow field from the simulation in [40], but this reference was abandoned after the initial wash-out. After that moment, the simulation was fully self-contained.

The main choices of this method are the location of the reference plane, and the scaling used to estimate the boundary layer thickness and the friction velocity at the inflow. In our case we use as the scaling length for the outer flow the Clauser-Rotta thickness

$$\Delta = \frac{\delta^* U_\infty}{u_\tau}, \tag{50}$$

where δ^* is the displacement thickness, and u_τ is the friction velocity. Our goal is to achieve a given inflow Reynolds number, $Re_{\theta,0} = U_\infty \theta_0/\nu$, given the measured $Re_{\theta,ref}$, where θ is the momentum thickness. The parameters needed for rescaling are the boundary layer thickness at the inflow, Δ_0, and the friction velocity at that point. From the ratio of the Reynolds numbers we compute θ_0, and, from it, Δ_0, given the measured Δ_{ref}, and assuming as a first approximation that θ is proportional to Δ. Finally, following [20], the inflow friction velocity is estimated as

$$u_{\tau,0} = u_{\tau,ref} \left(\frac{Re_{\theta,ref}}{Re_{\theta,0}} \right)^{1/8}. \tag{51}$$

The rest of the boundary conditions are straightforward. The free stream velocity is fixed (to $U_\infty = 1$) at the inlet plane. The vorticity and the vertical velocity are made to vanish at

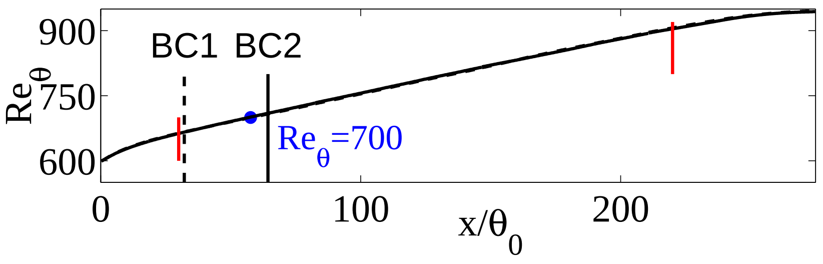


Figure 4: Reynolds number development for the zero-pressure-gradient test case. The two vertical lines labelled BC1 and BC2 mark the locations of the two reference planes used to compute the turbulent inlet condition in two different runs. The two results are plotted as dashed (BC1) and solid lines (BC2) for Re_θ, barely distinguishable. The two short vertical lines are the limits of the 'useful' numerical range.

the upper boundary, and the resulting mass imbalance is accommodated by an extra streamwise pressure gradient, as explained in the previous section. As a result, the boundary layer is slightly accelerated, with

$$\beta = \frac{\delta^*}{u_\tau^2}\frac{\partial P}{\partial x} \approx -0.04. \tag{52}$$

This gradient is within the range of those found in many laboratory experiments considered as nominally ZPG. The Reynolds number, Re_θ, varies from around 600 to 950, but only the range 660–900 can be considered to be relatively unaffected by the inflow and outflow boundary conditions (figure 4). The box size is $L_x \times L_y \times L_z \approx (185 \times 38 \times 38)\theta_{end}$, where the momentum thickness is measured at the 'effective' exit section where $Re_\theta = 905$. This corresponds to roughly $19 \times 4 \times 4$ times the boundary layer thickness at the effective outflow.

The simulation requires $n_x \times n_y \times n_z = 1282 \times 258 \times 384$ collocation points, or roughly 120 Mpoints. It uses about 20 CPU-minutes per time step, to be divided by the number of processors. It ran in several parallel machines, from 32 to 128 processors, with reasonably linear scaling. The time stepping was run at a constant $CFL = 0.6$, which is limited by the longitudinal velocity at the free stream. This implies about $n_x/CFL \approx 2000$ time steps per wash-out time, $T_w = L_x/U_\infty$. The statistics presented below were compiled over 20,000 time steps, which represent ten washouts. For this particular simulation, this is equivalent to roughly the same number of eddy-turnover times, $T_e = \delta_{end}/u_\tau$.

The streamwise numerical resolution is $\Delta x^+ \approx 6.3$. In the wall-normal direction Δy^+ changes from 0.7 at the wall to about 5 at the edge of the boundary layer. In terms of the local Kolmogorov viscous scale, η, the grid never exceeds $\Delta y \approx 2.5\eta$. In the spanwise direction, $\Delta z^+ \approx 6.5$ per Fourier mode, or about 4 in terms of collocation points. Those values are comparable to the ones used in spectral simulations of channel flow, even after taking into account the reduced resolution of the finite differences.

The evolution of the Reynolds number of the boundary layer is shown in figure 4. Two cases were run, with identical parameters but with different locations of the reference plane for the inflow condition, $x_{ref}/\theta_0 = 32$ (BC1), and $x_{ref}/\theta_0 = 64$ (BC2), where θ_0 is measured at the inflow.

There are only a few differences between the two cases, for which snapshots of longitudinal

