JOURNAL OF AIRCRAFT
Vol. 42, No. 1, January–February 2005

Implementation and Validation of the Spalart–Allmaras Turbulence Model in Parallel Environment

Stéphane Séror,* Theodor Rubin,† and Sergey Peigin‡
Israel Aircraft Industries, 70100 Ben-Gurion Airport, Israel
and
Boris Epstein§
The Academic College of Tel-Aviv Yaffo, 60160 Tel-Aviv, Israel

The paper describes the evolution of a Navier–Stokes code developed in the Israel Aircraft Industries and its application to complex turbulent high-Reynolds flows. The paper focuses on the following issues: 1) implementation and validation of the Spalart–Allmaras turbulence model, 2) parallelization of the computational framework, and 3) application to the design of the wing-body fairing of a generic business jet.The improved capability of the code allowed its use for large-scale flow simulations on a daily basis.

Introduction

A NAVIER–STOKES code, NES (Navier–Stokes Euler System), has been developed over the past 10 years at the CFD Group of Israel Aircraft Industries (IAI).[1–3] Originally a monoblock code, the code has evolved towards a robust multiblock code.[4,5] The code uses high-order essentially nonoscillating (ENO) scheme to compute transonic and supersonic flows including shocks and vortices, with a very low level of numerical dissipation. The need for such a code arose from the need for high-accuracy solutions of complex viscous flow fields about arbitrary aircraft or unmanned aerial vehicle missile configurations for a wide range of flight conditions.

Turbulent flows have hitherto been modeled by means of the Baldwin–Lomax model.[6] The special requirements of that model complicate the design of a multiblock topology with point-to-point connections at adjacent faces. Indeed, the Baldwin–Lomax model relies on surveying the velocity or vorticity profile on a smooth grid line, roughly orthogonal to the surface, thus being nonlocal, which significantly increases the effort required for grid generation, and limits the flexibility in terms of grid topology. This drawback, in addition to the poor accuracy expected when a boundary-layer model such as Baldwin–Lomax is applied to a massively separated flow, underlines the importance of having a more advanced turbulence model, which is also less grid dependent. This has justified the inclusion last year in NES[7,8] of a one-equation model, that is, the Spalart–Allmaras (SA) model.[9] This is a "local" model, widely used today in the computational-fluid-dynamics (CFD) community because of its proven superior prediction compared with the Baldwin–Lomax model and its robustness compared with two-equation models.

In the first section we start presenting the original model and discuss the problem of obtaining a conservation form of the model for compressible flows. Then we present the way chosen to implement it in the multiblock/multiface/multigrid program NES. Main issues encountered, essentially linked to the difficulties of obtaining a positive solution for the turbulent viscosity in the multigrid cycle, are briefly addressed.

The code has been parallelized in NES[10] to allow short turnaround time. The main highlights of this work are presented. The code is now routinely used at IAI for determining the drag of complex aerodynamic configurations.

To validate the model implementation, we compare our results to the ones given in the literature by Spalart–Allmaras. This is done considering the strong shock-wave boundary-layer interaction test case about the RAE2822 sharp trailing-edge airfoil at transonic conditions. Finally, we briefly illustrate on two complex three-dimensional geometries, a wing-pylon-pod and a wing-body ARA-M100, some interesting features of the model in an engineering context.

In the second section of this paper, a quasi-automatic design loop process is proposed aiming at improving the shape of the fairing of a generic business jet based on the drag analysis using the NES code in combination with the Euler solver MGAERO.[11]

The conjunction of the two codes has led to the definition of an acceptable shape for the fillet at the wing-body junction of a generic business jet at transonic cruise conditions. The strategy was to diminish the shock intensity near the wing-body junction by modifying the shape of the fillet in an iterative process using the Euler code. Then Navier–Stokes computations were realized to assess the gains in term of drag using the SA turbulence model. In a second step, the goal was to optimize the shape in term of drag using the NES code only. Because the bottleneck was the turnaround time for the mesh preparation of each new shape, programs have been developed performing automatically the mesh deformation and remeshing. Even when the changes in the mesh were just localized in the fillet area, building the new mesh manually was done in one week. Now, using the new procedure, it takes a few minutes to generate the new mesh. In that way, a significant number of different geometries could be analyzed by NES computations in the design loop. The results of the design are presented.

Implementation and Validation

Numerical Algorithm of the Code NES

In the Navier–Stokes code NES,[1–3] the numerical discretization for the convective part of the equation is based on the ENO scheme[12] the viscous operator uses classical central differences, and the source terms are discretized in a straightforward fashion. The time integration is based on the three-stage Runge–Kutta time-stepping scheme with the Navier–Stokes local time step.

Presented as Paper 2003-3404 at the AIAA 21st Applied Aerodynamic Conference, Orlando, FL, 22–26 June 2003; received 10 September 2003; revision received 23 February 2004; accepted for publication 23 February 2004.

*Aerospace Research Engineer, Engineering Division, Aerodynamic Department, Group CFD; sseror@iai.co.il. Member AIAA.

†Head, CFD Group, Engineering Division, Aerodynamic Department.

‡Research Scientist, Engineering Division, Aerodynamic Department, Group CFD.

§Consultant, Computer Science Department.

Turbulence Modeling

Turbulent flows have hitherto been modeled in NES by means of the Baldwin–Lomax model. The special requirements of that model complicate the design of a multiblock topology with point-to-point connections at adjacent faces. Indeed, the Baldwin–Lomax model relies on surveying the velocity or vorticity profile on a smooth grid line, roughly orthogonal to the surface, thus being non-local, which significantly increases the effort required for grid generation and limits the flexibility in terms of grid topology. This can be seen in Fig. 1 at the junction between the pod and the pylon of a generic wing + pod + pylon configuration. The visualization in Fig. 2 of the lack of orthogonality between the columns and the surface reveals some areas with high skewness. This drawback, in addition to the poor accuracy expected when a boundary-layer model such as Baldwin–Lomax is applied to a massively separated flow, underlines the importance of having a more advanced turbulence model, which is furthermore less grid dependent. This justifies the inclusion of a one-equation model such as that of Spalart-Allmaras,[9] which is a local model widely used today in the CFD community. Moreover one must stress the robustness of the SA model compared with two-equation models. The methodology adopted in implementing this model and its parallelization in the multiblock/multiface/multigrid program NES is summarized in the following section.

Finite Volume Formulation of the Model

There have been a number of one-equation turbulence models developed, which use a transport equation to solve for the eddy viscosity directly. The SA model belongs to the family of eddy-viscosity models. This family of models is based on the assumption that the Reynolds-stress tensor $\overline{-\rho u_i u_j}$ is related to the mean strain rate through an apparent turbulent viscosity called eddy viscosity ν_t, $\overline{u_i u_j} = \nu_t(\partial \bar{U}_i/\partial x_j + \partial \bar{U}_j/\partial x_i)$. In the SA model, the eddy viscosity is computed through a partial differential equation. In particular the eddy viscosity is computed by an intermediate variable $\tilde{\nu}$ through the relation $\nu_t = \tilde{\nu} f_{v_1}(\chi)$, where χ is the ratio $\chi = \tilde{\nu}/\nu$ and f_{v_1} is a damping function. The intermediate variable $\tilde{\nu}$ is computed by solving a partial differential equation that can be written in compact form as

$$\frac{\mathrm{D}\tilde{\nu}}{\mathrm{D}t} = P(S, \tilde{\nu}, d) - D(\tilde{\nu}, d) + \frac{1}{\sigma}\left\{\nabla \cdot [(\nu + \tilde{\nu})\nabla\tilde{\nu}] + c_{b_2}(\nabla\tilde{\nu})^2\right\}$$

where $P(S, \tilde{\nu}, d) = c_{b_1}\tilde{S}\tilde{\nu}$ is the production term of turbulence and $D(\tilde{\nu}, d) = c_{w_1} f_w(\tilde{\nu}/d)^2$ the destruction term. The difference between these terms represents the balance between the production and destruction source terms of turbulence, resulting from turbulence energy exchange from the mean motion to the fluctuation. These terms depend primarily on the variable $S, \tilde{\nu}, d$. S denotes the vorticity magnitude, and d the distance to the closest wall. The last term in the right-hand side is a diffusion term in which σ and c_{b_2} denote the turbulent Prandtl number and a calibration constant, respectively. More details and an extensive discussion about the SA model are reported in Ref. 9.

