New convergence acceleration techniques in the Joe code

By F. Palacios AND J. J. Alonso

1. Motivation and objectives

The Joe code is a 2nd-order accurate (in space), finite volume, unstructured cell-based, implicit RANS (Reynolds-averaged Navier-Stokes equations) solver with flamelet-based chemistry (Pecnik *et al.* 2010*a*,*b*) developed within the Stanford PSAAP Center. This code was created to simulate the flow path inside of scramjet engines and, during the past few years, has served as the platform for numerical simulations on complex multiphysics problems. However, some robustness and computational cost aspects of the original code were not sufficiently fine-tuned, and, therefore, the improvement of these aspects has motivated the present research.

At the beginning of 2011, Joe included the following numerical methods for solving the RANS equations: space integration using a HLLC (Harten, Lax, van Leer contact wave) approximate nonlinear Riemann solver (Toro *et al.* 1994), 2nd-order extrapolation of the convective terms using a MUSCL scheme (Monotone Upstream-centered Schemes for Conservation Laws), and 2nd-order space integration of the viscous terms. Steady-state time integration could be carried out using a multistage Runge-Kutta scheme or a backward Euler method where the linear system is solved using preconditioned GMRES (Generalized Minimum RESidual) or a BiCGSTAB (BiConjugate Gradient STABilized) method.

Using these numerical methods, and when convergence is reached, the Joe solver results in accurate simulations. However, several issues present robustness challenges and prevent the application of the software to full 3D geometry UQ (Uncertainty Quantification) studies. It is important to note that UQ studies using the complete 3D Hyshot II geometry are an important milestone of the PSAAP center and, for that reason, significant effort has been devoted to increasing the robustness and convergence rates of the solver. In particular, during this past year, the following numerical methods have been added to the baseline Joe solver:

- New numerical discretization of the convective terms: both the AUSM (Advection Upstream Splitting Method) approximate nonlinear Riemann solver (Liou & Steffen 1993) and a JST (Jameson -Schmidt-Turkel) centered scheme (Jameson *et al.* 1981).
- A geometric, agglomeration-based multigrid scheme implemented and tuned for flows with shocks.
- New linear solvers: BiCGSTAB with linelet preconditioning and a LU-SGS (LU Symmetric Gauss Seidel) method.
- New estimation of the local time stepping for viscous and turbulent flows to enable larger CFL numbers.

The objective of this paper is to briefly introduce the new methods that have been implemented in Joe and to show the numerical improvements (robustness, and speed) that those techniques have produced. The overall goal has been to enable the 3D UQ simulations that the Center needs to pursue during the coming year.

2. Overview of the convergence acceleration techniques

Convergence acceleration techniques are methods for reducing the computational time needed to solve a CFD problem while preserving the accuracy of the solution. It is important to note that most of these techniques are based on the special treatment of the widely varying scales that can be found in a typical flow problem including turbulence, boundary layers, shock waves, and combustion. In particular, during 2011 the following techniques have been implemented in Joe:

- Multigrid method: The objective of this method is to correct the baseline fine-grid solution using information obtained in coarser grid levels in a grid hierarchy. Each coarser grid level is designed to eliminate a particular frequency bandwidth in the numerical error spectrum. This technique has been implemented in Joe with particular attention to hypersonic simulations using a geometric grid agglomeration technique.
- Linelet preconditioning technique: The original system is replaced by a preconditioned system that results in better convergence properties for iterative solution than the original system. Implicitness is specifically added in areas of the flow where the solution is highly coupled.
- Local time stepping strategy: This methodology is based on computing the maximum time step for each computational cell. In this work, a more accurate estimation of the time-step limit for turbulent flows has been included, leading to more effective updating of the solution across the entire flow domain.

On the other hand, the implicit implementation (considered in addition to the convergence acceleration techniques mentioned above) has been reviewed, and important changes have been introduced to increase the solver robustness. Note that Joe includes two different implicit formulations:

- Un-coupled, where the mean-flow, combustion model, and turbulence model are solved implicitly, but independently of each other (and an external iteration is followed to ensure the convergence of the entire system).
- Semi-coupled, where the mean-flow and combustion are implicitly coupled, and the turbulence model is solved independently.

The fully coupled formulation is also included in the code but was not used because solving all equations simultaneously may yield the most rapidly convergent scheme in number of iterations (higher CFL number), but not necessarily in CPU time. For that reason, the semi-coupled option was selected as a compromise between high CFL numbers and effective use of computational resources. Fully implicit formulations will be investigated at a later time.