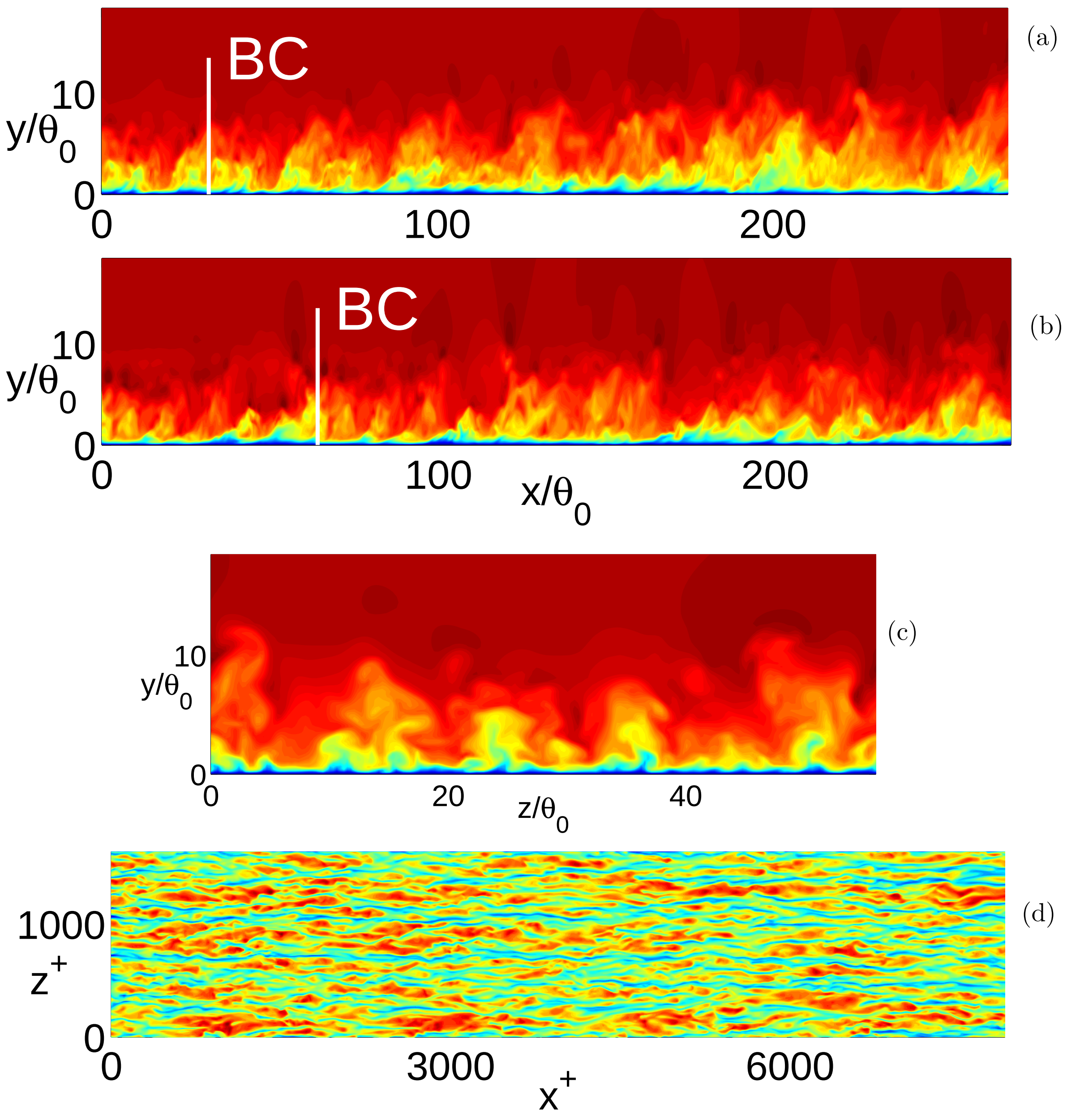


Figure 5: Instantaneous fields of the streamwise velocity. (a) On a longitudinal (x, y) plane, BC1. (b) Same for BC2. (c) On a transverse (z, y) plane, $Re_\theta = 800$. (d) On a wall-parallel plane, $y^+ = 15$.

planes of the streamwise field are shown in figures 5(a) and 5(b). There is in BC1 a noticeable streamwise periodicity in the flow field, with a wavelength of the order of the distance between the reference and inflow planes. This is probably the result of a weak feedback instability caused by the inflow condition as it travels to the reference plane, and is typical of many delay equations. If that were the case, it would be difficult to avoid, and it could only be weakened by moving the reference plane far enough from the inflow to allow for the phase of the velocity to decohere. In fact, the observed periodicity is much weaker in the case BC2, and figure 5 suggests that it would be essentially absent if x_{ref} had been chosen beyond roughly $100\theta_0$.

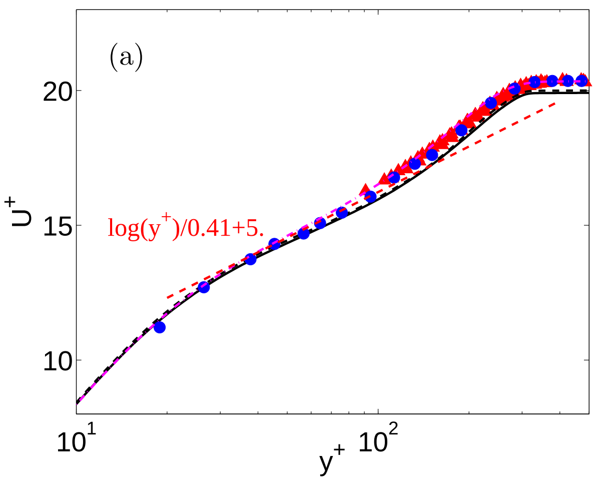
Some comments may be in order at this point on our implementation of the rescaling method in [20]. In the original paper, which was intended to feed an LES, there are two distinct simulations. Flow A is an auxiliary DNS code that runs a short initial stretch of the boundary layer, and whose only purpose is to provide the reference plane to be rescaled. One of its intermediate planes is then filtered as needed, and fed to the inflow of the LES code B. In our case, in which both codes are DNSes, there is no need for different codes. Flow A can be considered to be the part of the box contained between x_0 and x_{ref}, and the plane being fed to flow B could be any of the planes in that sub-box. There is nothing around x_{ref} to make that location special, and to require it to be discarded. It is just a location in which the flow is being observed. As such it could be expected that, even if the initial part of the box may be unphysical, that effect should appear in our simulation as an adaptation region for the inflow condition. In particular, we observe in our case that the pressure fluctuations near the inlet are too high, most probably because the rescaled plane does not satisfy continuity.

On the other hand, we found no problem in maintaining the flow. Such problems have been reported by several authors [11, 19], who have proposed remedies. In those cases, the trouble seems to be associated with the choice of the field used to initialize the simulation, which did not have the right Reynolds stresses. In our case, we found no such problems after initializing the upstream part of our box with a pre-stored field of the simulation in [40]. We also found helpful to use, during the first flow-through, pre-stored mean velocity profiles for the reference plane, and to keep after that a fixed *mean* velocity profile at the inflow.

Transverse and wall-parallel planes of the velocity field in BC2 are shown in figures 5(c) and 5(d), and they have all the characteristics of healthy low-Reynolds-number wall-bounded turbulence.