The SA turbulent transport equation is given in the substantial differential form, whereas the equations are required in conservation form in NES. The first task has been to rewrite the SA equation in a conservation form. Combining the preceding equation and the continuity equation, the new equation in conservation form can be written for the variable $\tilde{\mu} = \rho\tilde{\nu}$:

$$\frac{\partial\tilde{\mu}}{\partial t} + \nabla \cdot (\tilde{\mu}\boldsymbol{V}) = P(S, \tilde{\mu}, d) - D(\tilde{\mu}, d) + \frac{1}{\sigma}\left\{\nabla \cdot [(\mu + \tilde{\mu})\nabla\tilde{\nu}] + c_{b_2}\rho(\nabla\tilde{\nu})^2\right\}$$

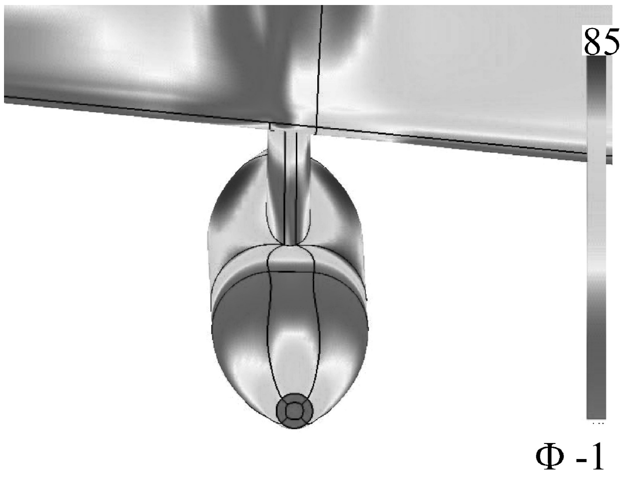


Fig. 2 Preprocessing—control of mesh orthogonality at the skin.

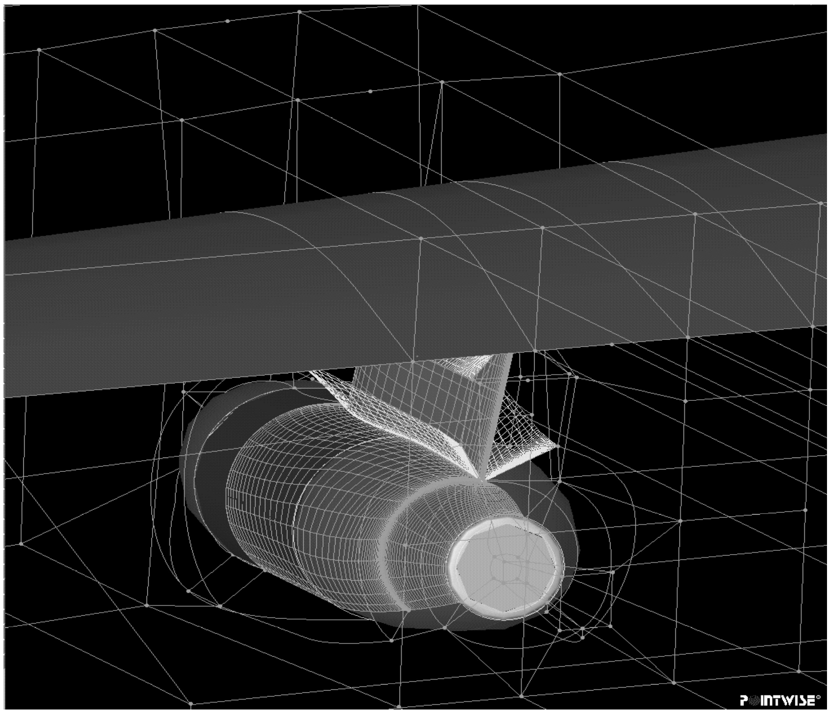


Fig. 1 GRIDGEN grid generation.

Because the second term in the diffusion term does not easily lend itself to a stable scheme,[9] it is preferred to modify the form of the diffusion term in the SA model, to obtain the form:

$$\frac{\partial \tilde{\mu}}{\partial t} + \nabla \cdot (\tilde{\mu} \boldsymbol{V}) = P(S, \tilde{\mu}, d) - D(\tilde{\mu}, d)$$

$$+ \frac{1}{\sigma}\left\{(1 + c_{b_2})\nabla \cdot [(\mu + \tilde{\mu})\nabla\tilde{\nu}] - c_{b_2}(\mu + \tilde{\mu})\nabla \cdot (\nabla\tilde{\nu})\right\}$$

where liberties have been taken with differentiation of the molecular viscosity as recommended by Spalart–Allmaras.[9]

Solution Procedure Implementation

The same defect correction multigrid method for solving the Navier–Stokes equation is used to solve the SA model. The adopted strategy is to solve the SA equation in a weakly coupled manner. Two options are possible (private communication with Dr. Spalart and Dr. Allmaras):

1) Freeze the SA solution, and perform a complete multigrid cycle for the Navier–Stokes equations. Then freeze the velocity, and do a complete multigrid cycle for the SA equation. This would involve building the same sort of multigrid code (e.g., restriction and prolongation operators) already existing, but for a single partial differential equation.

2) Incorporate both the Navier–Stokes and SA numerical procedure into a given multigrid cycle, but perform decoupled relaxation on each grid level. One advantage of this approach is that one does not need to duplicate the multigrid cycling code. All that is needed is to add an additional unknown/residual to the restriction and prolongation operators.

The second strategy has been chosen because this requires a smaller change in the code algorithm. The decision is based on code structure and modularity. Moreover it has been demonstrated recently by Allmaras[13] that this uncoupled approach on each level has identical convergence to tightly coupled relaxation. The solution procedure for a flow involves the five variables (ρ, ρu, ρv, ρw, ρe) for the Navier–Stokes equations and one $\tilde{\mu}$ for the SA turbulence closure model. The iterative procedure is represented by the flowchart in Fig. 3.

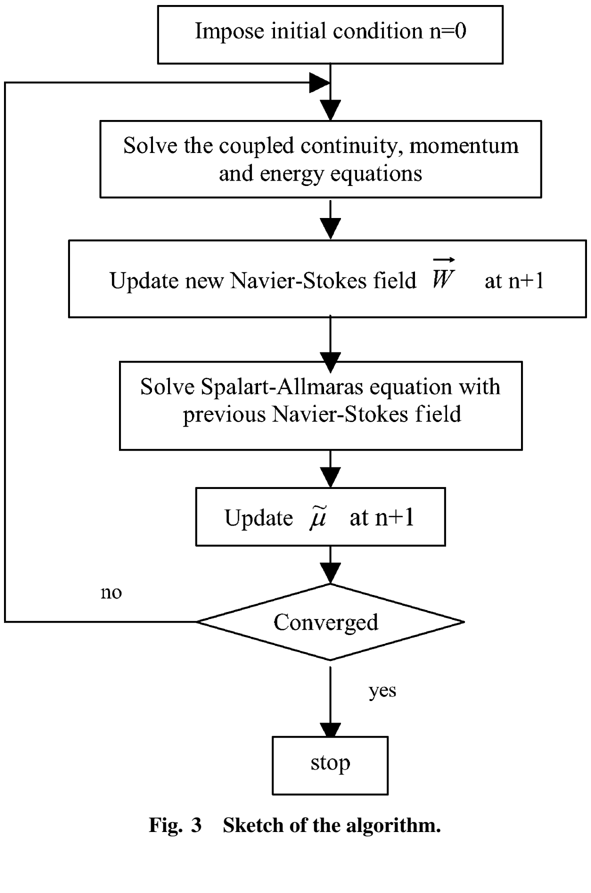

Impose initial condition n=0
Solve the coupled continuity, momentum and energy equations
Update new Navier-Stokes field $\overrightarrow{W}$ at n+1
Solve Spalart-Allmaras equation with previous Navier-Stokes field
Update $\tilde{\mu}$ at n+1
Converged
no
yes
stop

Fig. 3 Sketch of the algorithm.

