NUMERICAL SIMULATION OF LAMINAR DIFFUSION FLAMES

CRAIG C. DOUGLAS*, ALEXANDRE ERN† AND MITCHELL D. SMOOKE‡

Abstract. Not too long ago, anyone wanting to solve large science or engineering problems had to first get access to a supercomputer costing millions of dollars. Quite recently, a new breed of relatively inexpensive work stations became widely available. These machines have scalar peak speeds of 30–275 megaflops with ones on the horizon of 400 or more (which compares rather favorably with vector supercomputers of not so long ago). While these rates are only seen for simple problems like dense matrix–matrix multiplication, the rates seen for many problems are quite high.

In this article, we describe a class of problems which can now be solved on machines individuals can afford to own rather than just on ones costing millions of dollars. Of course, the problem with using a single one of these machines is that the option of connecting a collection of machines together or buying a parallel version of the work station becomes more and more tantalizing.

In fact, during the course of two years we did all of the above. We started on a single machine with a 100 megaflop peak rate (an IBM RISC System/6000 model 560 computer). Then we used a farm of the IBM's, an IBM SP1, and finally an SP2. Due to a nice feature of the communications' library we used (EUIH), the executables worked on the ethernet at Yale or on the fast switches in the SP1/SP2's without either recompiling or relinking.

PROBLEM FORMULATION

Advances in the development of computational algorithms and computer capabilities have provided new, extremely powerful tools with which to investigate chemically reacting systems that were computationally infeasible only a few years ago. Diffusion flames are one such example. When studying the interaction of heat and mass transfer with chemical reactions in commercial burners, gas turbines, and ram jets, these flames are important. When modeling turbulent reacting flows, in improving engine efficiency, and in investigating the processes by which pollutants are formed, the ability to predict the coupled effects of chemical reactions and complex transport phenomena is critical.

Practical combustion systems require a multidimensional study. In the past, the modeling has been done along two independent lines, where either chemistry or fluid dynamics effects are given priority. For many years, most of the detailed chemistry, computational studies were therefore strictly one dimensional configurations: freely propagating or burner stabilized premixed flames and counterflow premixed or diffusion flames. Recently, two dimensional configurations have been common. Three dimensional models combining both fluid dynamical effects with finite rate chemistry will appear shortly, thanks in large part to fast RISC based parallel machines with an adequate amount of memory per node (.125–2 gigabytes).

* IBM Research Division, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598-0218, USA and Department of Computer Science, Yale University, P. O. Box 208285, New Haven, CT 06520-8285, USA.

† Department of Mechanical Engineering, Yale University, P. O. Box 208286, New Haven, CT 06520-8286, USA and CERMICS, ENPC, La Courtine, 93167, Noisy-le-Grand Cedex, FRANCE.

‡ Department of Mechanical Engineering, Yale University, P. O. Box 208286, New Haven, CT 06520-8286, USA.

Here, the formulation and numerical solution of detailed chemistry, two dimensional, axisymmetric laminar diffusion flames in which a cylindrical fuel stream is surrounded by a coflowing oxidizer jet is discussed. A computationally feasible solution is obtained while being able to study the interaction of fluid flow and chemical reaction in the diffusion flames. The full elliptic problem, including up to 50 chemical species in addition to the temperature and the fluid dynamics variables, is treated.

Consider Figure 1. A fuel jet discharges into a laminar air stream. The concentric tubes through which the fuel and oxidizer flow have radii R_I and R_O, respectively. The two gases make contact at the outlet of the inner tube. Unless the flame is lifted (as it is in Figure 1), it resembles a candle once it forms. Since the solution is axisymmetric, the computational domain is only the upper right quadrant of Figure 1. The right and bottom boundaries are the z and r axes, respectively, in Figure 1.

Our model consists of the full set of two dimensional axisymmetric governing equations expressing the conservation of total mass, momentum, energy, and species mass. The dependent variables are the axial and radial velocity components, the temperature, and as many chemical species as considered in the chemical reaction mechanism (typically, 16–50). Boundary conditions are specified for the dependent unknowns at the axis of symmetry ($r = 0$), outer zone ($r = R_{max}$), inlet ($z = 0$), and exit ($z = L$). Several formulations for the fluid dynamics processes have been used to date, i.e., streamfunction–vorticity [7], primitive variables [8], and vorticity–velocity [3]. A brief review of the advantages and disadvantages of each is provided.