3. Nonlinear multigrid method for hypersonic flows

The multigrid method generates effective convergence at all length scales of a problem by employing a sequence of grids of varying resolution. Simply stated, the main idea is to accelerate the convergence of the numerical solution of a set equations by computing corrections to the fine-grid solutions on coarse grids and applying this idea recursively(Mavriplis 1998, 1995; Borzi 2003). It is well known that, owing to the nature of most iterative methods/relaxation schemes, high-frequency errors are usually well damped, but low-frequency errors (global error spanning the solution domain) are less damped by the action of iterative methods with stencils with a local area of influence.

An agglomeration Full Approximation Storage (FAS) multigrid has been implemented in Joe. The basic methodology is described below. Consider the nonlinear problem $L(w) = f$ defined in a domain Ω, and denote its discretization on a fine grid with

spacing h as

$$L_h(u_h) = f_h, \text{ in } \Omega_h, \tag{3.1}$$

where $L_h(\cdot)$ is a nonlinear discrete operator defined in Ω_h. The starting point is the definition of a suitable smoother (e.g. Jacobi, SGS, GMRES, etc.) and, after a small number of iterations of this method (possibly a single one, instead of fully solving the discrete equation), an approximate solution $\bar{u}_h$ and residual r_h are obtained on the fine grid. The resulting equation in the fine grid can be written as

$$L_h(\bar{u}_h) - f_h = r_h. \tag{3.2}$$

Subtracting equations 3.1 and 3.2 we obtain the following expression to be approximated in a coarse grid:

$$L_h(u_h) - L_h(\bar{u}_h) = -r_h, \tag{3.3}$$

where the exact solution u_h can be expressed as the approximate solution plus a correction c_h yielding:

$$L_h(\bar{u}_h + c_h) - L_h(\bar{u}_h) = -r_h. \tag{3.4}$$

Note that no assumptions about the linearity of the operator $L(\cdot)$ (or its discrete version) are made. As we stated before, the objective is to write 3.4 in a coarse grid of spacing H. In order to do that, two types of restriction operators will be defined: I_h^H, the restriction operator that interpolates the residual from the fine grid h to the coarse grid H (in a conservative way), and $\bar{I}_h^H$, which simply interpolates the fine grid solution onto the coarse grid. Formulating 3.4 on the coarse level by replacing $L_h(\cdot)$ by $L_H(\cdot)$, $\bar{u}_h$ by $\bar{I}_h^H \bar{u}_h$, and r_h by $I_h^H r_h$, we obtain the FAS equation:

$$L_H(\bar{I}_h^H \bar{u}_h + c_H) - L_H(\bar{I}_h^H \bar{u}_h) = -I_h^H r_h. \tag{3.5}$$

In this last expression, by definition, the approximate solution on the coarse grid is denoted as $\bar{u}_H := \bar{I}_h^H \bar{u}_h + c_H$, and the residual r_h can be written as $L_h(\bar{u}_h) - f_h$. Finally we obtain the following useful equation on the coarse level:

$$L_H(\bar{u}_H) = L_H(\bar{I}_h^H \bar{u}_h) - I_h^H(L_h(\bar{u}_h) - f_h). \tag{3.6}$$

This last expression can also be simply written as

$$L_H(\bar{u}_H) = f_H + \tau_h^H, \text{ in } \Omega_H, \tag{3.7}$$

where the source term on the coarse levels is interpolated $f_H = I_h^H f_h$ (not computed), and a new variable $\tau_h^H = L_H(\bar{I}_h^H \bar{u}_h) - I_h^H(L_h \bar{u}_h)$ is defined as the fine-to-coarse defect or residual correction. Note that without the τ_h^H term the coarse grid equation is the original system represented on the coarse grid.

The next step is to update the fine grid solution. For that purpose the coarse-grid correction c_H (which in principle is smooth because of the application of the smoothing iteration) is interpolated back on to the fine grid using the following formula

$$\bar{u}_h^{new} = \bar{u}_h^{old} + I_H^h(\bar{u}_H^{new} - \bar{I}_h^H \bar{u}_h^{old}), \tag{3.8}$$

where I_H^h is a prolongation operator that interpolates coarse grid correction to the fine grid. Note that we interpolate the correction and not the coarse-grid solution itself.

