

Preconditioned Multigrid Simulation of an Axisymmetric Laminar Diffusion Flame *

Samir Karaa, [†] Jun Zhang, [‡]

Laboratory for High Performance Scientific Computing and Computer Simulation,
Department of Computer Science, University of Kentucky,
773 Anderson Hall, Lexington, KY 40506-0046, USA

Craig C. Douglas [§]

Center for Computational Sciences, and
Department of Computer Science, University of Kentucky,
325 McVey Hall, Lexington, KY 40506-0045, USA

October 29, 2002

Abstract

We employ a damped Newton multigrid algorithm to solve a nonlinear system arising from a finite difference discretization of an elliptic flame sheet problem. By selecting the generalized minimum residual method as the linear smoother for the multigrid algorithm, we conduct a series of numerical experiments to investigate the behavior and efficiency of the multigrid solver in solving the linearized systems, by choosing several preconditioners for the Krylov subspace method. It is shown that the overall efficiency of the damped Newton multigrid algorithm is highly related to the quality of the preconditioner chosen and the number of smoothing steps done on each level. ILU preconditioners based on the Jacobian pattern are found to be robust and provide efficient smoothing but at an expensive cost of storage. It is also demonstrated that the technique of mesh sequencing and multilevel correction scheme provides significant CPU saving for fine grid calculations by limiting the growth of the Krylov iterations.

Key words - laminar diffusion flame, vorticity-velocity formulation, nonlinear methods, ILU preconditioning, Newton multigrid algorithm.

*Technical Report No. 359-02, Department of Computer Science, University of Kentucky, Lexington, KY, 2002. This research work was supported in part by the U.S. National Science Foundation under grants CCR-9902022, CCR-9988165, CCR-0092532, and ACI-0202934, by the U.S. Department of Energy Office of Science under grant DE-FG02-02ER45961, by the Japanese Research Organization for Information Science & Technology, and by the University of Kentucky Research Committee.

[†]E-mail: samir@csr.uky.edu.

[‡]E-mail: jzhang@cs.uky.edu, URL: <http://www.cs.uky.edu/~jzhang>.

[§]E-mail: douglas@ccs.uky.edu, URL: <http://www.ccs.uky.edu/~douglas>.

1 Introduction

Developing efficient solvers for nonlinear system of equations arising from combustion simulation problems is still of big interest. The difficulties associated with solving high heat combustion stem from the large number of dependent unknowns, the nonlinear characteristics of the governing partial differential equations and the different length scales present in the problem. Gaz-jet laminar diffusion flame is a problem of importance since it can be analyzed in detail with the current computer sources, and the understanding of the flame plays a central role in the modeling of more complex problems, such as detailed kinetics models or turbulent reacting flows.

In this work we are interested in a simple model of laminar diffusion flames. We consider a flame sheet problem with one-step chemical reaction. The governing equations for this model are highly nonlinear and the dependent variables are strongly coupled so that their solution constitutes a challenging test for nonlinear elliptic solvers. The flame sheet model adds only one field to the hydrodynamic fields that describe the underlying flow, whereas a detailed kinetics model adds as many fields as species considered in the kinetic mechanism. In order to model the flame structure more accurately, we use the vorticity-velocity formulation of the fluid flow equations [1, 3] instead of the traditional stream function-vorticity approach. The vorticity-velocity formulation allows for more accurate vorticity boundary conditions than the stream function-vorticity formulation does. It also yields better conditioned Jacobian, permits nonstaggered grids, and is easily extendible to three dimensions. A review of incompressible fluid flow computations using this formulation is well documented in [12].

The governing equations and boundary conditions are discretized on two dimensional unstructured rectangular grids using finite difference approximations. The numerical solution for the flame sheet problem is then obtained by solving a set of nonlinear algebraic equations of the form

$$F(U) = 0, \tag{1}$$

where U is the unknown vector. In the present work, we consider solving the nonlinear elliptic problem using a Damped Newton Multigrid algorithm (NMG). The nonlinear problem is recursively solved on a coarse grid and the solution interpolated onto a refined grid. NMG can be seen as a nested iteration multilevel algorithm (Full Multigrid method) where at each level a linearized system is solved using a multilevel correction algorithm and a damped Newton step is performed. The linear correction steps just use the last Jacobian formed (and saved) on the coarse level. The main difference between NMG and the Full Approximation Scheme (FAS) [8] is that NMG uses the multigrid algorithm as a part of a linearization technique (Damped Newton's method in our case) while FAS applies directly the multigrid algorithm using nonlinear iterative methods.