Mean and fluctuating velocity profiles are shown in figure 6, compared with other simulations, and with experiments at comparable Reynolds numbers. They generally agree, although the scatter among experiments and within simulations is considerable. This seems to be a property of low-Reynolds-number boundary layers, which preserve some memory of transition or of the tripping process. Our simulations are essentially tripped by the inflow condition, and that tripping is different from those in other cases. The experiments in [10] were specifically designed to test this effect. They were tripped in three different ways around $Re_\theta = 500$, and they are therefore roughly comparable to our simulations. The conclusion was that the effects of tripping were noticeable at $Re_\theta = 700$, but essentially absent for $Re_\theta = 1000$. This is broadly consistent with figures 5(ab), where the effect of the inflow disappears beyond $x/\theta_0 = 150$ ($Re_\theta \approx 800$).

Figure 7 summarizes the friction factors of these and of other experiments, and supports the same conclusion. Below $Re_\theta \approx 1000$, the data are widely scattered, including our simulations, and



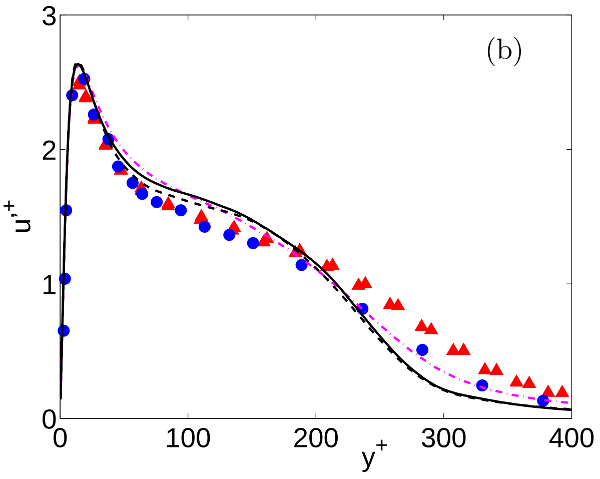


Figure 6: (a) Mean streamwise velocity. (b) Longitudinal fluctuation intensities. $Re_\theta \approx 700$. ---- , present simulation, BC1; ——— , BC2; —·— , Ref. [40] at $Re_\theta = 670$; ● , Ref. [29]; ▲ , Ref. [10].

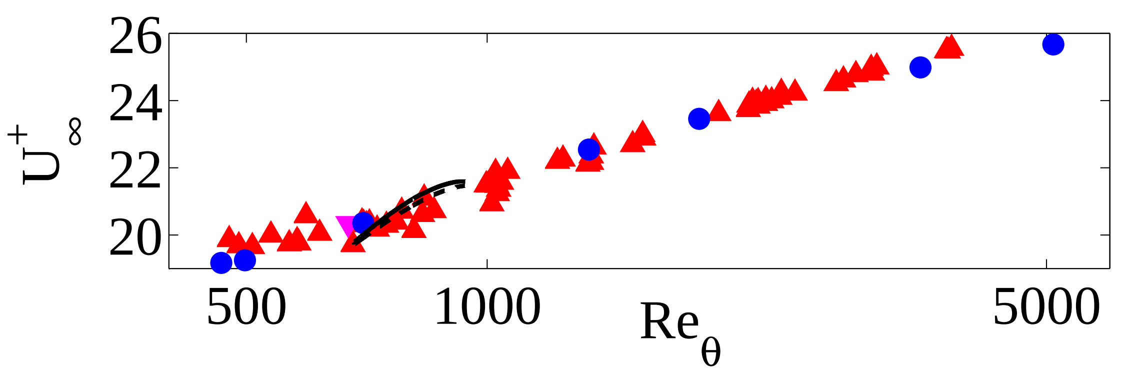


Figure 7: Friction coefficient, given as $U^+_\infty = (2/c_f)^{1/2}$, as a function of Re_θ. Symbols as in figure 6, except that Ref. [40] is here ▼ .

they only collapse to a single curve above that value. The general conclusion is that an interval of at least about $100 - 150\theta_0$ is needed before the effect of the inflow conditions is approximately forgotten. A recent survey in [5] of the streamwise development in boundary layer experiments concludes that tripping effects persist in the lowest-order statistics for $\Delta Re_\theta \approx 200$, which is roughly consistent with our previous criterion. Outer-flow quantities can take considerably longer to equilibrate.

4.2 Adverse Pressure Gradient

Boundary layers subject to an adverse pressure gradient (APG) are a more stringent test for a code than ZPG cases, in part because they deviate more strongly from conditions of parallel flow, but mostly because they generate stronger fluctuations away from the wall. This complicates both the choice of the grid resolution and of the outflow conditions. In fact, when running

		n_x	n_y	n_z	L_x/δ^*	L_y/δ^*	L_z/δ^*	$\beta = \delta^* \partial_x P / u_\tau^2$
APG1	◆	769	256	1024	29	3	17	$50-100$
APG2	■	769	256	512	27	4	9	$60-100$
APG3	●	769	256	512	68	10	21	8

Table 1: Parameters for the test cases with adverse pressure gradients.

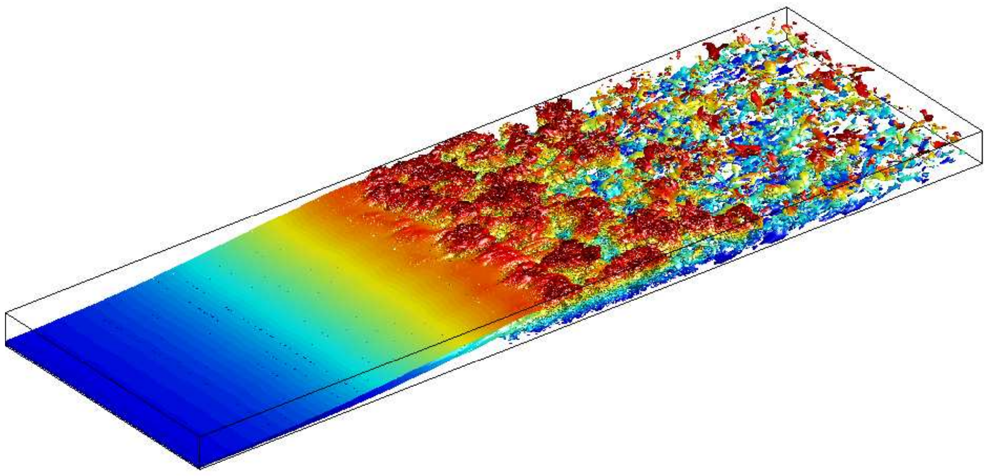


Figure 8: Simulation APG2. The representation is an isosurface of spanwise vorticity, coloured by the distance to the wall. Note the separation in the first third of the box, and the reattachment upon transition.