As part of the issues encountered, the convergence of the SA equation to the steady state was very difficult to obtain at third order (ENO = 3). Therefore, it has been assessed that it is possible to provide good aerodynamic solution even when solving the SA convective part at first order. It seems in view of the following results that the accuracy in obtaining a converged solution for the eddy viscosity has little effect on the aerodynamic field. Another issue concerns the convergence regularity through multigrid cycle. In a standard full approximation multigrid scheme for convergence to the steady state, one has to compute the coarse-grid H to fine-grid h correction interpolation:

$$\tilde{\mu}_h = \tilde{\mu}_h^0 + I_H^h\left(\tilde{\mu}_H - I_h^H \tilde{\mu}_h^0\right)$$

It has been found that this could lead to a negative update solution for the eddy viscosity after correction even if the modified version of the SA model proposed by Allmaras[13] to admit small negative turbulence solution $\tilde{\mu}$ is used. The idea of Allmaras was to alter the nonphysical transient behavior as an aid to achieve steady-state solutions. In our multigrid code this was not enough, and consequently the convergence was slow and difficult. A proposed solution, which we found better than using a cutoff at zero, or a simple underrelaxation, has been to scan the $\tilde{\mu}$ field after the coarse-to-fine correction interpolation. Each time a negative update was found it was replaced by the interpolation of the coarse solution to the fine grid, that is,

$$\tilde{\mu}_h = I_H^h(\tilde{\mu}_H)$$

The convergence was immediately faster, and the number of negative updates that could appear through the coarse-to-fine correction interpolation of $\tilde{\mu}$ progressively diminished to zero in a monotonic way during the multigrid cycles.

Parallel Implementation on PC's Cluster

The parallelization of NES is directly linked and totally dependent on the multiblock-multiface structure of the code.[4,5] Hence, the parallel implementation of the SA model followed the previous approach of the Baldwin–Lomax NES code.[14,15] The method is described in the following, prefaced by remarks regarding the code structure, which are pertinent to the parallelization issues.

Computations are performed on a sequence of grids, starting from a coarse grid and refining it successively. Structured blocks are united in multigrid levels, and a computation proceeds from the coarsest level to the finest. Multigrid cycles are performed on each current level, followed by interpolation of the solution to the next finer level. In the full approximation scheme, in the fine-to-coarse direction residuals are transferred, and new right-hand sides for the coarse level calculations are found.

In a multiblock approach, the global domain is divided into several smaller blocks for which i, j, k structures are more easily generated. The solution procedure is applied to each block in some prescribed sequence. In the present method, blocks are grouped into so-called grids (each grid can contain from one to several blocks), and each multigrid level consist of one or more grids. For the purpose of the numerical method, blocks themselves may be considered as being divided into one-dimensional segments. In the present algorithm, the block loop is placed deep within the algorithm, in order to allow a close simulation of a single-block calculation. At the top of the code architecture are routines controlling the multigrid strategy and thus the loops on levels. In the middle are routines controlling the block loops, and at the bottom are block-structured routines that perform operations on a given block.

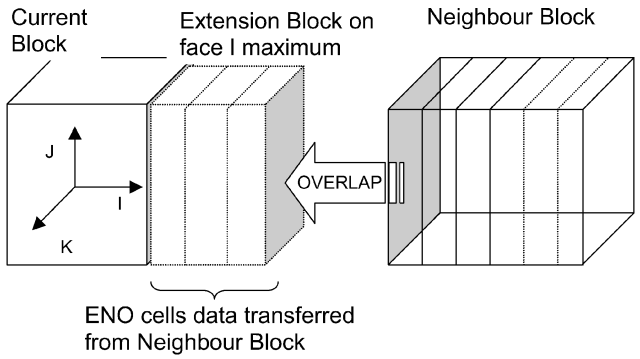
From a mathematical point of view and for the numerical scheme used here, there is no difference between a single-block and a multiblock solver. In particular, there is no difference in the implementation of the following block face boundary conditions: 1) SURFACE representing the configuration skin, 2) FARFIELD representing far-field condition, 3) SYMMETRY representing condition at the plane of symmetry, 4) INTERNAL representing the bounds of a local refinement covering only part of an existing block, and 5) INLET/OUTLET for inflow/outflow boundary conditions. However, there appears a difference because of the introduction of a block connectivity boundary condition (MERGE). The basic ideas underlying the present development are that each block should be able to internally identify the types of boundaries on its six sides (SURFACE, FARFIELD, SYMMETRY, MERGE, INLET, OUTLET). Moreover, in the present version of the multiblock solver it is assumed that grid lines are continuous across a block interface (exact point-to-point correspondence between neighbouring blocks), and each block face admits only one type of boundary condition. In addition, it is assumed that a block face might abut more than one neighbour block (multiface) in order to reduce the total number of blocks required when performing the domain splitting operation.

Therefore each computational block is provided with information of the types of boundaries on its six sides and is operated on using the ENO algorithm. The boundary conditions implementation in the multiblock solver has received special attention to obtain a high computational efficiency. Indeed, the implementation of the boundary conditions (including the block connectivity boundary condition) is uncoupled from the algorithm used to update the interior points. The concept of boundary arrays and ghost cells is used to transfer the boundary condition information to the algorithm for the interior points. On each face corresponding to a MERGE boundary condition, we build an extension block or boundary array having the necessary number of cells required by the order of the ENO method currently operated.[1–3] This extension block is filled by overlapping information from the neighbour block (see Fig. 4). The communication overhead caused by the data exchange among neighboring blocks is negligible as it uses no complicated data management such as connectivity lists. As a result, the code is highly suitable for efficient parallelization on an almost plug-in basis.

For each Runge–Kutta stage the block connectivity boundary condition is updated at the level of the grid, that is, after the completion of the loop on all blocks, whereas all other boundary conditions are updated for each Runge–Kutta stage at the level of the block itself. In this way, it is possible to treat each block in a multiblock simulation as if it were a single-block calculation (a multiblock calculation is seen as multiple single-block calculations).

The main idea of the parallelization of the NES code is that when two neighbors blocks are mapped into different processors the overlapping subroutine aiming at the exchange of information between the blocks to enable the flux computation at the common face is parallelized using a message-passing algorithm based on the parallel virtual machine (PVM) library. The separate multigrid algorithm of the SA code is parallelized using exactly the same basic ideas. The main part of the data transfer between the processors is associated with exchange of boundary data between neighboring blocks mapped into different processors. The good load balance of the parallel code NES is achieved through two essential characteristics. The first is the automatic splitting of the blocks before their mapping on the processors. This allows an approximately equal number of grid points per processor. As a consequence, it has been necessary to write an algorithm that enhances the existing automatic boundary condition determination algorithm, to the case where the common boundary of two adjacent blocks does not include a corner of these blocks (crossface condition). It was previously necessary to manually split such blocks during the mesh generation to avoid the crossface condition, especially when handling very large grids (more than about four million grid points). As a result, the maximum efficiency can be achieved by an optimal automatic splitting of the big blocks on the processors as it is shown in Fig. 5 for a typical run on 60 processors. The second reason that explains the high efficiency of the parallel algorithm in NES is the overlapped communication and computation concept. This means that the send procedures are independently executed in each blocks as soon as the data to be transferred are calculated. Such an approach reduces the losses caused by communication because the data are transferred during the load-balancing waiting time. Concerning more specifically the SA model, the computation of the closest distance to the wall, in the preprocessor of NES, has been parallelized. This computation is done once the blocks are mapped onto the processors that define the cluster. Once each processor has executed this task, it sends to the master the data in order to include the minimum distance parameter in the file outputted for a graphical check of the preprocessor stage.

The parallel version of the multiblock multiface turbulent Spalart–Allmaras Navier–Stokes version of NES is based on the PVM software package. The code is implemented on multiple instruction multiple data (MIMD) multiprocessors cluster of HP NetServer LP10000R 866 Mhz. Each node has two processors, 2 GB RAM memory, and full duplex 100 Mbps Ethernet interface. The cluster contained 106 processors managed by MOSIX software,[16] which enhances the LINUX kernel with cluster computing capabilities.