In the streamfunction–vorticity formulation, pressure is eliminated as a dependent variable from the momentum equations, the number of equations to be solved is reduced by one, and continuity is explicitly satisfied locally. Despite these three attractions, this formulation presents a severe difficulty in the specification of vorticity boundary conditions. Indeed, a zero vorticity boundary condition at the inlet of the computational domain results in a rough approximation of the true solution. On the other hand, the specification of vorticity boundary values in terms of the streamfunction requires the discretization of second order derivatives, which results in having to solve severely ill conditioned linear systems. The streamfunction–vorticity formulation is not extendible to three-dimensional configurations in a simple form.

The primitive variable formulation allows for accurate boundary conditions and can be used for three-dimensional unsteady problems, but requires a staggered grid arrangement. However, staggered mesh schemes have drawbacks in complex geometric configurations where non-orthogonal curvilinear coordinates are used.

The vorticity–velocity formulation is a relatively new formulation for reacting flows. It eliminates the pressure variable while replacing the first order continuity equation with additional second order equations. Unlike streamfunction–vorticity, vorticity–velocity is easily extendable to three dimensions and allows more accurate formulation of boundary conditions in a numerically compact way. Off diagonal convective terms in the linear systems that exert a strong influence in a streamfunction–vorticity formulation disappear. Another attractive feature of vorticity–velocity is that the governing equations can be discretized on a nonstaggered grid, thus allowing the implementation