In this brief introduction to the method only two grids have been considered. In real problems, however, the algorithm is applied in a recursive way using different grid level sizes to eliminate the entire spectrum of frequencies of the numerical error. In order to summarize the method, the basic multigrid algorithm is presented in pseudo-code below:

Algorithm *FAS Multigrid*

1. **if** $k = 1$
2. **then** solve $L_k(u_k) = f_k$ directly
3. **for** $l \leftarrow 1$ **to** ν_1
4. **do** Pre-smoothing steps on the fine grid:
5. $u_k^{(l)} \leftarrow S(u_k^{(l-1)}, f_k)$
6. Computation of the residual: $r_k \leftarrow f_k - L_k(w_k^{(\nu_1)})$
7. Restriction of the residual: $r_{k-1} \leftarrow I_k^{k-1} r_k$
8. $u_{k-1} \leftarrow \bar{I}_k^{k-1} u_k^{(\nu_1)}$
9. $f_{k-1} \leftarrow r_{k-1} + L_{k-1}(u_{k-1})$
10. Call γ times the FAS scheme to solve $L_{k-1}(u_{k-1}) = f_{k-1}$ using a V cycling strategy
11. Coarse-grid correction: $u_k^{new} \leftarrow u_k^{(\nu_1)} + I_{k-1}^k (u_{k-1} - \bar{I}_k^{k-1} u_k^{(\nu_1)})$
12. **for** $l \leftarrow 1$ **to** ν_1
13. **do** Post-smoothing steps on the fine grid:
14. $u_k^{(l)} \leftarrow S(u_k^{(l-1)}, f_k)$

Because of their structured-grid heritage, multigrid methods have traditionally been developed from a geometric point of view. In this particular implementation a geometric agglomeration multigrid method has been used. This strategy consists in choosing a seed point (a cell in a cell-based code such as Joe), which initiates a local agglomeration process whereby the neighboring control volumes are agglomerated onto the seed point. The topological fusing for the agglomeration multigrid method is a fundamental component of the algorithm: surface priorities and multiple restrictions to the set of volume (maximum number of points, volume, ratio surface/volume, boundary incompatibilities, etc) have been implemented. The most important advantage of the agglomeration technique is that it is not necessary to physically create independent meshes on the coarse levels: this task can be completely automated.

Whereas excellent results may be achieved for subsonic or transonic flows (see Fig. 1, and Fig. 2), unfortunately the standard multigrid method does not provides satisfactory results in hypersonic flows owing to the presence of strong shock waves (Koren & Hemker 1991; Kim 2001), highly stretched grids (Mavriplis 1998), and chemically reacting flows (Gerlinger *et al.* 1998, 2001). The problem is that if the source term contains strongly nonlinear parts, coarse grid values can differ too much from the corresponding fine grid values and, therefore, no longer represent the problem on the finest grid: even small coarse grid corrections interpolated to the finest grid may lead to strong changes that can prevent robust convergence.

It is important to remember that the efficiency of the multigrid strategy relies on the fact that the low-frequency errors are damped on the coarse levels (where a larger time step can be used). In chemically reacting flows, turbulent or strong shocked flows problems arise because of the high non-linearity of the solution: it is clear that the re-calculation of strongly nonlinear functions based on linearly averaged/interpolated variables from the fine grid causes major differences that prevent the convergence of the iterative method. In most of the cases the consequence is the failure of the multigrid approach, while in other situations the multigrid method simply does not aid in the convergence of the numerical scheme. For that reason, our objective is to create a selective multigrid method that is applied in the zones where it makes sense and does not apply any correction in those zones where multigrid could have a negative impact.

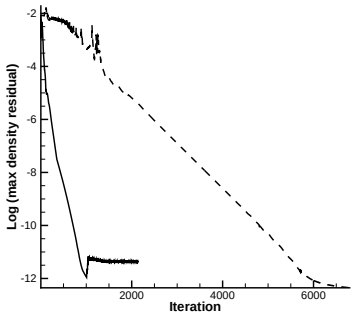

FIGURE 1. Supersonic flow around a cylinder (single grid – – – ; 5MG levels ———). First level 16384 hexahedra 1:4 reduction. 65856 faces. Speed up to residual level of 10^{-4} x17 (iterations)

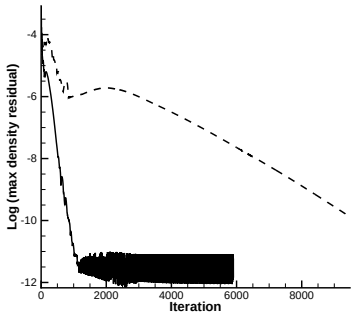

FIGURE 2. Subsonic laminar flow around a NACA0012 (single grid – – – ; 4MG levels ———). First level 16000 hexahedra 1:4 reduction. 64230 faces.Speed up to residual level of 10^{-6} x9 (iterations)

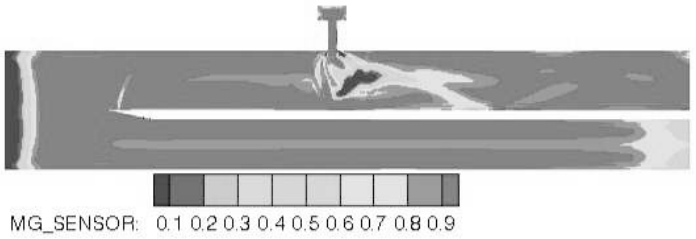

FIGURE 3. Visualization of the pressure and viscous sensors during a multigrid simulation of a jet in cross flow.