Current Block
Extension Block on face I maximum
Neighbour Block
J
I
K
OVERLAP
ENO cells data transferred from Neighbour Block
Fig. 4 Multiblock data management at merged faces.

Numerical Validation

The parallel implementation of the SA model has been validated on the two-dimensional RAE2822 supercritical airfoil at transonic flight conditions $Re = 6.5e^6$, $\alpha = 2.6$ deg, $M = 0.75$, the same test case as in Ref. 9, where the SA model was incorporated into the code FLO103 (Ref. 17). The fine grid possesses a minimum grid spacing normal to the wall of 10^{-6} m as in Ref. 9 Runs were performed on the medium (194 × 49) points and fine grid (387 × 97) points.

The main conclusions are as follows:

1) The SA solution convergence presents the same characteristics as the Baldwin–Lomax one (see. Fig. 6). It is emphasized that the reduction of the residuals by two to three orders of magnitude is characteristic of the ENO approach and is sufficient to provide convergent values of the aerodynamic coefficients in excellent agreement with the experiment.

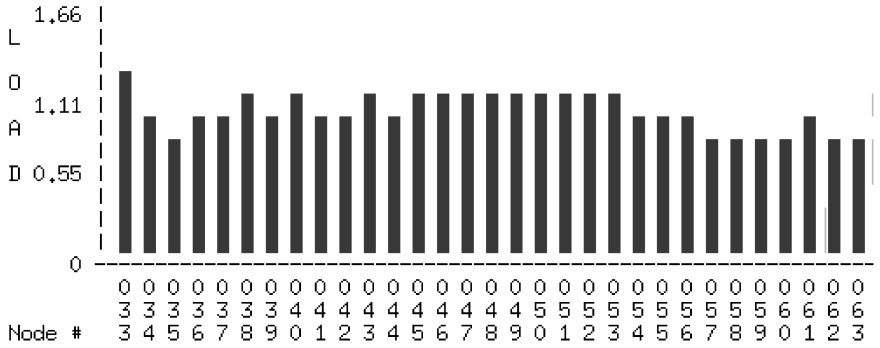


Fig. 5 Load balancing of the parallel algorithm.

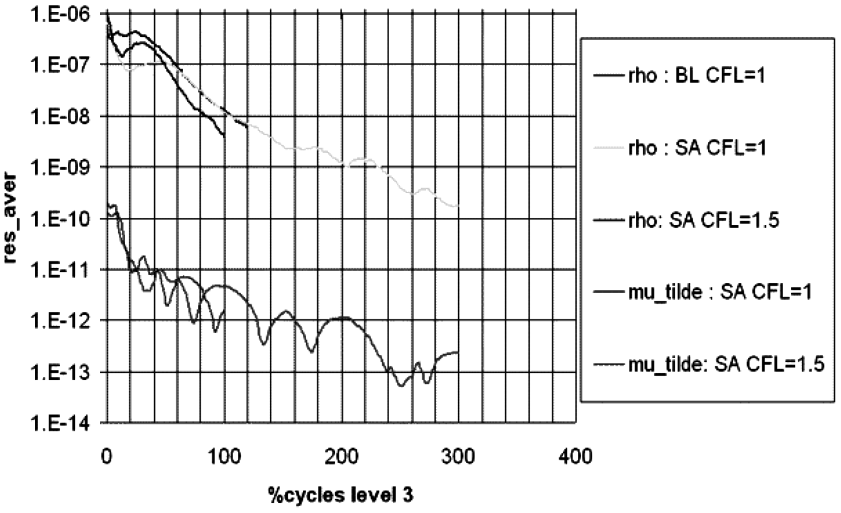


Fig. 6 RAE2822-SA vs BL-convergence.

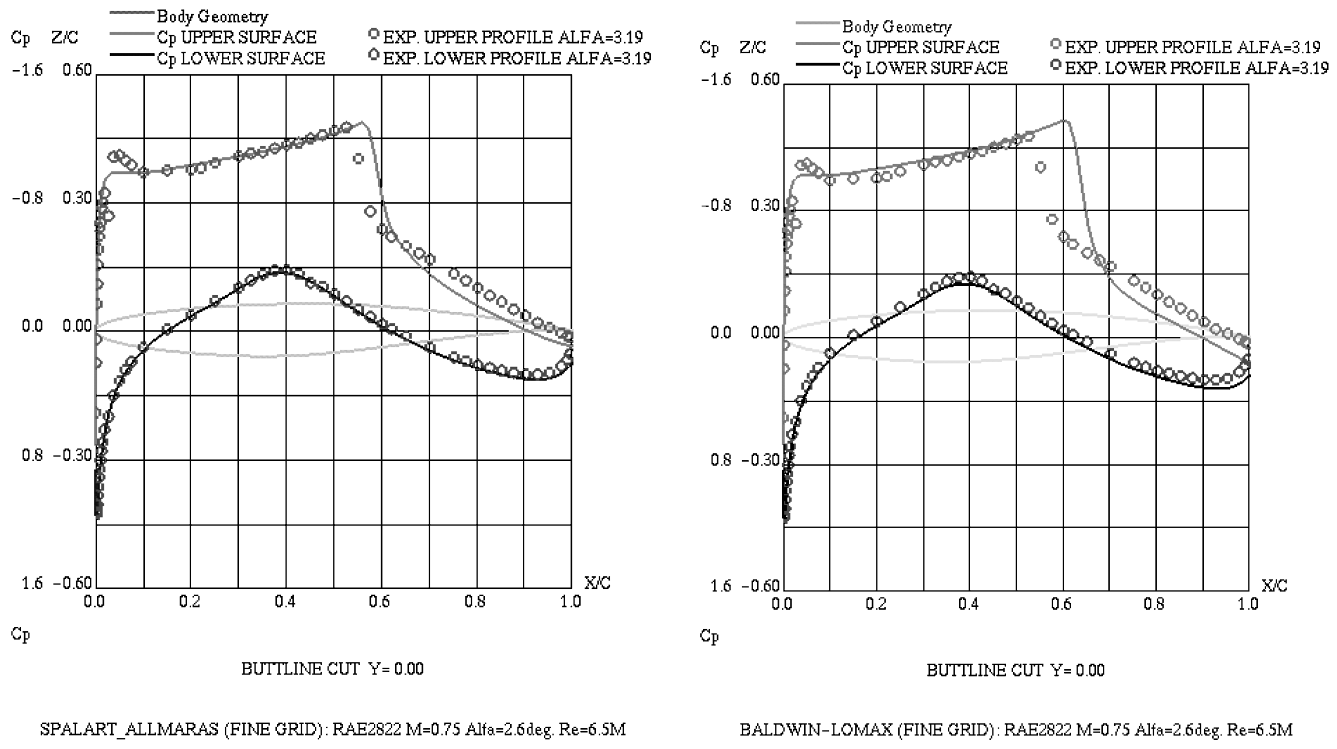


Fig. 7 RAE2822-SA vs BL-*Cp* profile.

2) The shock location is better predicted than with the previous Baldwin–Lomax model (see Fig. 7).

3) The turbulent eddy viscosity fields about the profile are quite different between the Spalart–Allmaras and Baldwin–Lomax (BL) models (see Fig. 8), especially in the wake area because the two models use different turbulent scale in that area.

4) A reattachment is obtained as in the experiment, as shown by the skin-friction coefficient (see Fig. 9).

5) The aerodynamic coefficient are very close to those obtained by Spalart and Allmaras in Ref. 9 (see Table 1). The slight difference might be caused by the fully turbulent calculation at fixed angle of attack with NES, whereas the result in Ref. 9 corresponds to a prescribed 3% transition at fixed $CL = 0.743$.

Industrial Applications

A brief overview is now given of three-dimensional capability for various cases of industrial interest that illustrate the ability of the code to handle complex aerodynamic configurations.