In practice, Newton method fails to converge when it starts from an initial vector not close enough to the solution. Some procedure must then be used in order to find an appropriate initial guess within the ball of the convergence of the Newton method. In general, the interpolation of a computed solution from a coarse level is not guaranteed

to fall within the Newton ball of the convergence of the next finer level. The use of an extra procedure on fine grids can be extremely expensive and may dominate the overall execution time. It is still a poorly understood question how to ensure that the interpolated solution is within the convergence domain of the Newton method on the next finer grid.

The NMG algorithm was extensively used in [9, 11] to solve the flame sheet problem. It was found that the dominant cost in the NMG algorithm in terms of computational and storage requirements is incurred by the code components in which the linearized system at each step of the Newton method is approximately solved. In [2], the flame sheet problem was solved using two grid levels. The authors investigated the performance of several preconditioners combined with Krylov subspace methods on a fine 81×81 tensor product grid. The numerical experiments showed that the Gauss-Seidel preconditioner yields better execution time with respect to the incomplete LU factorization preconditioner with zero level ILU(0). The disadvantage of using the ILU(0) preconditioner was mainly attributable to its ineffectiveness during the pseudo-transient continuation process to bring the iterate into the convergence domain of the steady Newton method. The preconditioner used in the previous studies by Douglas, Ern, and Smooke was a Gauss-Seidel left preconditioner [9, 11].

In the present work we test some more powerful preconditioners based on the incomplete LU factorization of the Jacobian matrix. We show that the use of ILU preconditioners is attractive in a Newton multigrid algorithm. The superiority of the ILU preconditioners over classical preconditioners such as the Gauss-Seidel preconditioner is expected during the fine grid computations of the steady Newton iterations.

It should be noted that there has been recently a growing interest in developing adaptive gridding methods for combustion problems, which is not considered in this work. Relevant references for adaptive gridding methods are [6] and [7], where mesh adaptive finite volume method is proposed. The efficiency of adaptive finite element and finite difference methods applied to laminar flames are studied in [4] and [5], respectively.

The remainder of the paper is organized as follows. The next section briefly describes the physical problem and the governing equations. In Section 3, we present the computational procedure, including the discretization techniques, the numerical evaluation of the Jacobian matrix and the pseudo-transient process. In Section 4, we conduct a series of numerical experiments and compare the performance of several preconditioners, and finally concluding remarks are given in Section 5.

2 Model description

The model for the flame sheet problem consists of a full set of two-dimensional axisymmetric Navier-Stokes equations for low-Mach number flow coupled together with a conserved scalar equation expressing the conservation of energy. The geometrical set-up of the problem is shown in Figure 1. It consists of an inner cylindrical fuel (methane) jet with radius $r_I = 0.2 \text{ cm}$ surrounded by a coflowing oxidizer (air) jet with radius $r_O = 2.5 \text{ cm}$. The methane and air are introduced through the inner and outer tubes with constant velocities,

both at the uniform temperature $T = 298K$. Since the solution is axisymmetric, the computational domain is two-dimensional and it covers the area from $r = 0$ to $r_{max} = 7.5 \text{ cm}$ in the radial direction and from $z = 0$ to $z_{max} = 30 \text{ cm}$ in the axial direction.

As already mentioned, we formulate the problem using the vorticity-velocity formulation for the Navier-Stokes equations. We introduce the velocity vector $v = (v_r, v_z)$ with radial and axial components v_r and v_z respectively. The vorticity ω is defined in terms of the components of the velocity as $\omega = \frac{\partial}{\partial z}v_r - \frac{\partial}{\partial r}v_z$. The vorticity transport equation is formed by taking the curl of the momentum equations, which eliminates the partial derivatives of the pressure field. A Laplace equation is obtained for each velocity component by taking the gradient of w and using the continuity equation.

Let ρ denote the density, μ the viscosity, g the gravity vector, $\text{div}(v)$ the cylindrical divergence of the velocity vector, S the conserved scalar, and D a diffusion coefficient. The components of $\bar{\nabla}\beta$ are $(\frac{\partial}{\partial z}\beta, -\frac{\partial}{\partial r}\beta)$. Then the governing system for the flame sheet problem is given by the following four equations:

1.
$$\frac{\partial^2 v_r}{\partial r^2} + \frac{\partial^2 v_r}{\partial z^2} = \frac{\partial \omega}{\partial z} - \frac{1}{r} \frac{\partial v_r}{\partial r} + \frac{v_r}{r^2} - \frac{\partial}{\partial r} \left(\frac{v \cdot \nabla \rho}{\rho} \right),$$
2.
$$\frac{\partial^2 v_z}{\partial r^2} + \frac{\partial^2 v_z}{\partial z^2} = -\frac{\partial w}{\partial r} - \frac{1}{r} \frac{\partial v_r}{\partial z} - \frac{\partial}{\partial z} \left(\frac{v \cdot \nabla \rho}{\rho} \right),$$
3.
$$\begin{aligned} \frac{\partial^2 \mu w}{\partial r^2} + \frac{\partial^2 \mu w}{\partial z^2} + \frac{\partial}{\partial r} \left(\frac{\mu w}{r} \right) = \\ \rho v_r \frac{\partial w}{\partial r} + \rho v_z \frac{\partial w}{\partial z} - \frac{\rho v_r}{r} w + \bar{\nabla} \rho \cdot \nabla \frac{v^2}{2} - \bar{\nabla} \rho \cdot g + \\ 2 \left(\bar{\nabla} (\text{div}(v)) \cdot \nabla \mu - \nabla v_r \cdot \bar{\nabla} \frac{\partial \mu}{\partial r} - \nabla v_z \cdot \bar{\nabla} \frac{\partial \mu}{\partial z} \right), \end{aligned}$$
4.
$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \rho D \frac{\partial S}{\partial r} \right) + \frac{\partial}{\partial z} \left(\rho D \frac{\partial S}{\partial z} \right) = \rho v_r \frac{\partial S}{\partial r} + \rho v_z \frac{\partial S}{\partial z},$$

which respectively represent the radial and axial velocities, the vorticity, and the conserved scalar S . The temperature and the major species profiles in the system are recovered from the conserved scalar, as detailed for instance in [16]. The density is computed using the perfect gas law as a function of the pressure. The flow's small Mach number allows the pressure to be approximated as a constant. The pressure field is then eliminated from the governing equations as a dependent unknown, and it can be recovered once a computed numerical solution of the governing equations is obtained, by solving a Laplace equation derived by taking the divergence of the momentum equations.

The system of governing equations is closed with appropriate boundary conditions on each side of the computational domain:

- Axis of symmetry ($r = 0$): $v_r = 0$, $\frac{\partial v_r}{\partial r} = 0$, $\omega = 0$, $\frac{\partial S}{\partial r} = 0$.
- Outer zone ($r = r_{max}$): $\frac{\partial v_r}{\partial r} = 0$, $\frac{\partial v_z}{\partial r} = 0$, $\omega = \frac{\partial v_r}{\partial z}$, $S = 0$.
- Inlet ($z = 0$): $v_r = 0$, $v_z = v_z^0(r)$, $\omega = \frac{\partial v_r}{\partial z} - \frac{\partial v_z}{\partial r}$, $S = S^0(r)$.

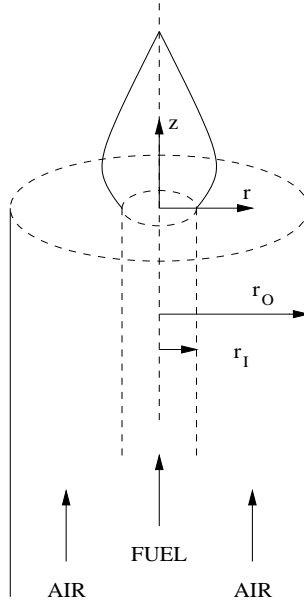


Figure 1: Physical configuration for the axisymmetric flame (not in scale).

- Exit ($z = z_{max}$): $v_r = 0$, $\frac{\partial v_z}{\partial z} = 0$, $\frac{\partial \omega}{\partial z} = 0$, $\frac{\partial S}{\partial z} = 0$.

To begin the solution process, a starting estimate for each dependent variables is needed. The v_r and v_z profiles are well known, and that of ω can be approximated by assuming negligible axial variations. The inlet profile of the conserved scalar, $S^0(r)$, is a slightly rounded step function that blends room temperature reservoirs of fuel and oxidizer by means of a narrow Gaussian in temperature centered at r_I [1].

3 Computational procedure

The PDE system and the boundary conditions are discretized on a two-dimensional tensor product grid using finite difference approximations. The grid is formed by the intersection of the lines of the horizontal mesh

$$\{0 = r_1 < \dots < r_{i-1} < r_i < r_{i+1} < \dots < r_{n_r} = r_{max}\},$$

and the vertical mesh

$$\{0 = z_1 < \dots < z_{i-1} < z_i < z_{i+1} < \dots < z_{n_z} = z_{max}\}.$$

The dependent variables are located at the regular mesh nodes (r_i, z_i) . Diffusion and source terms in the partial differential equations are approximated using central differences. We adopt two different schemes for the convection terms. We first use a monotonically preserving upwind scheme. For instance, the convection terms in the radial direction are