In Joe, two alternatives have been implemented in the quest for greater robustness: local damping of the restricted residual (based on temperature, pressure, and viscosity), using a pressure sensor

$$\epsilon_i = \left| \sum_{k=1}^{neigh} (p_k - p_i) \right| / \sum_{k=1}^{neigh} (p_k + p_i), \tag{3.9}$$

and uniform damping of the coarse grid correction. The advantage of the first methodology is that it has an apriori effect (aposteriori damping may not help). On the other hand, the uniform damping of the coarse-grid correction will have a positive effect on the robustness of the method but may strongly reduce the positive effects of the coarse-grid correction.

To sum up, in dealing with strong nonlinearities, the key idea is to not apply multigrid corrections in those zones where no positive effect is expected. In order to do this, it is important to damp part of the correction to avoid the incorrect reconstruction of the solution. This kind of technique must be applied in highly nonlinear situations, but should be avoided when dealing with subsonic and transonic applications of the multigrid algorithm. In Fig. 3 and Fig. 4 a successful application of this technique is presented. The example deals with the convergence of a calculations of a jet in a supersonic cross flow that was accelerated by a factor of 4 using this new multigrid approach.

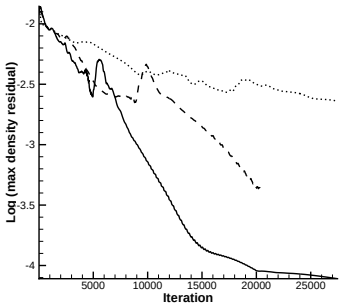

FIGURE 4. Convergence of the jet in cross flow problem: multigrid produces a speed-up of 4× (single grid ····· ; 3V MG, damping factor 0.75 — — — ; 3V MG, damping factor 0.25 ———).

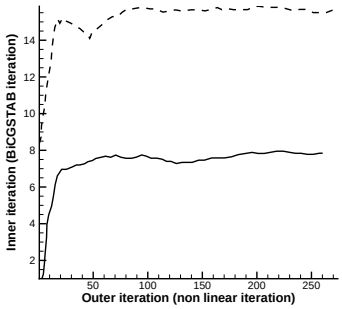

FIGURE 5. Effect of the preconditioning on a RANS simulation (Reynolds number $6.5\,10^6$). Linelet preconditioner (———) versus Jacobi preconditioner (— — —), the convergence criteria is 0.1.

4. Linelet preconditioning for solving RANS equations on highly stretched grids

Preconditioning is the application of a transformation to the original system that makes it more suitable for numerical solution (Pierce & Giles 1997). In particular, a linelet preconditioner has been implemented in Joe to improve the convergence rate of the preconditioned biconjugate gradient stabilized method (BiCGSTAB) solver for anisotropic meshes. The idea is (Soto *et al.* 2003; Mavriplis 1998) to construct lines in the mesh in the direction normal to the grid stretching. Then, the preconditioning matrix is built by assembling the diagonal entries of the system matrix and the non-diagonal entries of the edges that belong to these linelets. If the appropriate numbering is used, a block tridiagonal matrix is obtained, and the preconditioned system can be directly inverted using the Thomas algorithm. Note that in those zones where linelets are not defined, a Jacobi preconditioner is used.

In summary, in order to solve the system $Ax = b$ using the preconditioned BiCGSTAB method, it is necessary to solve (twice in each iteration) a system of equations of the form $Pz = r$, where P is the linelet preconditioner. The steps of the algorithm are:

1. Build grid lines (linelets) in the direction normal to the grid stretching.
2. Build the preconditioner matrix: assemble the diagonal and the non-diagonal entries (linelets) of the Jacobian matrix.
3. Do a nodal renumbering following the linelets to obtain a tridiagonal structure for the preconditioner.
4. Use the preconditioned formulation of the iterative scheme.
5. Solve the tridiagonal system for the linelets using the Thomas algorithm (direct method).

The linelet creation begins with the identification of all the points that are on the solid surface of the geometry (and where boundary layers and/or wakes are likely to exist), and the computation of an edge weight for each vertex on the surface, this weight is computed as:

$$w_{ij} = \frac{1}{2} S_{ij} \left(\frac{1}{V_i} + \frac{1}{V_j} \right), \tag{4.1}$$

where S_{ij} is the area of the face that separates nodes i and j and V is the volume of the control volume associated with each node. The line is built by adding to the original

vertex the vertex which is most strongly connected to the current vertex (maximum value of the weight). This new vertex is added only if it has not already been added to another linelet and if the weight is greater than a certain quantity that is used to mark the termination of the linelet. When an entire line is completed, the procedure is repeated starting with another vertex on the surface.