Table 1 Aerodynamic coefficients

Case	Cl	Cd	Cdf	Cm
Medium BL	0.824	275	54	−0.120
Fine BL	0.823	273	57	−0.118
Medium SA	0.741	236	54	−0.101
Fine SA	0.756	238	54	−0.103
Ref. 9	0.743	238	∂/m	−0.104
Exp. $\alpha = 3.19$ deg	0.743	242	∂/m	−0.106

Analysis of the results demonstrated a high level of parallel efficiency (speed up) of the algorithm. This enabled the reduction of the execution time for industrial computations employing three million of grid points from an estimated 20 days on the SGI ORIGIN 2000 machine (in the serial single-user mode) to about five hours on a 100-processor cluster. In general the parallel efficiency is above 90%, provided a good load balance is achieved through an automatic method of splitting large blocks prior to computation.[15]

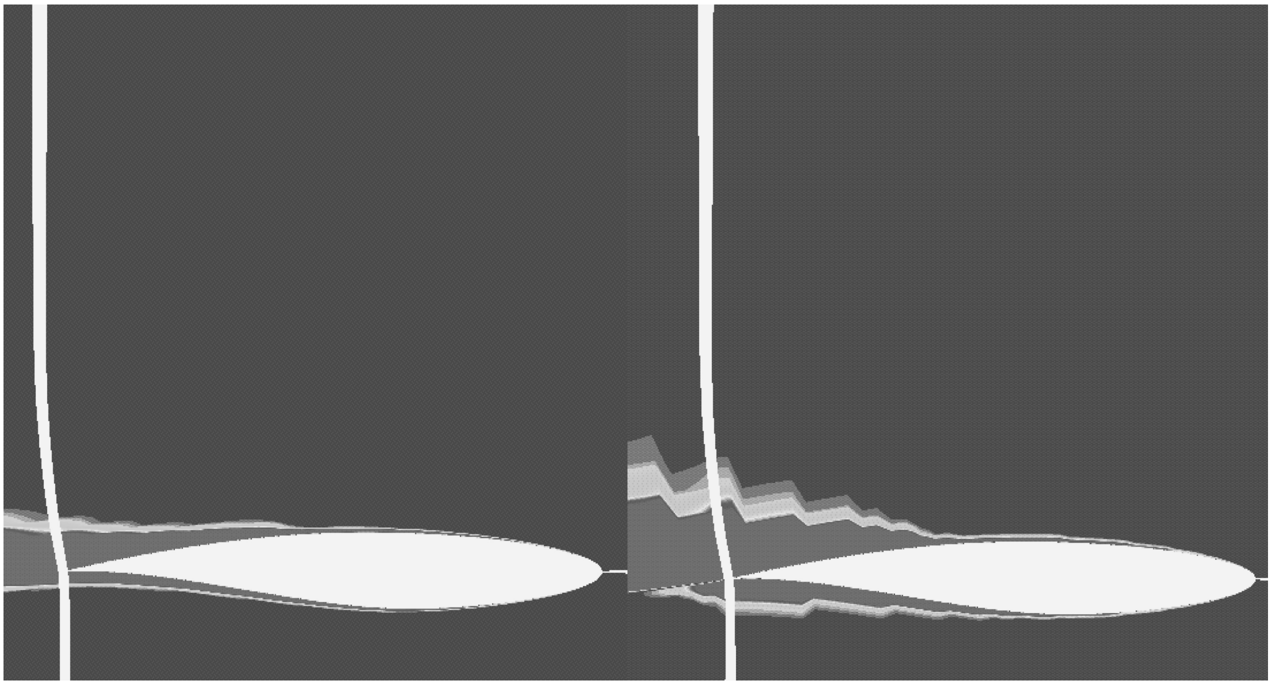


Fig. 8 RAE2822-SA (left) vs BL (right)-eddy viscosity field.

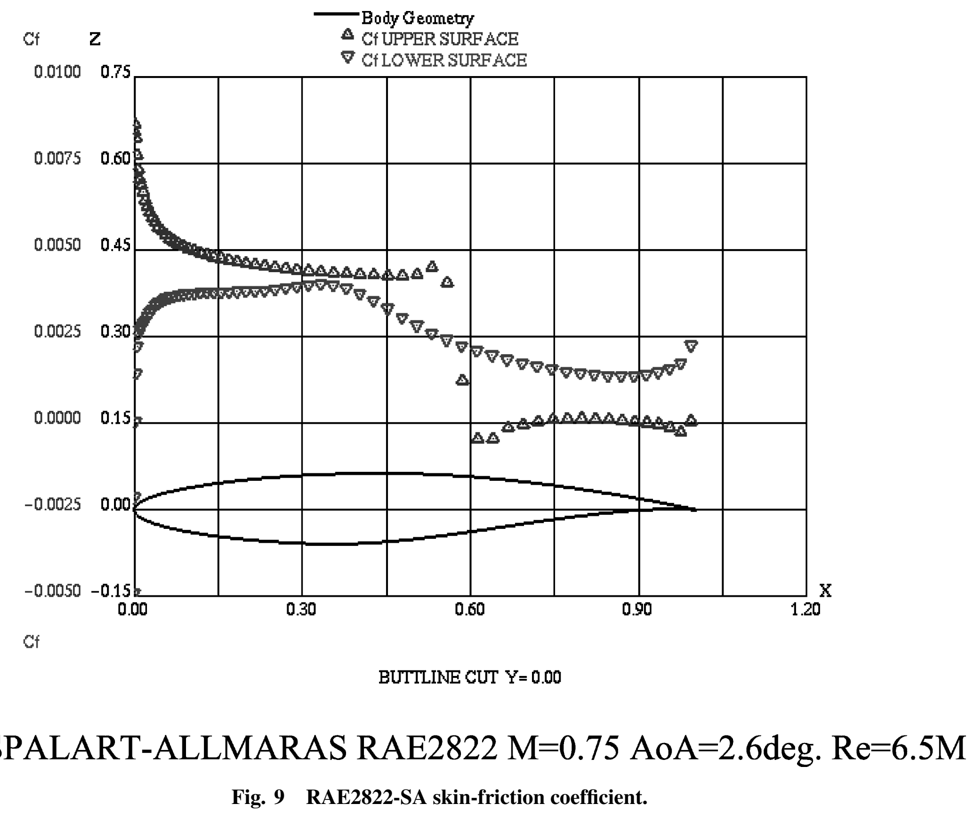


Fig. 9 RAE2822-SA skin-friction coefficient.

Generic Wing Pod Pylon and Wing Body

Figure 10 shows a comparison between the two turbulence models for the skin-friction distribution on the wing-pylon-pod case. This reveals as mentioned earlier the strong dependency of the Baldwin–Lomax model to the skewness of the grid, justifying the need to have a more local model as the Spalart–Allmaras one. The Spalart–Allmaras and Baldwin–Lomax models have been compared on a three-dimensional generic wing-body configuration, ARA-NASA M100, at Mach $= 0.8027$, $\alpha = 2.873$. Figure 11 shows the CL vs CD polar. The comparison between the Spalart–Allmaras and Baldwin–Lomax models on a medium grid shows the general improvement given by the SA model. In Figs. 12 and 13 a comparison is shown between the Spalart–Allmaras and Baldwin–Lomax models of turbulence for two different flight points. The first corresponds to $M = 0.8027$, $\alpha = 2.873$ and the second to $M = 0.8792$, $\alpha = 0.467$. This latter case is more challenging because the shock boundary-layer interaction is stronger. For that point it is seen that the Spalart–Allmaras model is superior with regard to the shock position. Concerning the first point, the NES results are similar to those of Ref. 18.