discretized as

$$\left(v_r \frac{\partial S}{\partial r}\right)_{i,j} = \max\{(v_r)_{i-\frac{1}{2},j}, 0\} \frac{S_{i,j} - S_{i-1,j}}{r_i - r_{i-1}} - \max\{-(v_r)_{i+\frac{1}{2},j}, 0\} \frac{S_{i+1,j} - S_{i,j}}{r_{i+1} - r_i},$$

where the subscript $i \pm \frac{1}{2}$ refers to the value averaged between the nodes i and $i \pm 1$. This upwind difference scheme makes the discrete system easy to solve and increase the efficiency of the linear system solution method. We next apply the following central difference approximation to the first derivative:

$$\left(\frac{\partial S}{\partial r}\right)_{i,j} = \frac{S_{i+1,j}(\Delta r_i)^2 - S_{i-1,j}(\Delta r_{i+1})^2 + S_{i,j}[(\Delta r_{i+1})^2 - (\Delta r_i)^2]}{\Delta r_{i+1} \Delta r_i (\Delta r_{i+1} - \Delta r_i)},$$

where $\Delta r_i = r_i - r_{i-1}$ and $\Delta r_{i+1} = r_{i+1} - r_i$. This approximation has a second order truncation error on any grid [19], and will be useful to test the performance of our preconditioners. The boundary conditions involve only Dirichlet or the first order derivatives. A proper discretization of the boundary conditions is extremely important since it may exert a strong influence on the overall accuracy of the numerical solution. A detailed study is presented in [1], where second order accurate discretizations are sought for all boundary conditions.

The discretization of the PDE system and the boundary conditions generates a set of nonlinear algebraic equations of the form (1), which is then solved using a damped Newton method. At the n th iteration of the solution method, the $(n+1)$ th solution iterate is found from the n th iterate through solving the following equation:

$$J(U^n)(U^{n+1} - U^n) = -\lambda^n F(U^n), \quad n = 0, 1, \dots, \quad (2)$$

where the Jacobian matrix is given by $J(U^n) = \partial F(U^n)/\partial U$, and λ^n ($0 < \lambda^n \leq 1$) is the damping parameter. The Newton iteration (2) is performed until the norm of the discrete vector $U^{n+1} - U^n$ is reduced appropriately, e.g., below 10^{-5} .

Since all computational stencils in the finite difference approximations use at most nine points, the Jacobian matrix is a block 9-diagonal matrix in which each block is a 4×4 dense matrix. Due to the tensor product nature of the grid and the compact form of the computational stencil, an inexpensive way based on the idea outlined in [15] is used to numerically evaluate the Jacobian matrix. The columns of the Jacobian matrix are formed using vector function evaluations with the grid nodes split into nine independent groups, which are perturbed simultaneously [16].

To obtain appropriate starting estimate for the Newton iteration (2), the original elliptic problem is converted into a time-dependent parabolic problem with appropriate initial conditions. The original nonlinear discrete problem is then transformed, using an implicit Euler scheme, into

$$F(U^{(n+1)}) + D \frac{U^{(n+1)} - U^{(n)}}{\Delta t^{(n+1)}} = 0, \quad (3)$$

where D is a diagonal scaling matrix and Δt^{n+1} is the $(n+1)$ st time step. The new nonlinear problem is again solved by a damped Newton method. The number of timesteps

needed to bring the initial guessed solution into the convergence domain of the Newton method depends on the size of the grid. The coarser the grid is, the fewer timesteps are needed.

4 Numerical experiments

In this section, we present several numerical results obtained on a single 750 MHz HP PA-RISC 8700 processor of an HP superdome supercluster at the University of Kentucky. We consider the finest grid to be a 161×193 tensor product grid, and we construct a sequence of nested grids by successively discarding every other node from one grid to the coarser one. In the first part of the experiments, the inlet velocities of the fuel and oxidizer are set at 35 cm/s everywhere, which yields a Reynolds number of 550, and an upwind difference scheme is used for the convection terms. The original grid is highly anisotropic and most grid points are placed in the high activity regions near the origin.

The nonlinear problem (1) is first solved on a coarse grid, and the resulting solution is interpolated to the next finer grid. The nonlinear problem is then solved on the finer grid using the interpolated solution as an initial guess for the damped Newton method. This procedure is repeated recursively until the problem on the finest level is approximated well enough.