APG cases with the present code, it proved necessary to include near the outflow a short buffer region of higher viscosity and of favourable pressure gradient. That was not required in the ZPG case. Imposing the desired pressure gradient also presents problems, specially in the presence of separation, because the rapid growth of the layer interacts strongly with the mean stream.

We discuss here three simulations, at relatively low Reynolds numbers, which were undertaken in the context of turbomachinery flow. In all the cases the flow separates while still laminar. It then transitions within the separated bubble, and, as a consequence, reattaches. Preliminary results have been reported in [32, 33, 34, 35, 36], and will be discussed in detail elsewhere. A general view of the vorticity field of one of the simulations is shown in figure 8. Here we are only interested in the ability of the code to simulate turbulent boundary layers, and we will therefore centre on the short APG region that develops after reattachment.

The basic numerical parameters of the simulations are summarized in table 1. In all cases the flow is forced by a constant suction velocity at the top of the numerical box, intended to generate a uniform deceleration of the free stream. The limitations of this boundary condition are evident from figure 9(a), which displays the mean streamwise velocity near the top of the boundary layer. All the cases have a plateau above the separation bubble. This is a physical effect, present in real flows, but it complicates the control of the simulation. For example, the main difference between the simulations APG1 and APG2 is the height of the simulation box, which proved to be too low in the first case. The suction velocity in the second case was chosen

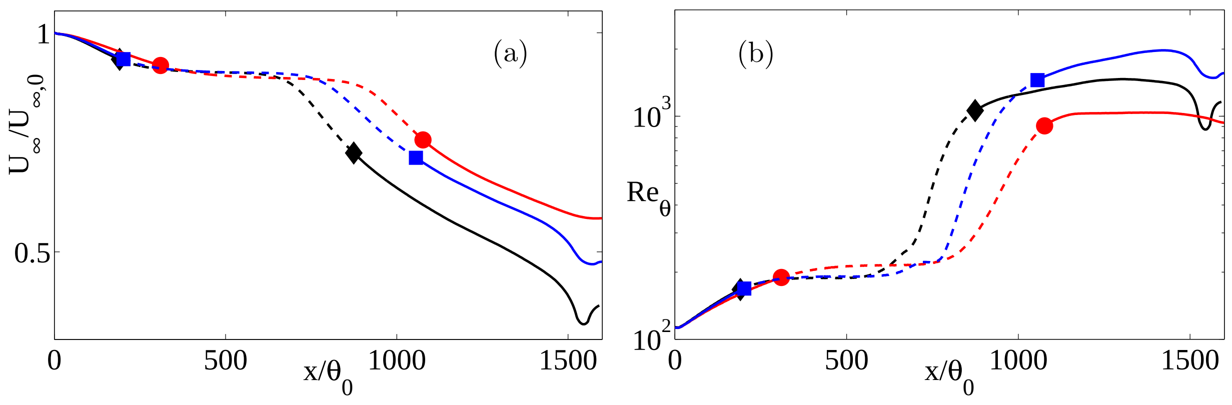


Figure 9: Adverse-pressure-gradient simulations. (a) Mean streamwise velocity above the boundary layer. (b) Evolution of the Reynolds number. Symbols as in table 1. Solid lines denote attached regions, and dashed ones are separated.

with the intention of matching the velocity distribution of the first one, but it is clear from the figure that this was only imperfectly achieved. Imposing the pressure gradient directly at the upper numerical boundary has similar problems, unless the box is very tall. The only way to obtain a given outer potential flow would probably be to iterate either the suction velocity or the pressure distribution to match them to the outer solution at some particular height. Note that this difficulty is shared by laboratory experiments.

A further ambiguity of the present cases concerns transition. Spectral codes, such as the present one in the spanwise direction, do not break the initial symmetries, and do not therefore develop three-dimensionality unless seeded. In the present cases, which were intended to match an existing experiment, a steady three-dimensional perturbation was added to the inflow laminar boundary layer, with a maximum r.m.s. amplitude $u'/U_\infty \approx 0.006$. This is enough to force transition and reattachment reasonably early in the separation bubble, and provides us with the desired APG attached flows. The evolution of Re_θ is given in figure 9(b). It grows rapidly through separation, and reaches values at the exit that are somewhat higher than those of the ZPG case considered in the previous section.

Note the effect of the outflow boundary in both figures 9(ab), which extends roughly over the last 15% of the box. In the following, we will concentrate on the attached APG layer towards the end of the box, discarding in all cases the final 15%. The friction coefficients over those regions are given in figure 10(a). For the purpose of the present comparison, we arbitrarily choose to consider as useful only the segment downstream of the point in which c_f first reaches 80% of its maximum value. In the following we will only discuss results in this region, with the abscissae referred to the point of reattachment, and normalized with the momentum thickness at that point. The limits are marked in figure 10(a).

The pressure gradients in the free stream are given in figures 10(bc), normalized both in outer units,