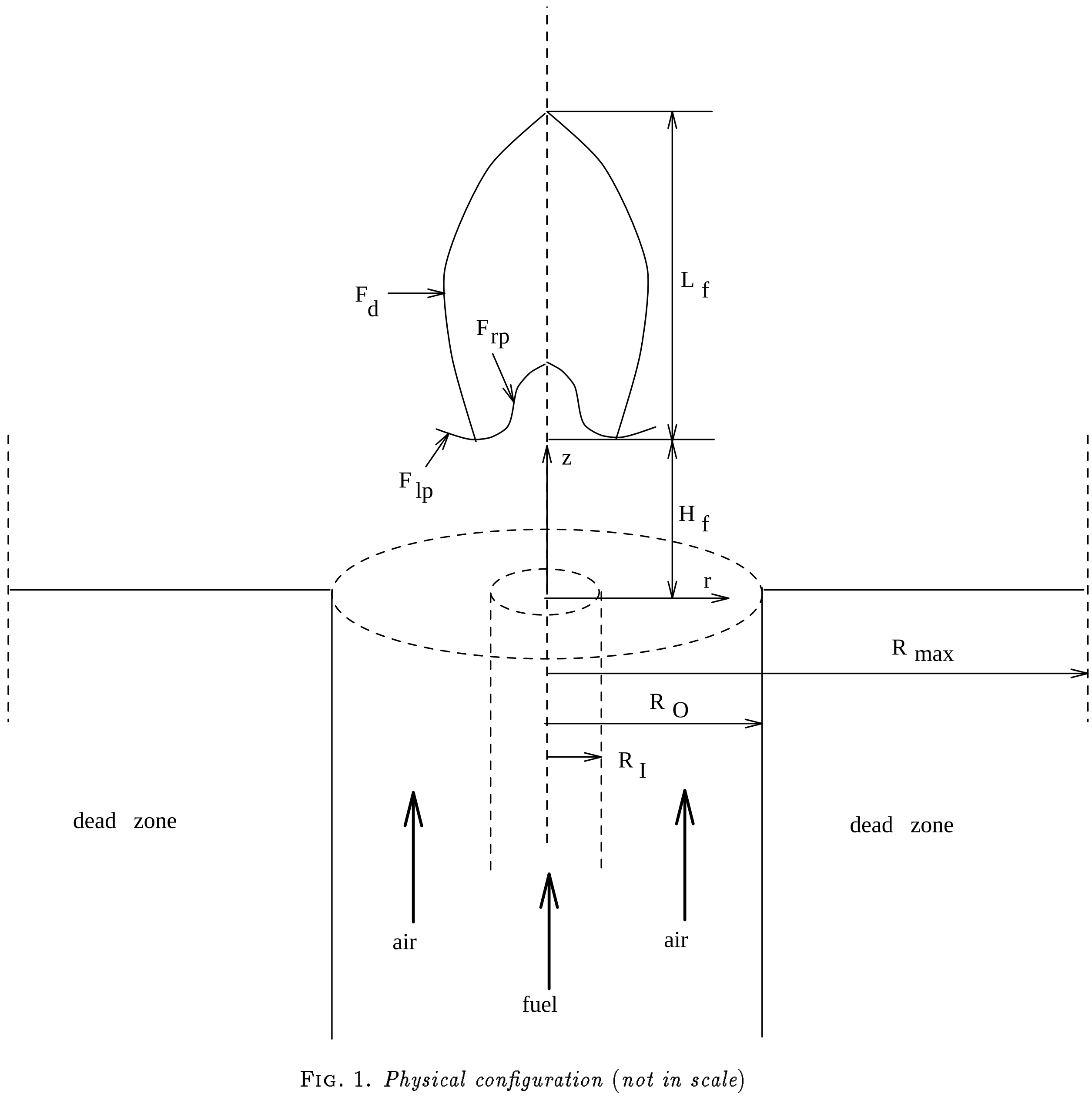

Fig. 1. *Physical configuration (not in scale)*

of a multigrid algorithm easily.

Once the selection of the fluid dynamics model is made, the models that need to be implemented are the evaluation of the chemistry, thermodynamic, and transport properties of the mixture. Such a task is an important and often expensive part of the numerical calculation. A set of data bases and general-purpose subroutines are available. Data in the literature is used for the chemical reaction mechanism, which may include up to 50 chemical species along with 100 elementary reactions. New, rigorously derived, approximate expressions for all the transport coefficients of gas mixtures have become recently available [5]. These expressions constitute an alternative to direct numerical inversions and to empirical averaged expressions for transport property evaluation. For example, a typical chemical reaction is of the form

$$CH_3 + H \rightleftharpoons CH_4$$

with some constants associated with how this is accomplished. The complete 46 reaction table we used is in [4].

Some Details

The present laminar diffusion flame model consists of the full set of two-dimensional, axisymmetric equations expressing the conservation of total mass, momentum, energy, and species mass. As already motivated, we adopt the vorticity-velocity formulation of the Navier-Stokes equations. The vorticity ω is defined in terms of the radial and axial components of the velocity vector $v = (v_r, v_z)$ as follows:

$$\omega = \frac{\partial v_r}{\partial z} - \frac{\partial v_z}{\partial r}. \tag{1}$$

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 (1) and using the continuity equation. A complete description of the diffusion flame governing equations in vorticity–velocity form can be found in [3] so that these equations are simply restated here.

Let us introduce just a little bit of notation. Let ρ denote the mass density of the mixture. Define μ as the shear viscosity of the mixture. Let T be the temperature. The diffusion velocity of the kth species is denoted by V_k and has components in the radial and axial directions given by $V_k = (V_{k_r}, V_{k_z})$. Let g be the gravitational acceleration. For convenience, we define the components of $\overline{\nabla}\beta$ to be $(\frac{\partial}{\partial z}\beta, -\frac{\partial}{\partial r}\beta)$ for any scalar β. Finally, the cylindrical divergence of the velocity vector v is denoted by $\text{div}(v) = \frac{1}{r}\frac{\partial}{\partial r}(rv_r) + \frac{\partial}{\partial z}v_z$. The unknowns are T, v_r, v_z, ω, and the V_k. The diffusion flame governing equations may then be written as follows:

Radial Velocity.

$$\frac{\partial^2 v_r}{\partial r^2} + \frac{\partial^2 v_r}{\partial z^2} = \frac{\partial \omega}{\partial z} - \frac{1}{r}\frac{\partial v_r}{\partial r} + \frac{v_r}{r^2} - \frac{\partial}{\partial r}\left(\frac{v \cdot \nabla \rho}{\rho}\right). \tag{2}$$

Axial Velocity.

$$\frac{\partial^2 v_z}{\partial r^2} + \frac{\partial^2 v_z}{\partial z^2} = -\frac{\partial \omega}{\partial r} - \frac{1}{r}\frac{\partial v_r}{\partial z} - \frac{\partial}{\partial z}\left(\frac{v\cdot\nabla\rho}{\rho}\right). \tag{3}$$

Vorticity.

$$\begin{aligned}\frac{\partial^2 \mu\omega}{\partial r^2} + \frac{\partial^2 \mu\omega}{\partial z^2} + \frac{\partial}{\partial r}\left(\frac{\mu\omega}{r}\right) &= \rho v_r \frac{\partial \omega}{\partial r} + \rho v_z \frac{\partial \omega}{\partial z} - \frac{\rho v_r}{r}\omega + \overline{\nabla}\rho\cdot\nabla\frac{v^2}{2} - \overline{\nabla}\rho\cdot g + \\ & 2\left(\overline{\nabla}(\mathrm{div}(v))\cdot\nabla\mu - \nabla v_r\cdot\overline{\nabla}\frac{\partial\mu}{\partial r} - \nabla v_z\cdot\overline{\nabla}\frac{\partial\mu}{\partial z}\right).\end{aligned} \tag{4}$$

We note that, due to the high temperature gradients present in the flame, no viscosity derivatives in the right-hand side of the vorticity transport equation (4) have been neglected.