At each Newton step on a given level, a Jacobian is formed and the linearized system is approximately solved by a Multigrid V-cycle algorithm, making use of the last Jacobians computed on all levels coarser than the current level. At each grid level a given number of relaxation steps are performed, except on the coarsest grid where this number may increase so that the linear system can be solved to a high accuracy. Standard full-weighting and bilinear interpolation are employed as the intergrid transfer operators. We anticipate that as the size of the mesh spacing gets smaller, the interpolated solution from one grid lies in the convergence domain of the Newton method on the next finer level. In most experiments, this was found to be the case between all levels. We notice that an inner iteration may require many V-cycles to converge. In the case that the number of V-cycles exceeds 8, the time stepping process is automatically launched to make the Jacobian easier to invert.

For large scale linear systems, iterative methods based on Krylov subspaces such as BiCGSTAB [14] or GMRES [13], are known to be better suited than direct methods. BiCGSTAB is a smoothly convergent version of the Lanczos-based method BiCG. It has lower memory requirements but because of the lack of an optimality property of the residual, oscillatory behavior may be encountered, especially on fine grids. GMRES is however a minimal residual algorithm based on the use of the Arnoldi process. In a full GMRES implementation the storage requirements grow linearly and the work quadratically with the number of iterations. Hence, in practice it is often necessary to use a restarted version, GMRES(k), where k indicates the selected dimension of the Krylov subspace. In this case, the algorithm is restarted after k iterations. Restarting may cause GMRES to exhibit slow convergence, but prevent the algorithm from possible breakdown due to

memory depletion. The effectiveness of both iterative methods for solving reacting flow problems has been illustrated in [1] and [2].

In our experiments we found that it is more advantageous to use GMRES as the linear smoother in the multigrid algorithm. The convergence of GMRES will be based on the norm of the left preconditioned linear residual using an absolute tolerance proportional to the tolerance of the Newton iteration. Experience suggests that a suitable choice of the termination criterion can bring enough information on the update vector $U^{(n+1)} - U^{(n)}$ back to the Newton iteration.

In what follows, we compare the performance of the block Gauss-Seidel (GS) preconditioner and incomplete LU factorization based preconditioners. The incomplete LU factorization has been one of the best known preconditioning technique, and is often used to improve the convergence of Krylov subspace methods. In [17], it is shown that for any M-matrix B , there is a unique ILU factorization $Q = LU$, such that L is unit lower triangular, U is upper triangular, $l_{ij} = 0$ and $u_{ij} = 0$ for $(i, j) \notin \mathcal{N}$, and $[Q - B]_{ij} = 0$ for $(i, j) \in \mathcal{N}$, where \mathcal{N} is an index set containing all diagonal indices (i, i) . Experiments in various application fields, such as computational fluid dynamics, reveal that incomplete LU factorization preconditioners exist for a large class of problems, even though the coefficient matrix of the discrete problem is not an M-matrix. More discussions on different Krylov subspace methods and various type of incomplete LU factorization preconditioning techniques can be found in [18].

The dimension of the Krylov subspace is set at 30 in all experiments, so GMRES restarts after 30 iterations. Since our current interest is to study the robustness of the preconditioners, we only report the number of linear iterations and the execution time for each preconditioner as a measure of its relative performance. As the number of outer iterations was found to be nearly insensitive to the choice of the preconditioner, we will not report the nonlinear behavior of the Newton method.

Tables 1-3 contain the performance data of the GS and ILU(l) preconditioners using 3 levels, where l is the level of fill-in for the incomplete LU factorization. Before starting the steady Newton iteration on the coarsest level, we perform 30 preliminary timesteps with a initial timestep equal to 10^{-3} . This initial timestep is adaptively halved during the computation whenever the Newton iterations converge slowly or the nonlinear residual does not decrease after two iterations. In each table, we report the total number of GMRES linear smoothing iterations done on each level. The CPU time refers to the total CPU time in minutes needed to solve the nonlinear problem (1), including the 30 preliminary timesteps, while t_f is the time spent on the finest level. The number m is the maximum number of relaxation steps allowed on each non-coarsest level.

The first table shows the results for the GS preconditioner. The number of smoothing steps $m = 10$ was found insufficient for the inner linear iterations to converge within eight V-cycles. We found that the V-cycle multigrid algorithm still performs poorly after taking additional timesteps. The case $m = 15$ also meets some difficulties, while for m greater than 20 the V-cycle multigrid algorithm performs well without any additional timesteps except the first 30 ones done on the coarsest grid. The numerical results show that the

Table 1: Performance data of GMRES/Gauss-Seidel using three levels.

m	level1	level2	level3	CPU	t_f
20	211	548	2421	1.85	1.19
30	232	546	1801	1.92	1.30
40	254	586	1569	2.02	1.38

shortest overall execution time on the solution is obtained when performing 20 relaxation steps on each non-coarsest level. The CPU time spent (before reaching the finest level) to obtain an appropriate initial guess for the Damped Newton method on the finest level is equal to the difference $\text{CPU} - t_f$, representing about 40 seconds for $m = 20$. However, performing 30 timesteps on the finest level took about 9 minutes, much more expensive than solving the whole problem using a meshing sequence procedure. This demonstrates the severe cost of developing a global solution when starting on a fine grid.