$$\beta = \frac{\delta^*}{u_\tau^2}\frac{\partial P}{\partial x}, \tag{53}$$

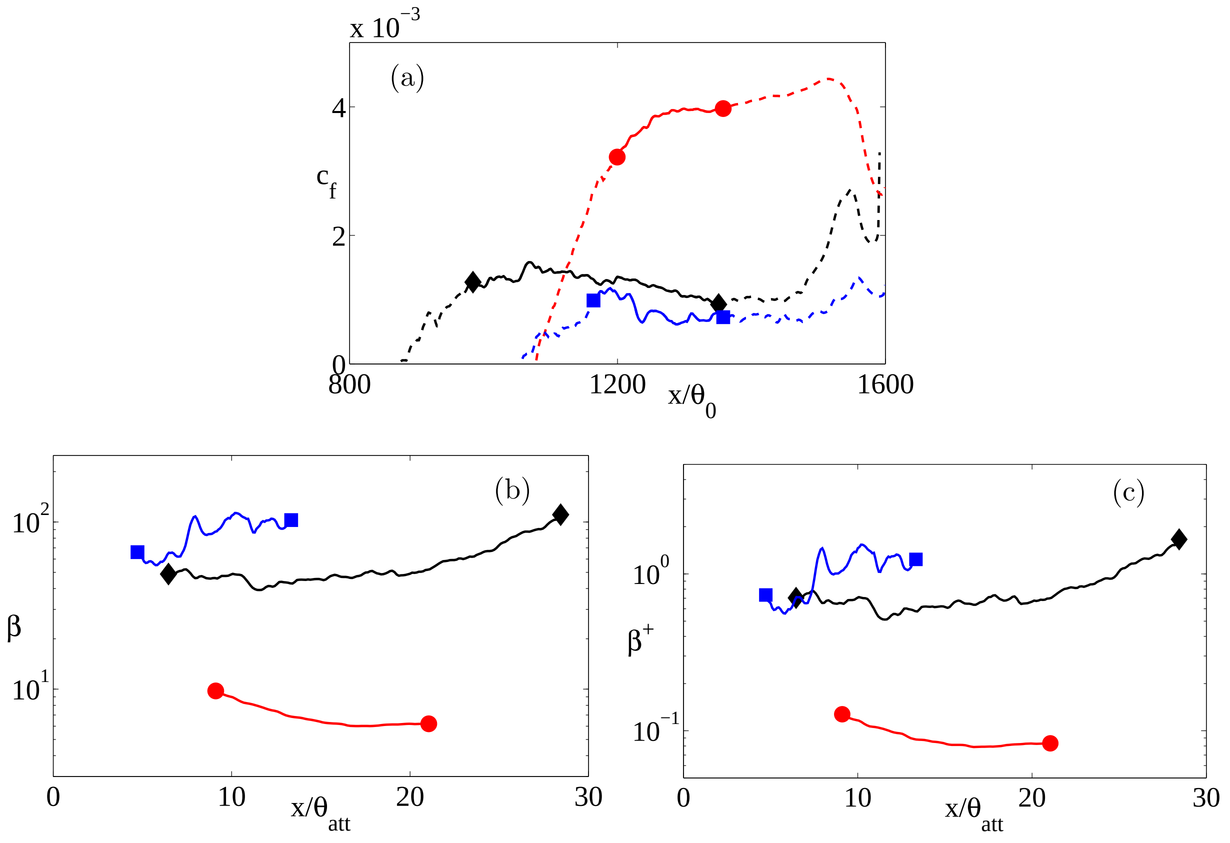


Figure 10: (a) Friction coefficient over the reattached regions of the APG simulations. The symbols, and solid lines, mark the limits of the segments considered in the following as useful. (b) Normalized pressure gradients for the simulations, measured over the useful regions defined in the text. Outer units. (c) Wall units.

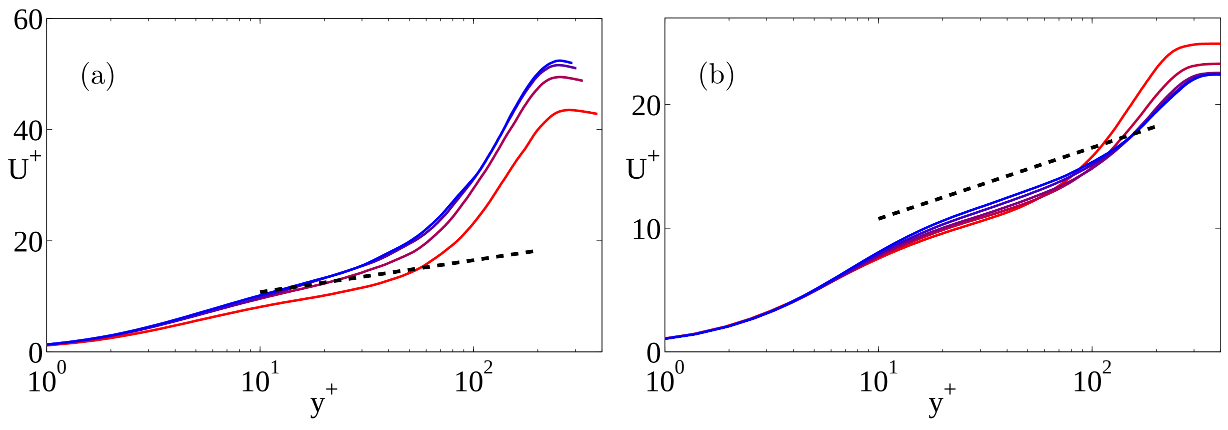


Figure 11: Evolution of the mean velocity profile in the reattached region. The layer progresses downstream from red to blue. The dashed line is $U^+ = \log(y^+)/0.4 + 5$. (a) APG2. (b) APG3.