Once the list of linelets is completed, the nodal points must be renumbered following the linelets to obtain the desired tridiagonal structure of the preconditioner. First, the nodes of a linelet are renumbered from one end to the other. Then, a second linelet is renumbered, and so on until all linelets have been covered. Finally the rest of the points also have to be renumbered to accommodate the renumbering of the vertices that belong to the collection of linelets that have been identified.

The final step is to solve the system $Pz = r$ using the Thomas algorithm, in order to do that, the first step is to decompose the preconditioner matrix into upper- and lower-trianguler matrices U and L using the following algorithm (Soto *et al.* 2003):

$$\begin{pmatrix} D_1 & F_1 & 0 & \cdots & 0 \\ E_2 & D_2 & F_2 & \cdots & 0 \\ 0 & E_3 & D_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & E_n & D_n \end{pmatrix} = \begin{pmatrix} I & 0 & 0 & \cdots & 0 \\ L_2 & I & 0 & \cdots & 0 \\ 0 & L_3 & I & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & L_n & I \end{pmatrix} \times \begin{pmatrix} U_1 & F_1 & 0 & \cdots & 0 \\ 0 & U_2 & F_2 & \cdots & 0 \\ 0 & 0 & U_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & 0 & U_n \end{pmatrix} \tag{4.2}$$

Algorithm *Thomas algorithm*

1. $U_1 \leftarrow D_1$
2. **for** $i \leftarrow 2$ **to** n
3. **do** Obtain U_{i-1}^{-1}
4. $L_i \leftarrow E_i U_{i-1}^{-1}$
5. $U_i \leftarrow D_i - L_i F_{i-1}$

Once L and U have been obtained, the solution of the system $Pz = r$ is computed by performing the following substitutions:

$$y_i = r_i - L_i y_{i-1}, (y_1 = r_1), i = 2, ..., n; \tag{4.3}$$

$$z_i = U^{-1}(y_i - F_i z_{i+1}), (z_{n+1} = 0), i = n, ..., 1. \tag{4.4}$$

It is important to highlight that the LU decomposition is done only once, and it can be done on a per-linelet basis. On the other hand, an LU factorization is performed at each block of the implicit block matrix to compute the inverse of the block matrix. As it was pointed above, thanks to the particular renumbering chosen, in those points where a linelet is not defined, a Jacobi preconditioner is recovered by default.

The objective of the linelet preconditioner is to reduce the number of inner iterations in an implicit solver to obtain a particular level of convergence. In Fig. 5 it is possible to obtain a 40% reduction in the number of inner iterations (with respect to Jacobi preconditioner) for a transonic airfoil simulation at a Reynolds number, $Re = 6.5\,10^6$.

5. Implicit formulation and local time stepping

Implicit formulations and local time stepping techniques are powerful methods for convergence acceleration. These techniques require the estimation of eigenvalues and first-order approximations to the Jacobians. With respect to the local time step, our

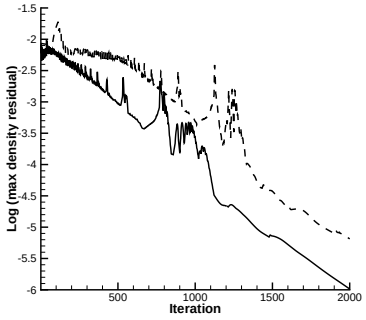

FIGURE 6. Effect of the new boundary Jacobian in the convergence of the hypersonic (1st-order) flow around a cylinder (old formulation – – – ; new formulation ———).

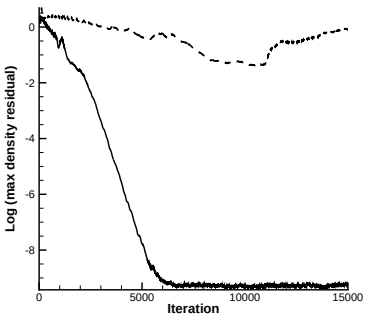

FIGURE 7. Effect of the new boundary jacobian, and time step evaluation in the convergence of the hypersonic (2nd-order) flow around a cylinder (old formulation – – – ; new formulation ———).