Table 2: Performance data of GMRES/ILU(1) using three levels.

m	level1	level2	level3	CPU	t_f
10	66	156	627	1.13	0.68
15	84	180	587	1.30	0.83
20	98	213	584	1.40	0.91
30	140	253	549	1.71	1.22
40	176	282	548	1.94	1.42

Table 3: Performance data of GMRES/ILU(2) using three levels.

m	level1	level2	level3	CPU	t_f
10	68	135	442	1.46	0.92
15	85	168	441	1.76	1.13
20	101	189	410	1.85	1.26
30	137	218	390	2.17	1.56
40	166	210	368	2.34	1.71

The second and third tables present the performance data of the ILU(1) and ILU(2) preconditioners, respectively. Unlike the GS preconditioner, a number of relaxation steps equal to 10 is sufficient for the inner linear iterations to converge within a few number of V-cycles on all levels. The ILU(1) preconditioner performs better than the ILU(2) and the Gauss-Seidel preconditioners. The use of 10 relaxation steps on each non-coarsest

level delivers the lowest execution CPU time in both cases, which also represents a saving in storage when using GMRES. The fact that $m = 10$ works well compared to the first case is due to the fact that the ILU preconditioners are more robust than the Gauss-Seidel preconditioner. This may also be seen by comparing the numbers of iterations for each preconditioner performed on each level. The improvement in time comes of course at a higher cost of storage, since the ILU preconditioner approximately doubles the memory requirement of the problem. We notice that the GS preconditioner converges quite faster than the ILU preconditioners during the time stepping process on the coarsest level. This may be due to fact that, since the linear system on the coarsest grid requires a short computer time, the time spent for the setup of 30 ILU factorizations has a nonnegligible cost.

Table 4 shows the numerical results obtained using the $ILUT(p, \tau)$ preconditioner [18], with a fill-in parameter p and a threshold drop tolerance τ . Here, $m = 30$ for all cases. Unlike the $ILU(1)$ and $ILU(2)$ preconditioners based on the Jacobian pattern, the $ILUT$ preconditioner is based on threshold values and provides a flexible computational tool that allows the control of the memory requirement. The quality of the $ILUT$ preconditioner can be adjusted by choosing suitable values for the parameters p and τ . Among the four preconditioners compared, the $ILUT$ preconditioner is the most ineffective one. We notice that the use of the $ILU(0)$ preconditioner with no fill-in was also ineffective for this problem, as was reported in [2], and so its experimental performance data were omitted.

Table 4: Performance data of $ILUT(p, \tau)$ using three levels.

p	τ	level1	level2	level3	CPU	t_f
20	10^{-2}	356	921	2911	3.73	2.65
30	10^{-3}	257	562	1347	3.54	2.71
30	10^{-4}	212	561	1886	3.82	2.66
40	10^{-4}	209	441	1074	3.70	2.83
50	10^{-5}	184	370	866	2.80	1.92

The numerical results using four grid levels are shown in Table 5. The $ILU(1)$ preconditioner still performs better than the two other preconditioners, and the best performance is obtained for $m = 10$. The use of four levels help saving the overall CPU time but without a significant gain. Comparing the results reported in Table 5 with those of the Tables 1-4, we conclude that the saving in time is mainly due to the decrease of the CPU time spent on the third level. We note that the use of five grids did not perform better than the last case since it is found that the interpolated coarsest grid solution is not within the convergence domain of the Newton method on the next finer grid, so additional timesteps was needed. The use of very coarse grids causes however the Newton iteration to diverge.

In Table 6, we present the numerical results obtained when using a central difference scheme instead of an upwind difference scheme. The inlet oxidizer and fuel velocities are

Table 5: Comparison of different preconditioners using four multigrid levels.

m	level1	level2	level3	level4	CPU	t_f
Gauss-Seidel						
20	211	551	1955	1667	1.81	1.20
30	230	560	1422	1594	1.90	1.30
ILU(1)						
10	68	154	318	468	1.02	0.71
15	88	180	353	401	1.20	0.84
20	99	211	381	378	1.34	0.93
ILU(2)						
10	69	137	274	346	1.35	0.93
15	85	163	301	306	1.53	0.65
20	100	187	252	266	1.67	1.24

both set at 20 cm/s to make sure that no physical oscillations will be encountered. The Jacobian matrix is expected to be more difficult to invert compared to the preceding case. As before, thirty preliminary timesteps are performed before starting the steady Newton iteration with an initial time step equals 10^{-3} . This number was sufficient to bring the initial solution into the convergence domain of the Newton method.