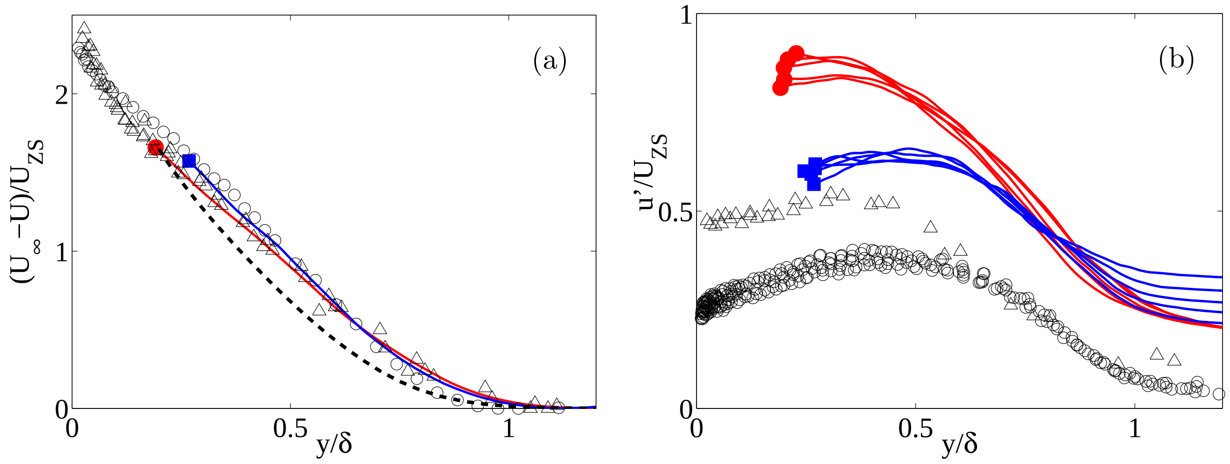


Figure 12: (a) Mean velocity profile in defect form, scaled as in [45]. (b) R.m.s. fluctuations of the streamwise velocity. Solid lines are present simulations. In (a) only the profile at the end of the useful range is included, to avoid clutter. ○ , APG boundary layer from [37]. $Re_\theta = 40,000 - 53,000$, $\beta = 20$. As before, (a) also includes the station $Re_\theta = 44,000$; △ , Recovering ZPG boundary layer from [39]. $Re_\theta = 2 - 12 \times 10^3$. $(x - x_{att})/\theta_{att} \approx 30 - 70$. The dashed line in (a) is the reference ZPG layer in [40]. Only $y^+ > 60$ is plotted in all cases.

and in inner ones,

$$\beta^+ = \frac{\nu}{u_\tau^3}\frac{\partial P}{\partial x}. \tag{54}$$

The first normalization measures the effect of the pressure gradient on the outer flow, while the second one measures its influence on the viscous layers. The figures suggest that, while APG3 is likely to be affected by the pressure gradient only away from the wall, both APG1 and APG2 should be affected everywhere. It is clear from the figures that these two latter cases have roughly similar parameters, and from now on we will only use APG2, since we have mentioned that APG1 had a marginally low numerical box. It is obvious from figures 10(bc) that none of the reattached layers is long enough to satisfy the criterion discussed in the previous section to reach equilibrium. Figures 11 show the evolution of the velocity profiles in the two cases. Both reach some semblance of equilibrium of the outer wake component towards the end of the simulation, but in none of the two does the logarithmic layer reach anything close to an equilibrium behaviour. It should however be noted that the Reynolds numbers are too low to develop a healthy logarithmic layer, and that [4] have suggested that APG layers reach equilibrium faster than ZPG ones.

It was also suggested in [4] that, when the mean velocity defect is properly scaled, all the APG boundary layer should quickly collapse to a universal form. Testing that conclusion is beyond the scope of the present report, but our profiles are compared in that way to other boundary layers in figure 12(a). Our profiles are plotted in defect form at the end of their useful range, scaled with

$$U_{ZS} = U_\infty \frac{\delta^*}{\delta}, \tag{55}$$

as suggested in [45]. That scaling forces all the curves in figure 12 to have unit area, but its main utility is that, if the profiles collapse, U_{ZS} is the characteristic velocity difference for the boundary layer outside the near-wall region. For example, if we consider the outer layer as a free shear flow, its characteristic velocity gradient would be U_{ZS}/δ. Our profiles are compared in this way to the APG boundary layer in [37], which has a comparable value of β, although a much higher Reynolds number, and to the ZPG in [39], which is recovering from separation behind a backwards facing ramp. The latter is an interesting experiment, because it spans Reynolds numbers of the same order as our simulations, and because it should help in differentiating the effect of the recovery from that of an equilibrium APG. It is seen in the figure that the present simulations match the recovering layer better, specially near the free stream, although all the cases are far from the reference ZPG profile.

A more revealing comparison is that of the fluctuation intensities, which are shown in figure 12(b). In APG boundary layers, as opposed to ZPG ones, the peak fluctuation intensity moves away from the wall into the middle of the outer flow, and resembles the fluctuation profiles found in free-shear flows. The scaling of these profiles is still controversial, but [21] have recently reported that U_{ZS} collapses the fluctuations best within a given experiment. They however find that different experiments collapse to different profiles, and suggest that no universal behaviour can be expected. This seems to be the case in figure 12(b). Each of the four experiments or simulations collapse to a different curve, and all that can be said is that at least all the peaks of the fluctuations profiles are at roughly the same location. This figure also supports the conclusion from the mean velocities that our simulations are not equilibrium APG boundary layers, and that they are actually relaxing from separation. Given their recent history, and their short fetch, that is reasonable. It is also clear from this figure that the fluctuation level in the free stream is unacceptably high. Since that was not the case in the ZPG layer in figure 6(b), the reason is probably the insufficient height of the computational box, which was much taller in the ZPG case.

5 Conclusions

A second-order-accurate code has been developed for the simulation of spatially-developing boundary layers, with resolution properties comparable to those of fourth-order compact finite differences. This code has been tested by simulating both a zero-pressure-gradient turbulent boundary layer at moderate Reynolds numbers $Re_\theta = 650-900$, and laminar separation bubbles that transition to turbulence and reattach. In the latter cases, the Reynolds numbers of the reattached turbulent APG boundary layers reach $Re_\theta \approx 1500$.

The ZPG simulation agrees well with previous experimental and numerical results at similar Reynolds numbers. Most of the observed differences can be attributed to residual effects of the tripping, which in this case is done by the inflow boundary condition. Similar effects are found in experiments at these Reynolds numbers, and the simulations are within the experimental scatter. A criterion was derived for the minimum required development length, $150\theta_0$.

The APG cases were created by a uniform suction over the whole upper boundary of the numerical domain. This generated a roughly uniform gradient, and the reattachment was entirely due to transition. No favourable gradient was applied, except for a short buffer region at the outflow, for numerical reasons. Two cases were run, with respectively strong and weak pressure gradients. Since the emphasis here is on the behaviour of the code in turbulent flows, only the results in the reattached regions have been compared with experiments. In both cases the reattached layers are too short to reach equilibrium, but the results agree qualitatively with the few available experiments, resembling most the recovery region of boundary layers as they reattach behind obstacles.

The code was run in several distributed-memory machines, with fairly linear scaling in the range of 32 to 300 processors. Typical total times per time step were of the order of 15 CPU-minutes for grids of 100 Mpoints.

It is concluded that the code is ready for large-scale simulations of boundary layers of moderate Reynolds numbers, which are now under preparation. Two areas that need some work are the oscillations generated by the turbulent inflow, and an upper-wall boundary condition that limits the pressure oscillations in the free stream. In both cases the problem can be controlled by sufficiently large grids, but cheaper solutions are under consideration.