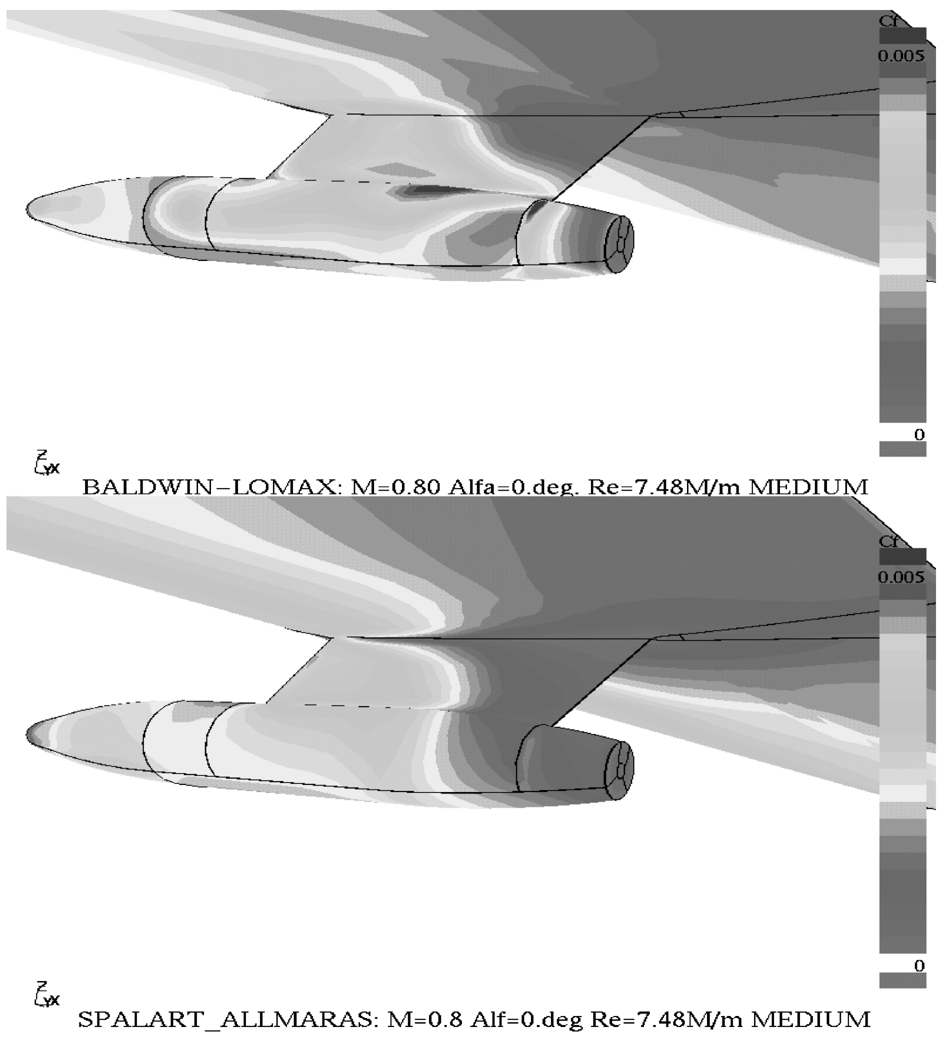

Cf
0.005
0
BALDWIN–LOMAX: M=0.80 Alfa=0.deg. Re=7.48M/m MEDIUM
Cf
0.005
0
SPALART_ALLMARAS: M=0.8 Alf=0.deg Re=7.48M/m MEDIUM

Fig. 10 Comparison Spalart–Allmaras vs Baldwin–Lomax—*Cf* on wing pod pylon.

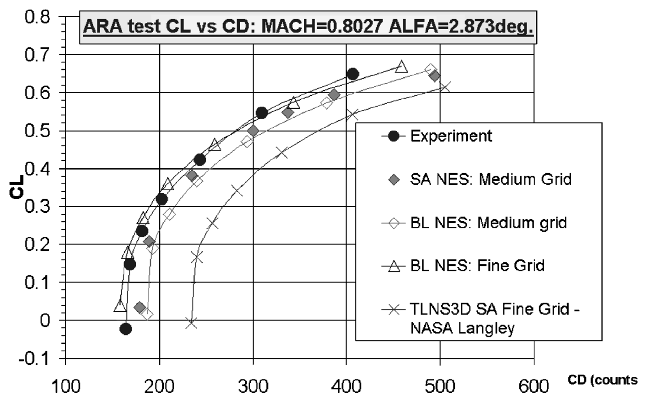


Fig. 11 Polar curve-lift coefficient CL vs drag coefficient CD-ARA M100 M = 0.8027, Re = 13.1 M.

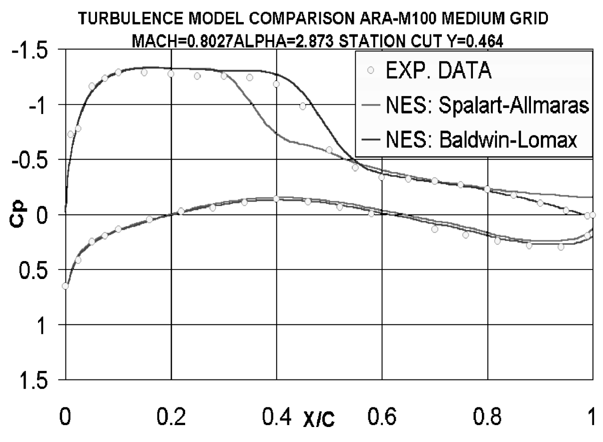


Fig. 12 SA vs BL at crank station-M = 0.8027, α = 2.873.

Wing-Body Fairing Design

The fillet between the upper wing surface and the fuselage of a generic business jet was modified in order to reduce drag at transonic cruise.

Design Process

The design concept centered on diminishing the shock, which occurred at about $\frac{2}{3}$ of the wing-root chord on the nominal configuration, as shown in Fig. 14. The CFD tools used were 1) MGAERO, an Euler solver using Cartesian grids, and 2) NES, the Navier–Stokes solver just described. Initially, the design effort concentrated on MGAERO, using as a criterion the reduction of the local Mach number at the wing-root shock (Fig. 14). It was verified that the shock strength was only slightly affected by the presence of the aft-mounted nacelle, justifying the calculation of a body-wing-winglet-only configuration for the NES runs. At this stage, only a limited number of NES runs were conducted, in order to obtain quantitative drag evaluations of the more promising fillet designs.

At a later stage, the design relied totally on NES runs, driving the modification cycles on the basis of drag evaluation directly. New fillet shapes were chosen on a trial-and-error basis, within the framework of the deformation modes described next. Some of the

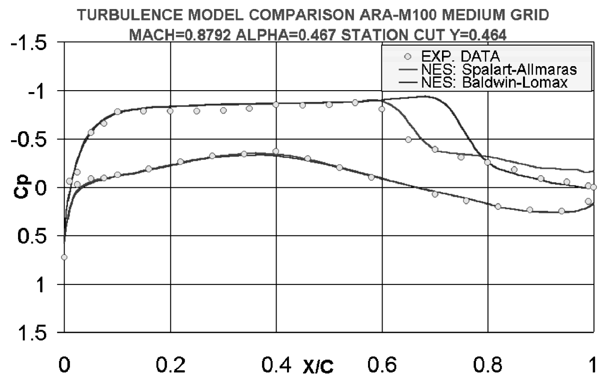


Fig. 13 SA vs BL at crank station-$M = 0.8792$, $\alpha = 0.467$.

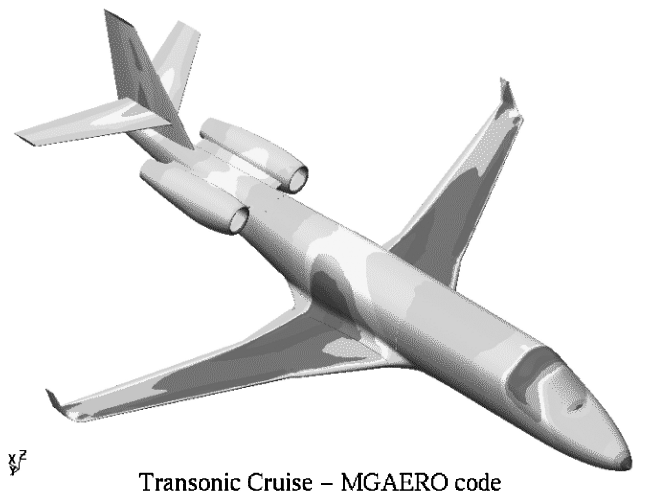


Fig. 14 MGAERO CFD computation at transonic design flight point.