The use of a left GS preconditioner meets with difficulties in the unsteady phase of the computation and fails to provide a good initial guess for the steady Newton iterations, even with the use of very small timestep size. The increase of the number of timesteps did not help reduce the norm of the nonlinear residual. The right GS preconditioner performs better on the coarsest level, but the computations terminate after a huge number of linear iterations. Approximately ten thousands GMRES iterations are performed during the unsteady and steady phases on the coarsest level. In the next finer level, the inner iteration fails to converge even after taking several timesteps to improve the Jacobian matrix.

In contrast to the GS preconditioner, the ILU preconditioners perform well and the inner iterations converge at all levels. The ILU(2) preconditioner performs much better than the ILU(1) preconditioner. The multigrid algorithm is very slow when the number of smoothing steps m is less than 40 in the ILU(1) case, while it converges quickly with $m = 10$ in the ILU(2) case. We also remark that, in contrast to the previous case, the best performance in the ILU(2) case is obtained when $m = 15$. In the experiments, the time stepping process was needed only on the coarsest level. The numerical results obtained during the time stepping process on each level are reported in Table 7. The time stepping using the ILU(1) preconditioner requires 17 seconds on the coarsest grid as opposed to over 13.2 minutes on the finest grid, thus yielding a speedup of about 47. We obtain a

Table 6: Comparison of ILU preconditioners using three levels (central difference scheme).

m	level1	level2	level3	CPU	t_f
ILU(1)					
40	405	630	1055	3.83	3.12
50	332	448	938	3.32	2.61
60	397	545	938	3.77	3.00
ILU(2)					
10	186	320	695	2.46	1.92
15	117	226	500	1.81	1.27
20	123	223	441	1.83	1.25
30	149	218	383	2.10	1.51

speedup of about 27 when using the ILU(2) preconditioner. We notice that the ILU(0) and ILUT preconditioners were found once again ineffective for this problem. The use of 4 levels also meets with difficulties since the Newton iteration fails to converge on the coarsest level.

Table 7: CPU time in minutes for running the simulation at each level during the time stepping phase.

	level1	level2	level3
ILU(1)			
CPU	13.20	1.38	0.28
Speedup	1.0	9.56	47.14
ILU(2)			
CPU	9.11	1.66	0.34
Speedup	1.0	5.48	26.79

In order to investigate the efficiency of using a multigrid algorithm as the linear solver for the Newton method, we rerun the computations and solve the linearized system on each level without any multilevel correction scheme. The last Jacobian on each coarser grid is no longer needed and so not saved. This procedure, where each of the coarser grids is used only to initialize the next finer grid, is sometime called one way nonlinear multigrid algorithm [9]. The numerical results in Table 8 show a significant growth of the Krylov iterations, especially at the finest level. A comparison with the results in Table 6 shows that the use of a residual correction scheme significantly decreases the overall execution time. This improvement is especially due to the saving in time at the finest

Table 8: One way multigrid using three levels (central difference scheme).

	<i>lev 1</i>	<i>lev 2</i>	<i>lev 3</i>	<i>CPU</i>	<i>t_f</i>
ILU(1)	605	208	788	4.57	3.84
ILU(2)	286	173	290	2.83	2.18

level, where most of the overall time of the one way multigrid method is spent in the smoothing iterations. It is worthwhile to point out that this saving in time comes at a higher cost in storage since the residual correction scheme requires saving a Jacobian matrix at each level. We finally performed Newton multilevel iterations using a W-cycle multigrid algorithm as the linear solver, but this did not help saving the overall solution time.

For completeness, a computed temperature profile of the simulated flame sheet in the first case is shown in Figure 5.

5 Concluding remarks

In this paper, a nonlinear problem arising from a finite difference discretization of a flame sheet problem is solved using a multilevel damped Newton method where the linearized system in the Newton method is solved using a V-cycle multigrid algorithm. By selecting different preconditioners for the Krylov subspace method GMRES chosen as the linear smoother, we conducted numerical experiments to investigate the efficiency of the multigrid algorithm in solving the linearized system. The Multigrid algorithm with a Gauss-Seidel preconditioner performs well when an upwind difference scheme is used to discretize convection terms, but found inefficient when a central difference scheme is used. The ILU(1) and ILU(2) preconditioners however are robust and deliver a less overall execution time than the Gauss-Seidel preconditioner. The ILU(0) and ILUT preconditioners are found inefficient for this problem. The use of a residual correction scheme in the inner iteration of the Newton method increases significantly the effectiveness of the overall solution algorithm, but at a very expensive cost of storage, since at each level a Jacobian and its corresponding incomplete LU factors are saved. The present study shows in particular that the multigrid algorithm without a strong enough smoother may perform very poorly.