Acknowledgments

The development of this code, and the simulations used to test it, were supported in part by the CICYT grant DPI2003–03434, by the EU FP6 Wallturb Strep AST4-CT-2005-516008, by ITP S.A., and by Airbus Europe within the CAFEDA program. MPS, GH and SV were supported at various times by the EU FP5 Training and Mobility Network HPRN-CT-2002-00300, on Mixing and Stirring. Generous grants of computer time were received from the Barcelona Supercomputing Centre, and from the Magerit machine of the U. Politécnica de Madrid.

References

[1] K. Akselvoll and P. Moin. An efficient method for temporal integration of the Navier-Stokes equations in confined axisymmetric geometries. *J. Computat. Phys.*, 125:454–463, 1996.

[2] M. Alam and N.D. Sandham. Direct numerical simulation of short laminar separation bubbles with turbulent reattachment. *J. Fluid Mech.*, 410:1–28, 2000.

[3] S. Armfield and R. Street. The fractional-step method for the Navier-Stokes equations on staggered grids: The accuracy of three variations. *J. Computat. Phys.*, 153:660–665, 1999.

[4] L. Castillo and W. K. George. Similarity analysis for turbulent boundary layer with pressure gradient: Outer flow. *AIAA J.*, 39:41–47, 2001.

[5] K. A. Chauhan and H. M. Nagib. On the development of wall-bounded turbulent flows. In Y. Kaneda, editor, *IUTAM Symp. on Computat. Phys. and New Perspectives in Turbulence*, page to appear. Springer, 2006.

[6] J.A. Chorin. *Computational fluid mechanics: selected papers.* Academic Press, 1989.

[7] C. Y. Chu. *The fast Fourier transform on hypercube parallel computers.* PhD thesis, Dept. Computer Sci., Cornell Univ., 1987.

[8] R. Cortez D. L. Brown and M. L. Minion. Accurate projection methods for the incompressible Navier-Stokes equations. *J. Computat. Phys.*, 168:464–499, 2001.

[9] J.K. Dukowicz and A.S. Dvinsky. Approximate factorization as a high order splitting for the implicit incompressible flow equations. *J. Computat. Phys.*, 102:336–347, 1992.

[10] L. P. Erm and P. N. Joubert. Low-Reynolds-number turbulent boundary layers. *J. Fluid Mech.*, 230:1–44, 1991.

[11] A. Ferrante and S. E. Elgobashi. A robust method for generating inflow conditions for direct simulations of spatially-developing turbulent boundary layers. *J. Computat. Phys.*, 198:372–387, 2004.

[12] P.F. Fischer and A.T. Patera. Parallel simulation of viscous incompressible flows. *Ann. Rev. Fluid Mech.*, 26:483–527, 1994.

[13] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8:2182–2187, 1965.

[14] E. Jackson, Z.-S. She, and S.A. Orszag. A case study in parallel computing: I. Homogeneous turbulence on a hypercube. *J. Sci. Comput.*, 6:27–46, 1991.

[15] J. Kim and P. Moin. Application of a fractional-step method to incompressible Navier-Stokes equations. *J. Computat. Phys.*, 59:308–323, 1985.

[16] J. Kim, P. Moin, and R. D. Moser. Turbulence statistics in fully developed channel flow at low Reynolds number. *J. Fluid Mech.*, 177:133–166, 1987.

[17] W. Y. Kwok, R. D. Moser, and J. Jiménez. A critical evaluation of the resolution properties of B-spline and compact finite difference methods. *J. Computat. Phys.*, 174:510–551, 2001.

[18] H. Le and P. Moin. *Direct numerical simulation of turbulent flow over a backwards-facing step.* PhD thesis, Thermosciences Div., Dept. of Mech. Engng., Stanford Univ., 1994.

[19] K. Liu and R. H. Pletcher. Inflow conditions for the large-eddy simulation of turbulent boundary layers: A dynamic procedure. *J. Computat. Phys.*, 219:1–6, 2006.

[20] T. S. Lund, X. Wu, and K.D. Squires. Generation of turbulent inflow data for spatially-developing boundary layer simulations. *J. Computat. Phys.*, 140:233–258, 1998.

[21] Y. Maciel, K. S. Rossignol, and J. Lemay. Self-similarity in the outer region of adverse-pressure-gradient turbulent boundary layers. *AIAA J.*, 44:2450–2464, 2006.

[22] H. L. Meitz and H. F. Fasel. A compact-difference scheme for the Navier–Stokes equations in vorticity-velocity formulation. *J. Computat. Phys.*, 157:371–403, 2000.

[23] J. J. Modi. *Parallel algorithms for matrix computations.* Clarendon Press, Oxford, 1988.

[24] Y. Na and P. Moin. *Direct numerical simulation of turbulent boundary layers with adverse pressure gradient and separation.* PhD thesis, Thermosci. Div., Dept. Mech. Engng., Stanford Univ., 1996.

[25] S. Nagarajan, S.K. Lele, and J.H. Ferziger. A robust high-order compact method for large eddy simulation. *J. Computat. Phys.*, 191:329–419, 2003.

[26] P. Orlandi. *Fluid Flow Phenomena.* Kluwer, 2000.

[27] R. B. Pelz. The parallel pseudospectral method. *J. Computat. Phys.*, 92:296–312, 1991.

[28] J. B. Perot. An analysis of the fractional step method. *J. Computat. Phys.*, 108:51–58, 1993.

[29] L. P. Purtell, P. S. Klebanoff, and F. T. Buckley. Turbulent boundary layers at low Reynolds numbers. *Phys. Fluids*, 24:802–811, 1981.

[30] U. Rist and H.F. Fasel. Direct numerical simulation of controlled transition in a flat-plate boundary layer. *J. Fluid Mech.*, 298:211–248, 1995.

[31] M. P. Simens. *Simulation and control of transitional separation bubbles in low-pressure turbines.* PhD thesis, School of Aeronautics, U. Politécnica Madrid, 2007.

[32] M. P. Simens and J. Jiménez. Control of large two-dimensional laminar separation bubbles by shear-layer instabilities. In H.I. Andersson and P. A. Krogstad, editors, *Advances in turbulence X*, pages 725–728. CIMNE, 2004.