In addition, there are equations for the energy and species which can be found in any textbook on the area.

The density is computed from the ideal gas law as a function of the pressure p, the mean molecular weight of the mixture $\overline{W}$, the universal gas constant R, and the absolute temperature T as follows:
Equation of State.

$$\rho = \frac{p\overline{W}}{RT}. \tag{5}$$

Since we only consider low Mach number flames, we can use the constant outlet pressure in (5) in order to compute the density. The pressure field is then eliminated from the governing equations as a dependent unknown. Once a computed numerical solution of (2)–(4) plus the energy and species' equations is obtained, pressure can be recovered by solving a Laplace type equation derived by taking the divergence of the momentum equations.

METHODS OF SOLUTION

A combination of a steady state and a time dependent approach is used computationally. The latter is used to find an approximate solution on a coarse grid using a flame sheet starting estimate.

A discrete solution of the governing equations is computed on the (tensor product) mesh M_2 whose initial nodes are at the intersection of the lines of the mesh

$$M_r = \{0 = r_0 < r_1 < \cdots < r_i < \cdots < r_{Nr} = R_{max}\}$$

and the mesh

$$M_z = \{0 = z_0 < z_1 < \cdots < z_j < \cdots < z_{Nz} = L\}.$$

The grid M_2 is refined using local mesh refinement techniques on M_r and M_z producing a new tensor product mesh M_2. A steady state solution is attempted on the new M_2.

Damped Newton Method

The spatial operators in the governing equations are approximated using a finite difference procedure. Diffusion and convection terms are approximated using central and upwind differences, respectively. An approximate solution to the analytic one is then found at each node of the mesh. With the difference equations written in residual form, a solution U^* is sought to the system of nonlinear equations

$$F(U) = 0, \tag{6}$$

starting from an initial guess U^0. If this is sufficiently close to U^*, then the damped Newton iteration

$$J(U^n)(U^{n+1} - U^n) = -\lambda^n F(U^n), \quad n = 0, 1, \cdots, \tag{7}$$

converges to the correct solution. Here, $J(U^n) = \partial F(U^n)/\partial U$ is the Jacobian matrix and λ^n, $0 < \lambda^n \leq 1$, is the n-th damping parameter.

The Jacobians are effectively nine point operators, but the points are dense square blocks of size equal to the number of components in the calculation. Hence, for a flame calculation with 50 components, the Jacobians have up to 450 nonzeroes per row. Whether or not the Jacobian is still considered a sparse matrix depends on how many spatial points are in the mesh.

Once the Jacobian is formed, the Newton equations (7) are solved using either a preconditioned (Gauss–Seidel) Bi–CGSTAB or GMRES procedure. Rather than working with dimensionless variables, a scale factor is introduced for each dependent variable. The scaling factors are chosen so that the Newton correction $U^{n+1} - U^n$ for each variable are of a similar size to the others of equal importance. The Newton iteration continues until the scaled norm of the discrete vector $U^{n+1} - U^n$ is reduced appropriately. It is worthwhile to point out that an appropriate choice of the scale factors can yield significant savings in the execution time, on the order of a factor of 10.

Adaptive Mesh Multigrid Methodology

The solution of the governing equations contains regions in each coordinate direction in which the dependent variables exhibit high spatial activity (steep front and sharp peaks). In some cases, the solution components can vary by three orders of magnitude between neighboring mesh points. The active regions must be refined adequately. Adaptive techniques that attempt to equidistribute positive weight functions have been used successfully. Flames with over 50 chemical species and 100 chemical reactions can be solved efficiently using this technique [7].

A *nested iteration* multigrid method is used. First, (6) is solved on a coarse mesh. This is refined to get another mesh. The solution from the coarsest mesh is interpolated (using linear or cubic interpolation) onto the new, finer mesh. Then (6) is solved on the new mesh. This can be repeated until there are k meshes and the k-th one is adequately refined to resolve the flame.