References

- [1] A. Ern, Vorticity-velocity modeling of chemically reacting flows, *Ph. D. thesis, Yale University, February 1994, Mechanical Engineering Department.*

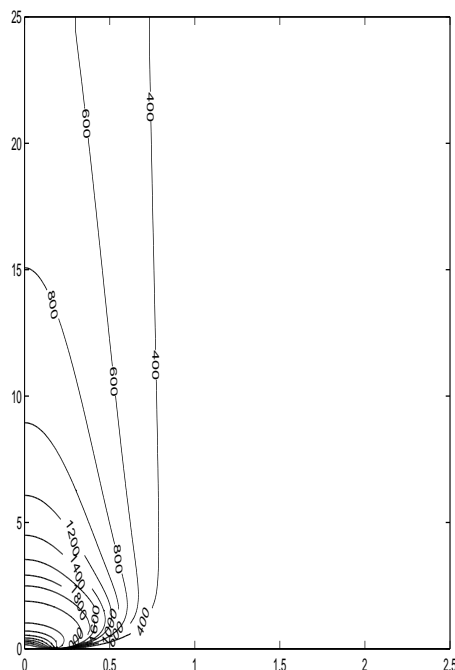


Figure 2: A computed temperature profile of the simulated flame sheet.

- [2] A. Ern, V. Giovangigli, D. E. Keyes and M. D. Smooke, Towards polyalgorithmic linear system solvers for nonlinear elliptic problems. *SIAM J. Sci. Comput.* **15**, 681–703, (1994).
- [3] A. Ern and M. D. Smooke, Vorticity-velocity formulation of three-dimensional steady compressible flows, *J. Comput. Phys.* **105**, 58–71, (1993).
- [4] R. Becker, M. Braack and R. Rannacher, Numerical simulation of laminar flames at low Mach number by adaptive finite elements. *Combust. Theory Modelling*, **3**, 503–534, (1993).
- [5] B. A. Bennett and M. D. Smooke, Local rectangular refinement with application to axisymmetric laminar flames, *Combust. Theory Modelling*, **3**, 503–534, (1993).
- [6] H. C. de Lange and L. P. H. de Goey, Numerical flow modelling in a locally refined grid, *Int J. Numer. Meth. Eng.*, **37**, 497–515, (1994).
- [7] L. M. T. Somers and L. P. H. de Goey, A numerical study of a premixed flame on a slit burner, *Combust. Sci. Technol.*, **108**, 121–132, (1994).
- [8] A. Brandt, Multi-level adaptive solution to boundary-value problems, *Math. Comp.* **31**, 333–390, (1977).

- [9] C. C. Douglas and A. Ern, Numerical solution of flame sheet problems with and without multigrid methods, in *Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods*, N. D. Melson, et al. (ed.), NASA, Langley, VA, 1993, 143–157.
- [10] C. C. Douglas, A. Ern, and M. D. Smooke, Detailed chemistry modeling of laminar diffusion flame on parallel computers. *Int. J. Supercomput. Appl. High. Perf. Comput.* **9**, 167–186, (1995).
- [11] C. C. Douglas, A. Ern, and M. D. Smooke, Multigrid solution of flame sheet problems on serial and parallel computers, *Paral. Alg. Appl.* **10**, 225–236, (1997).
- [12] T. B. Gatski, Review of incompressible fluid flow computations using vorticity-velocity formulation, *Appl. Numer. Math.* **7**, 227–239, (1991).
- [13] Y. Saad and M. H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.* **7**, 856–869, (1986).
- [14] H. A. van der Vorst, A fast and smooth converging variant of Bi-CG for the solution of nonsymmetric linear system, *SIAM J. Sci. Stat. Comput.*, **31**, 631–644, (1992).
- [15] A. R. Curtis, M. J. D. Powell and J. K. Reid, On the estimation of sparse Jacobian matrices. *J. Ind. Math. Appl.*, **17**, 117–122, (1974).
- [16] M. D. Smooke, R. E. Mitchell and D. E. Keyes, Numerical solution of two-dimensional axisymmetric laminar diffusion flames, *Combust. Sci. and Tech.*, **67**, 85–122, (1989).
- [17] J. A. Meijerink and H. A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, *Math. Comp.*, **31**, 148–162, (1977).
- [18] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, New York, NY, (1996).
- [19] J. H. Ferziger and M. Peric, *Computational Methods for Fluids Dynamics*, Springer-Verlag, 2nd Edition, 1999.