scheme avoids the use of a single time step over the entire computational domain for meshes with inhomogeneous cell densities. The local time step is computed (Eliasson 2007) for each node i according to

$$\Delta t_i = CFL \min\left(\frac{V_i}{\lambda_i^{conv}}, \frac{V_i}{\lambda_i^{visc}}\right), \tag{5.1}$$

where V_i is the volume of the cell i and λ_i^{conv} is the integrated convective spectral radius computed as

$$\lambda_i^{conv} = \sum_{k=1}^{Neigh} (|\vec{u}_{ik} \cdot \vec{n}_{ik}| + c_{ik}) S_{ik}, \tag{5.2}$$

and $\vec{u}_{ik} = (\vec{u}_i + \vec{u}_k)/2$, and $c_{ik} = (c_i + c_k)/2$ denote the velocity and the speed of sound at the cell face. $\vec{n}_{ik}$ denotes the normal direction of the control surface, and S_{ik} its size. On the other hand, the viscous spectral radius λ_i^{visc} is computed as

$$\lambda_i^{visc} = \sum_{k=1}^{Neigh} C \frac{\mu_{ik}}{\rho_{ik}} S_{ik}^2, \tag{5.3}$$

where C is a constant, μ_{ik} is the sum of the laminar and eddy viscosities in a turbulent calculation, and ρ_{ik} is the density.

In addition, the implicit formulation of Joe has been reviewed, and the Jacobian evaluation on the symmetry boundaries (slip walls) has been changed using the analytical flux on the slip wall (Dwight 2006) (weak formulation imposed by setting only the flux over the boundary face) $f_{simm} = (0, pn_x, pn_y, pn_z, 0)^T$, and the Jacobian due to this flux is computed using the chain rule where initially computing the derivatives with respect to the primitive variables, simplifies the calculation considerably. After that change, the convergence of the un-coupled version of Joe has improved considerably in basic problems that we were not able to converge before. In particular, in Fig. 6 it is possible to understand the effect of introducing the new boundary condition, and in Fig. 7 the combined effect of the new boundary condition and new time step computation is presented for a basic hypersonic simulation.

6. Hyshot II simulation (full 3D geometry)

The objective of this section is to summarize the application of the algorithmic developments to increase the speed and robustness of the Joe code within the context of the simulation of the Hyshot II problem (full 3D geometry, $2.8\,10^6$ cells). The ability to simulate this case repeatedly (under changes to the operating characteristics of the scramjet) is of fundamental importance to the 3D UQ analysis we are pursuing this year at the PSAAP center: increasing the robustness and speed of the solver has become a high priority.

The original version of Joe (prior to 2011) was unable to fully converge the Hyshot II simulation in a robust manner. For that reason, before introducing more advanced convergence acceleration techniques (multigrid, and linelet), the objective was simply to increase the robustness of the code by re-implementing the following areas for the solver: boundary conditions, time step evaluation, numerical discretization of the convective terms and viscous terms, linear solver, and integration of the source term of the progress variables. After a thorough study of the convergence problems, three factors were identified as the most relevant to increase the convergence rate of the code:

- Time step evaluation and appropriate imposition of the numerical boundary conditions.
- The reacting solution starting from a well defined cold solution (at the same equivalence ratio).
- Implicit smoothing of the source term of the progress-variable equation helping the convergence of the reacting simulation.

Using those ideas, it is now possible to converge the un-coupled formulation of the Joe code even for reacting flows (in previous version of Joe this simulation did not converge). Although was now the convergence of the un-coupled simulation possible for reacting simulations, the required CFL number was very low and, for that reason, only the semi-coupled implementation is presented in this document.

All the simulations presented in this section are for reacting flows, and two different spatial reconstruction are compared: 2nd-order reconstruction for all the convective terms (including the reacting terms), and a mixed reconstruction which consists of 2nd-order for the mean-flow convective terms, but 1st-order for the rest of the scalars. The pure 1st-order discretization was ruled out because the low accuracy of the results (despite better convergence rates). Finally, to evaluate the convergence of the code with UQ purposes, some different metrics are shown in this section. In particular, we compare the percentage of subsonic flow in the domain and the weighted pressure on the device.

The initial test was performed using an equivalence ratio of 0.3. In Fig. 8 the convergence study for the mixed formulation is presented: it is important to highlight that the density residual has decreased by three orders of magnitude. On the other hand, in Fig. 9 the 2nd-order simulation is presented and a noteworthy CFL ramp from 0.5 to 3.0 used.

In Fig. 10 one can see the convergence of the weighted pressure metric for an equivalence ratio of 0.3. This metric is computed as the volume averaged pressure on the entire device, and was selected because of its relation with the prediction of the unstart phenomena. The percentage of supersonic flow in the device has been also computed to evaluate the convergence of the code. In Fig. 11 the convergence of that metric is studied for different CFL ramps.

Once the convergence of the second and mixed methods have been verified, the next step is to check if the solution provided by the mixed formulation is good enough in terms of accuracy. In Fig. 12 it is possible to see that the mixed order approximation

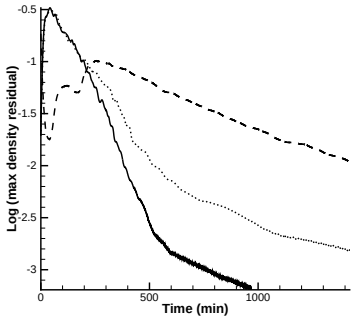

FIGURE 8. Residual convergence of the mixed order Hyshot simulation using an equivalence ratio of 0.3 (CFL 0.5 ····· ; CFL ramp from 0.5 to 2.0 — — — ; CFL ramp from 0.5 to 3.0 ———).