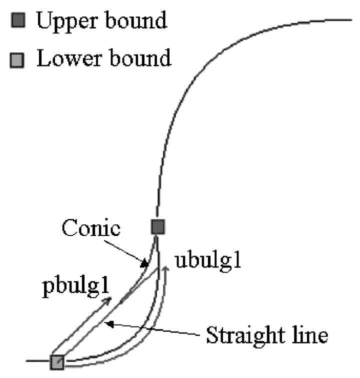


Fig. 15a BULGE1 mode.

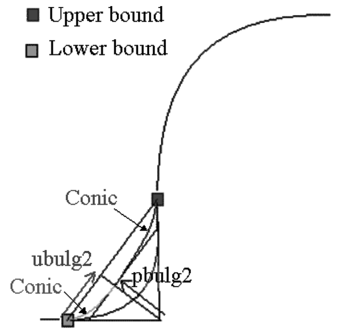


Fig. 15b BULGE2 mode.

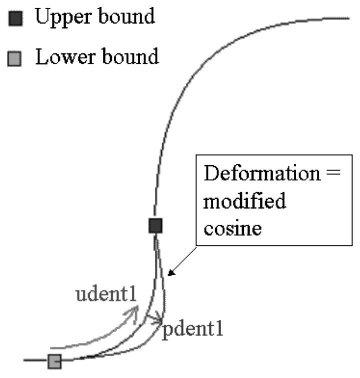


Fig. 15c DENT1 mode.

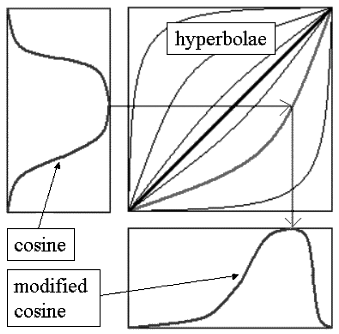


Fig. 15d Modified cosine.

initial guesses led to very unfavorable results, but within about a week some more promising modifications began to appear. Following up these variants with sensitivity checks for one parameter at a time, a significantly improved configuration was found within two more weeks.

Parameterization of the Fillet Modification

Defining the coordinate system as in Fig. 14, the modifications were based on altering sections normal to the fuselage axis, that is, at X constant stations.

The bounds of the modified region were specified as follows: 1) X station, representing the upstream bound; 2) X station, representing the downstream bound; 3) curve projected on configuration skin, representing the lower bound (on the wing-root upper surface); and 4) curve projected on configuration skin, representing the upper bound (on the side of the fuselage).

Various modes of deformation were constructed for modifying individual X constant sections (see Figs. 15a–15d: 1) BULGE1, a straight line stemming from lower bound, joining a conic to meet skin surface tangentially at upper bound (Fig. 15a); 2) BULGE2, two conics joining tangentially and meeting skin surfaces tangentially at lower and upper bounds (Fig. 15b); and 3) DENT1, normal deformation (Fig. 15c) given by a modified cosine function (Fig. 15d) tailing off smoothly to zero at the lower and upper bounds.

Parameters given for each deformation are noted: 1) for which X station the section received the maximum effect of the deformation, and 2) the amplitude of this maximum deformation.

At other sections between the upstream and downstream bounds, a blending of original and deformed sections is formed, so that the deformed geometry merges smoothly into the original geometry at these bounds. The degree of blending is controlled by a modified cosine function (Fig. 15d).

An additional parameter is noted: 3) the location at which the maximum deformation on a section is achieved.

Summarizing, if the preceding four boundaries are regarded as describing a roughly rectangular region of modification for a particular deformation mode, then the location of this mode's maximum change is defined lengthwise and heightwise by parameters 1) and 3) respectively, with amplitude defined by 2).

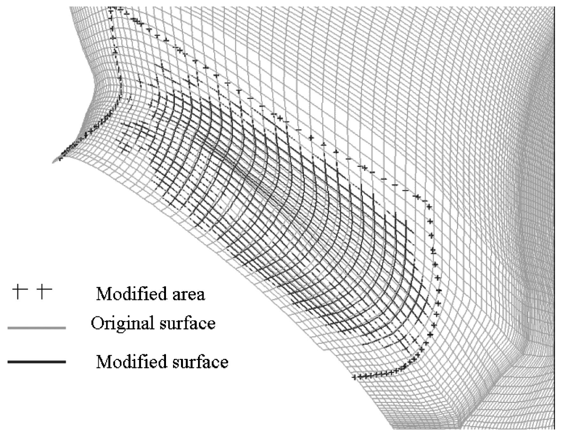


Fig. 16 Fillet patches modified automatically.

Combinations of the preceding deformation modes were permitted (each with their own upstream and downstream bounds and station of maximum effect). However, it was soon evident that the modification which proved most beneficial was a single application of the DENT1 mode.

Geometric Tools

Three programs implemented the preceding deformations on the original geometry (Fig. 16):

1) M1 creates a modified surface geometry input for MGAERO, given the original surface geometry input and the deformation parameter file.

2) M2 creates modified surface patches for NES, based on original surface patches covering the fillet region, and the deformation parameter file.

3) M3 modifies the multiblock grid input for NES, based on the original grid input, and the surface patches from M2.

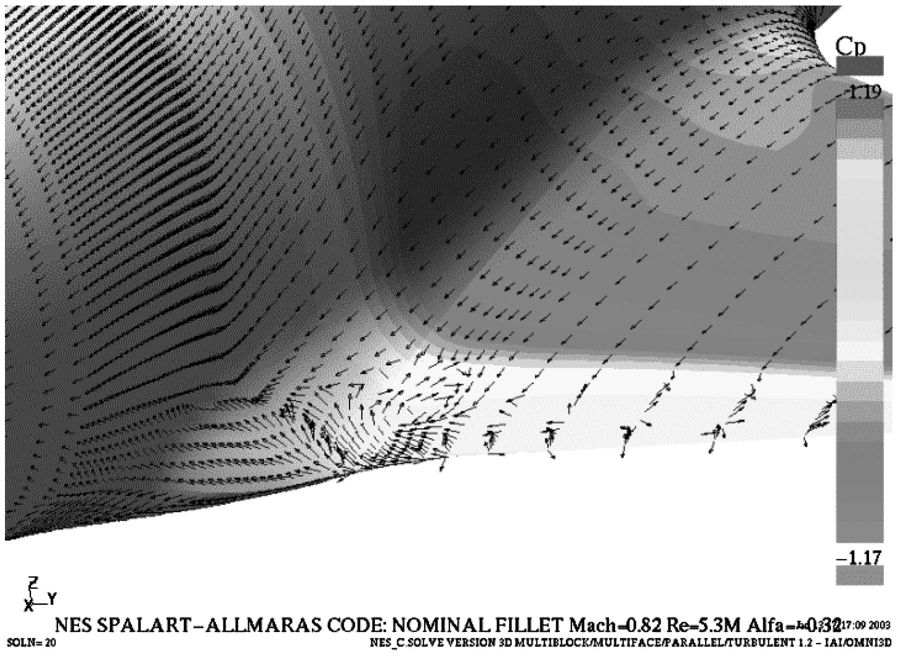


Fig. 17 NES SPALART-ALLMARAS CODE: Nominal fillet.