[33] M. P. Simens and J. Jiménez. Control of separation bubbles by shear-layer instabilities. In *Proc. Am. Phys. Soc. Ann. Meeting (Fluid Mech.)*, pages AG–2, 2004.

[34] M. P. Simens and J. Jiménez. Alternatives to Kelvin–Helmholtz instabilities to control separation bubbles. In *Power for Land, Sea and Air*, Paper GT2006-90670. ASME, 2006.

[35] M. P. Simens and J. Jiménez. Control of laminar separation bubbles far from the Kelvin–Helmholtz regime. In *Int. Symp. in honour of I. Wygnanski, on flow control, turbulence and VSTOL*, 2006.

[36] M. P. Simens and J. Jiménez. High amplitude forcing of boundary layers to control separation. In J. P. Bonnet and J. F. Morrison, editors, *IUTAM Symp. Flow Control and MEMS*, to appear. Kluwer, 2006.

[37] P. E. Skare and P. A. Krogstad. A turbulent equilibrium boundary layer near separation. *J. Fluid Mech.*, 277:1–21, 1994.

[38] M. Skote and D.S. Henningson. Direct numerical simulation of a separated turbulent boundary layer. *J. Fluid Mech.*, 471:107–136, 2002.

[39] S. Song and J. K. Eaton. Reynolds number effects on a turbulent boundary layer with separation, reattachment, and recovery. *Exp. Fluids*, 36:246–258, 2004.

[40] P. R. Spalart. Direct simulation of a turbulent boundary layer up to $Re_\theta = 1410$. *J. Fluid Mech.*, 187:61–98, 1988.

[41] P. R. Spalart, R. D. Moser, and M. M. Rogers. Spectral method for the Navier–Stokes equations with one infinite and two periodic dimensions. *J. Comput. Phys.*, 96:297–324, 1991.

[42] P.R. Spalart and J.H. Watmuff. Experimental and numerical study of a turbulent boundary layer with pressure gradients. *J. Fluid Mech.*, 249:337–371, 1993.

[43] A. A. Wray and R. S. Rogallo. Simulation of turbulence on the Intel Gamma and Delta. Tech. Memo unpublished, NASA, 1992.

[44] X. Wu, R. G. Jacobs, J. C. R. Hunt, and P.A. Durbin. Simulation of boundary layer transition induced by periodically passing wakes. *J. Fluid Mech.*, 398:109–153, 1999.

[45] M. V. Zagarola and A. J. Smits. Mean-flow scaling of turbulent pipe flow. *J. Fluid Mech.*, 373:33–79, 1998.

A Approximation errors

Consider the spectral response of the finite difference schemes used in the code, and assume that the grid is uniform and, when required, isotropic in two dimensions. The dispersion relation of the staggered, compact, fourth-order first derivative, when applied to $f_j = \exp(\mathrm{i}\widetilde{k}j)$, is

$$h f_j'/f_j = F_4^{(1)} = \mathrm{i}\,\frac{4\cos(\widetilde{k}/2)}{3+\cos(\widetilde{k})}\,\frac{24\sin(\widetilde{k}/2)}{11+\cos(\widetilde{k})}, \tag{56}$$

where the second factor corresponds to the derivative itself, and the first one to the required interpolation of the staggered variables. This may be compared with the standard second-order formula, for which

$$F_2^{(1)} = \mathrm{i}\sin(\widetilde{k}). \tag{57}$$

The performance of these formulas can be quantified by comparing their relative errors

$$\epsilon^{(1)} = \frac{|F - \mathrm{i}\widetilde{k}|}{|\widetilde{k}|}. \tag{58}$$

Examining the performance of the pressure-correction operation cannot be done in one dimension, where continuity is degenerate. Consider a two-dimensional equilateral Cartesian grid, and define a wave vector with components $\widetilde{k}_1$ and $\widetilde{k}_2$. If we define F_{Gi} and F_{Di} as the dispersion relations of the numerical approximations to the gradient and divergence operators along the coordinate i, the projection operation

$$\vec{f} = \vec{f}^{*} - \nabla\,\nabla^{-2}\left(\nabla\cdot\vec{f}^{*}\right), \tag{59}$$

has the numerical counterpart

$$\begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} f_1^* \\ f_2^* \end{pmatrix} - \widehat{P}\begin{pmatrix} f_1^* \\ f_2^* \end{pmatrix}, \tag{60}$$

where

$$\widehat{P} = \frac{1}{F_{G1}F_{D1} + F_{G2}F_{D2}} \begin{pmatrix} F_{G1}F_{D1} & F_{G1}F_{D2} \\ F_{G2}F_{D1} & F_{G2}F_{D2} \end{pmatrix}. \tag{61}$$

The analytic equivalent is

$$P = \frac{1}{\widetilde{k}_1\widetilde{k}_1 + \widetilde{k}_2\widetilde{k}_2} \begin{pmatrix} \widetilde{k}_1\widetilde{k}_1 & \widetilde{k}_1\widetilde{k}_2 \\ \widetilde{k}_2\widetilde{k}_1 & \widetilde{k}_2\widetilde{k}_2 \end{pmatrix}. \tag{62}$$

In the staggered second-order scheme, both F_D and F_G are $2\mathrm{i}\,\sin(\widetilde{k}/2)$, but the properties mentioned below do not depend on this equality.

It is easily shown that the eigenvalues of both (61) and (62) are 0 and 1. Their L_2 norms (the spectral radii) are therefore unity. This shows that, except for boundary effects that are not contemplated in the spectral analysis, the pressure projection step can never increase the L_2 norm of the flow velocity. The kinetic energy can therefore at most be conserved, and it usually decreases. This step is unconditionally stable.

The operations P and $\widehat{P}$ are both projections, although on slightly different planes. A rigorous estimate of the relative error is impossible without some knowledge of the properties of the initial point f^*, but a useful measure is

$$\epsilon_P = \|\widehat{P} - P\|_2, \tag{63}$$

where $\| \, \|_2$ is the L_2 norm, since we know that $\|P\|_2 = 1$.

Figure 2 in section 3.2 plots $\epsilon_2^{(1)}$ and $\epsilon_4^{(1)}$ versus $\widetilde{k}$, and ϵ_P versus $\|\widetilde{k}\|_2$.