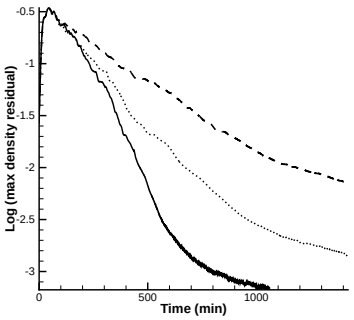

FIGURE 9. Residual convergence of the second order Hyshot simulation using an equivalence ratio of 0.3 (CFL 0.5 ····· ; CFL ramp from 0.5 to 2.0 — — — ; CFL ramp from 0.5 to 3.0 ———).

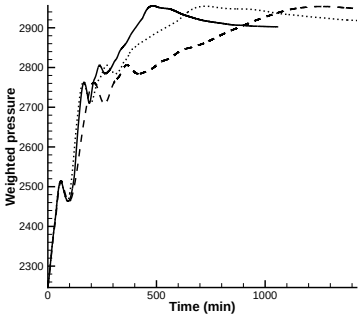

FIGURE 10. Weighted pressure in the entire domain for the second order formulation using an equivalence ratio of 0.3 (CFL 0.5 ····· ; CFL ramp from 0.5 to 2.0 — — — ; CFL ramp from 0.5 to 3.0 ———).

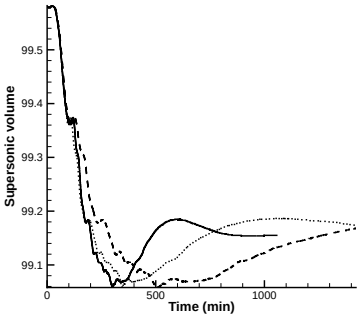

FIGURE 11. Supersonic flow percentage in the entire domain for the second order formulation using an equivalence ratio of 0.3 (CFL 0.5 ····· ; CFL ramp from 0.5 to 2.0 — — — ; CFL ramp from 0.5 to 3.0 ———).

does not exhibit the same level of accuracy as that of the 2nd-order one in the pressure distribution on the body. For that reason, the mixed order formulation was ruled out. Moreover, a scalability study has been performed (see in Fig. 13), the conclusion is that the code scales linearly up to 40000 cells per core.

Numerical experiments have been also performed at higher equivalence ratios. In particular, the results for an equivalence ratio of 0.4 are shown in Fig. 14, and in Fig. 15 the convergence of the weighted pressure is presented. Note that an unsteady behavior develops: this prevents full convergence of the steady state solver.

In conclusion, with the current version of Joe, it is possible to obtain a Hyshot II scramjet simulation with a good level of convergence for 2nd-order simulations using equivalence ratios between 0.3 and 0.4.

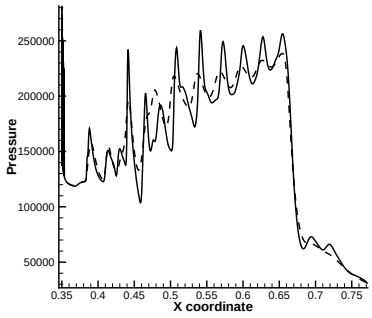

FIGURE 12. Pressure distribution on the body wall of the scramjet using a second (———) and a mixed method (— — —).

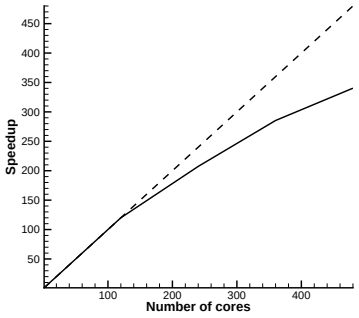

FIGURE 13. Scalability study of the Hyshot II reacting simulation (Joe scalability ——— ; lineal scalability — — —).

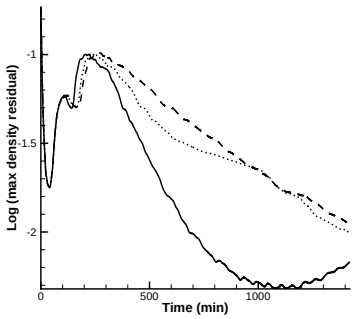

FIGURE 14. Residual convergence for equivalence ratio of 0.4, and second order method (CFL 0.5 ····· ; CFL ramp from 0.5 to 1.5 — — — ; CFL ramp from 0.5 to 2.0 ———).