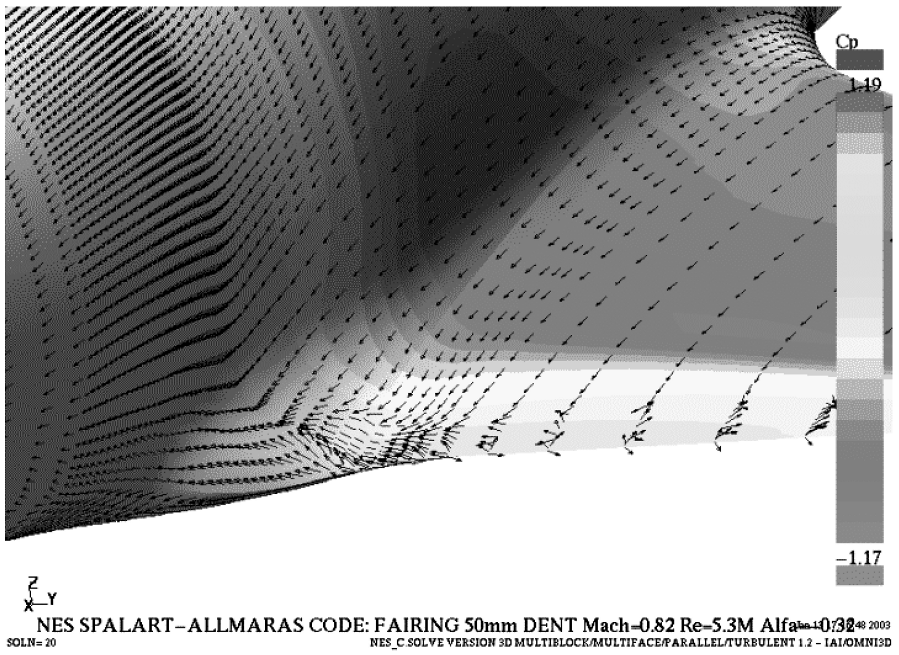


Fig. 18 NES SPALART-ALLMARAS CODE: Improved Filled (DENT 50mm).

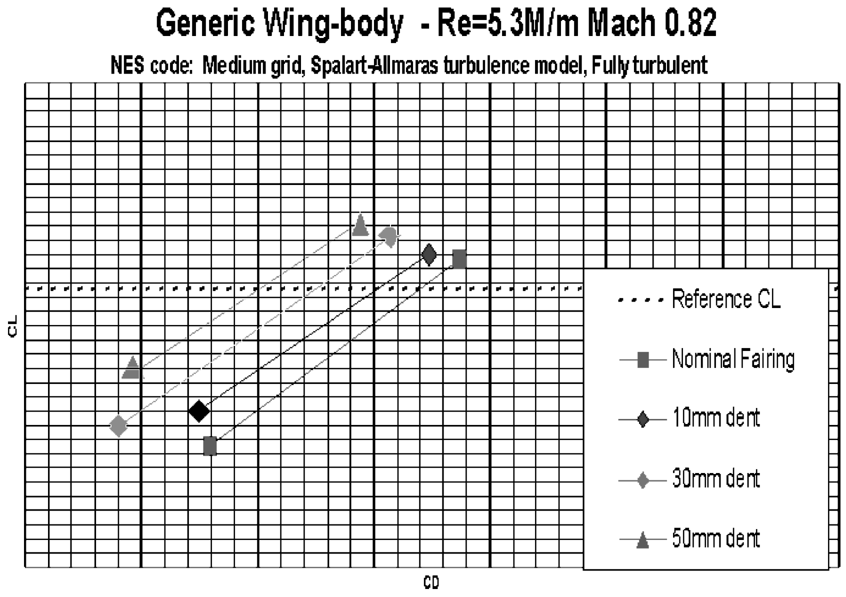


Fig. 19 CL versus CD Polars-Impact of design on drag performance.

Results of Fillet Design

Figures 17 and 18 show the Cp distribution superimposed with the velocity vectors close to the wall respectively for the nominal fillet and for the improved one (50 mm DENT). The intensity of the root vortex at the trailing edge of the junction wing body has been strongly lowered on the improved fairing.

CL vs CD polars are shown in Fig. 19, which indicated drag reductions of about 10 counts, at transonic cruise conditions. Various polars are shown for different suggested fillets, which have varying implications on the accompanying structural changes. The final choice of the optimal fillet rests on a tradeoff between aerodynamic performance and manufacturing feasibility.

Conclusions

The Spalart–Allmaras one-equation turbulence closure model implemented and validated in the cell-centered finite volume solver Navier–stokes Euler System (NES) has been parallelized. This model improves significantly the shock position compared with the Baldwin–Lomax model, and hence leads to an improved prediction of the lift and drag. The parallel capability of the code enabled the handling of runs of complex turbulent flows on realistic geometries on a daily basis. In that context the design of the wing-body fairing of a generic business jet has been implemented, based on the NES code.

Acknowledgments

We gratefully acknowledge the support of S. Gali of the Directorate for Defence Research and Development (MAFAT) in enabling this research.

References

[1]Epstein, B., Jacobs, A., and Nachshon, A., "Aerodynamically Accurate Three-Dimensional Navier–Stokes Method," *AIAA Journal*, Vol. 35, No. 6, 1997, pp. 1089, 1090.

[2]Epstein, B., and Nachson, A., "An ENO Navier–Stokes Applied to 2D Subsonic Transonic and Hypersonic Aerodynamic Flows," AIAA Paper 94-0303, Jan. 1994.

[3]Epstein, B., Averbuch, A., and Yavneh, I., "An Accurate ENO Driven Multigrid Method Applied to 3D Turbulent Transonic Flows," *Journal of Computational Physics*, Vol. 168, No. 2, 2001, pp. 316–338.

[4]Séror, S., Rubin, T., and Epstein, B., "Construction of a Multiblock 3D Full Navier–Stokes Code for Practical Aerodynamic Computations," *Proceedings of the 41st Israel Annual Conference on Aerospace Sciences*, Vol. 1, Technion Press, Haifa, Israel, 2001, pp. 231–241.

[5]Epstein, B., Rubin, T., and Séror, S., "An Accurate ENO Driven Navier–Stokes Solver for Complex Aerodynamic Configurations," *AIAA Journal,* Vol. 41, No. 4, 2003, pp. 582–594.

[6]Baldwin, B. S., and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257, 1978.

[7]Séror, S., Rubin, T., Epstein, B., and Arad, E., "Recent Enhancement of the 3D Navier–Stokes Code NES," *Proceedings of the 42th Israel Annual Conference on Aerospace Sciences*, Technion Press, Haifa, Israel, 2002.

[8]Séror, S., Rubin, T., Peigin, S., and Epstein, B., "Development of an Accurate Multiblock Multiface Parallel 3D ENO Driven Multigrid Cycling Navier–Stokes Code NES with the Spalart-Allmaras Model for Complex Aerodynamic Configurations," *Proceedings of the West East High Speed Flow Fields Conference*, Marseilles, France, CIMNE Press, Barcelona, 22–24 April 2002.

[9]Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, Jan. 1992.

[10]Séror, S., "Parallelization of the SA Model in NES—Final Status," Internal Document Israel Aircraft Industries, Lod. Israel, April 2002.

[11]Epstein, B., Luntz, A., and Nachshon, A., "Multigrid Euler Solver About Arbitrary Aircraft Configurations with Cartesian Grids and Local Refinements," AIAA Paper 89-1960, June 1989.

[12]Harten, A., Engquist, B., Osher, S., and Chakravarthy, S., "Uniformly High Order Accurate Non-Oscillatory Schemes. 1," *Journal of Computational Physics*, Vol. 71, 1987, pp. 231–303.

[13]Allmaras, S. R., "Multigrid for the 2D Compressible Navier–Stokes Equations," *Proceedings of the 14th Computational Fluid Dynamics Conference*, Norfolk, VA, 28 June–1 July 1999.

[14]Peigin, S., Epstein, B., Rubin, T., and Séror, S., "Parallel Multiblock Full Navier–Stokes Code NES: 5 Million Points Complete Aircraft Flow Simulation," *Proceedings of 42th Israel Annual Conference on Aerospace Sciences*, Technion Press, Haifa, Israel, 2002.

[15]Peigin, S., Epstein, B., Rubin, T., and Séror, S., "Parallel Large Scale High Accuracy Navier–Stokes Computations on Distributed Memory Clusters," *The Journal of Supercomputing*, Vol. 27, No. 1, 2004, pp. 49–68.

[16]Barak, A., Guday, S., and Wheeler, R., "The MOSIX Distributed Operating System, Load Balancing for UNIX," *Lecture Notes in Computer Science*, Vol. 672, Springer-Verlag, 1993.

[17]Martinelli, L., and Jameson, A., "Validation of a Multigrid Method for the Reynolds Averaged Equations," AIAA Paper 88-0414, Jan. 1988.

[18]Marconi, F., Siclari, M., Carpenter, G., and Chow, R., "Comparison of TLNS3D Computations with Test Data for a Transport Wing/Simple Body Configuration," AIAA Paper 94-2237, June 1994.