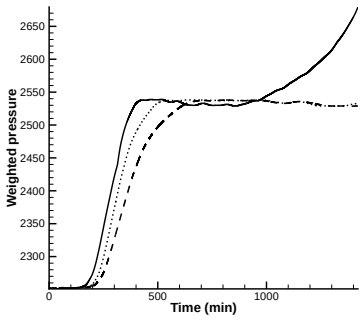

FIGURE 15. Weighted pressure convergence for equivalence ratio of 0.4, and second order method (CFL 0.5 ····· ; CFL ramp from 0.5 to 1.5 — — — ; CFL ramp from 0.5 to 2.0 ———).

7. Conclusions and future work

The current version of Joe now includes two of the most successful convergence acceleration techniques: multigrid, and linelet preconditioning. These techniques have been implemented in parallel and have been tested in RANS simulations with satisfactory results (the code is between 3 and 4 times faster).

Due to the improvements in the basic numerics, the robustness has improved dramatically for the Hyshot II problem and it is now possible to converge the simulations three orders of magnitude (maximum CFL number of 3.0) in 24 hours using 100 processors (certainty.stanford.edu cluster). This is a remarkable benchmark that allows the application of UQ techniques to the entire system. In addition, the un-coupled formulation (with lower memory requirements) is also converging using a lower CFL number.

Finally, several metrics have been tested, and more than 100 simulations of the entire Hyshot II geometry (3D) have been performed to adjust the parameters to maximize convergence. The next step is the integration of the convergence acceleration techniques in the Hyshot II simulation to reduce the computational cost by three or four times.

Acknowledgments

This work was supported by the U.S. Department of Energy under the Predictive Science Academic Alliance Program (PSAAP).

REFERENCES

Borzi, A. 2003 Introduction to multigrid methods. *Tech. Rep.*. Institut für Mathematik und Wissenschaftliches Rechnen (Karl-Franzens-Universität Graz).

Dwight, R. P. 2006 Efficiency improvements of rans-based analysis and optimization using implicit and adjoint methods on unstructured grids. *Tech. Rep.*. German Aerospace Center (DLR).

Eliasson, P. 2007 Edge. theoretical formulation. *Tech. Rep.*. Defence and Security, Systems and Technology (Swedish Defence Research Agency).

Gerlinger, P., Mobus, H. & Bruggemann, D. 2001 An implicit multigrid method for turbulent combustion. *J. Comput. Phys.* **167**, 247–276.

Gerlinger, P., Stoll, P. & Bruggemann, D. 1998 An implicit multigrid method for the simulation of chemically reacting flows. *J. Comput. Phys.* **146**, 322–345.

Jameson, A., Schmidt, W. & Turkel, E. 1981 Numerical solutions of the euler equations by finite volume methods using runge-kutta time-stepping schemes. *AIAA Paper 81-1259* .

Kim, S. 2001 Artificial damping in multigrid methods. *Appl. Math. Letters* **14** (3), 359–364.

Koren, B. & Hemker, P. W. 1991 Damped, direction-dependent multigrid for hypersonic flow computations. *Appl. Math. Letters* **7**, 309–328.

Liou, M.-S. & Steffen, C. 1993 A new flux splitting scheme. *J. Comput. Phys.* **107**, 23–39.

Mavriplis, D. J. 1995 Multigrid techniques for unstructured meshes. *Tech. Rep.*. Institute for computer applications on science and engineering (ICASE).

Mavriplis, D. J. 1998 On convergence acceleration techniques for unstructured meshes. *Tech. Rep.*. Institute for computer applications on science and engineering (ICASE).

Pecnik, R., Terrapon, V. E., Ham, F. & Iaccarino, G. 2010*a* Full-system RANS of the Hyshot II scramjet. Part 1: Numerics and non-reactive simulations. *Tech. Rep.*. Center for Turbulence Research, Annual Research Brief 2010.

Pecnik, R., Terrapon, V. E., Ham, F. & Pitsch, H. 2010*b* Full-system RANS of the Hyshot II scramjet. Part 2: Reactive cases. *Tech. Rep.*. Center for Turbulence Research, Annual Research Brief 2010.

Pierce, N. & Giles, M. 1997 Preconditioned multigrid methods for compressible flow calculations on stretched meshes. *J. Comput. Phys.* **136**, 425–445.

Soto, O., Lohner, R. & Camelli, F. 2003 A linelet preconditioner for incompressible flow solvers. *Int. J. Numer. Meth. Heat Fluid Flow* **13** (1), 133–147.

Toro, E., Spruce, M. & Speares, M. 1994 Restoration of the contact surface in the hll-riemann solver. *Shock Waves* **4** (1), 25–34.