The standard approach uses a *one way multigrid* method. This means that the coarser meshes are used only to initialize the next finer ones. By saving the last Jacobian used in the damped Newton iteration (7), *correction problems* can be calculated on the coarser meshes. This can be used to accelerate the iterative solver used in solving (7) [1] [2].

Flame Sheet Starting Estimates

The solution procedure just described requires an adequate starting estimate in order to converge. Determining an estimate that is *good enough* can be challenging. The difficulty is due to the exponential dependence of the chemistry terms on the temperature and to the nonlinear coupling of the hydrodynamic and the thermochemistry solution fields. One approach that has been used for more than a decade (cf. [7]) consists in solving first a flame sheet problem. In the flame sheet model, the fuel and oxidizer are assumed to obey an infinitely fast and irreversible reaction of conversion into stable products in the presence of an inert gas. Such approach provides profiles for each of the major components in the flame, including temperature and major species, and is used to initialize detailed chemistry diffusion flame problems. In our calculations, two dimensional flame sheet models are used in which the vorticity–velocity equations are coupled with a Shvab–Zeldovich equation of the form

$$\frac{1}{r}\frac{\partial}{\partial r}(r\rho D\frac{\partial S}{\partial r}) + \frac{\partial}{\partial z}(\rho D\frac{\partial S}{\partial z}) = \rho v_r\frac{\partial S}{\partial r} + \rho v_z\frac{\partial S}{\partial z}.$$

Here, S is a conserved scalar and D is a diffusion constant. The temperature and major species profiles are recovered from the conserved scalar S.

Time Relaxation

The adaptive mesh multigrid methodology and the flame sheet starting estimate help eliminate some of the convergence difficulties associated with solving the governing equations directly [7]. Nevertheless, neither the interpolated solution from one mesh to the next finer one nor the flame sheet starting estimate generally lie in the convergence domain of (7). Therefore, a parabolic in time problem is produced by appending a scaled pseudo transient term $D_{scale}\partial U/\partial t$ to the left hand side of the conservation equations. The time derivative is replaced by a backward Euler approximation [7]. At each time step, a system of nonlinear equations is solved using again a damped Newton iteration. This new problem is quite similar to (6); in fact, the major difference is that the diagonal of the steady state Jacobian is weighted by the reciprocal of the time step. After an appropriate amount of time steps, a switch over to the steady-state form of the equations becomes possible.

Parallel Computing Methodology

Reasonable solvers for this class of problems separate the chemistry parts of the code from the algebra parts. Parallelizing this at first appeared to require a complete rewrite of a few thousand lines of code. Luckily, it no longer needs to be done. This is due to the large amount of memory per node on recently delivered parallel computers. Similar remarks hold for workstations used in cluster configurations for distributed computing.

The Jacobian matrices have a regular block sparse structure which we exploited in our parallelization strategy. Additionally, we wanted our parallel code to operate similarly to our serial code.

The four principal sets of operations we needed to parallelize were the following:

1. Jacobian matrix construction
2. Matrix–vector multiplication
3. Nonlinear residual evaluation
4. Inner products and norms (i.e., simple level 1 BLAS)

The vast majority of the serial computer time is spent in 1–3.

The Jacobians have the block structure in Figure 2. Each block has a similar block structure with 3 subblocks. The subblocks are $n_c \times n_c$ and dense. Hence, the Jacobians resemble 9 point operators, but have $9n_c$ nonzeroes per row on average. Assuming $n_c \in [16, 50]$, the Jacobians use a considerable amount of memory in comparison to the solution vector.

We decided to use a sparse matrix domain decomposition method. By this, we mean that we considered the Jacobian matrix to be a two dimensional domain, and decomposed that by rows of blocks. This corresponds to a strip domain decomposition method. For example, in Figure 2, we decomposed every 4 rows of blocks. Hence, in doing a matrix–vector multiply, only the $n_c{\cdot}n_r$ unknowns associated with end blocks must be transferred between processors.

We considered two other domain decomposition methods. Schur complement methods simply use too much extra memory computing the complement of a matrix with $9n_c$, $16 \le n_c \le 50$ nonzeroes per average row. Alternating methods on two-dimensional subdomains, additive or multiplicative, have two problems: First, the gradients of the solutions along the internal subdomain boundaries are quite large. Second, standard theory based on condition numbers indicate that a huge number of iterations of the alternating procedures would be required to solve our problems.

We have two matrix–vector multiplication routines. One which has the preconditioner prefactored into its operations and one which does not. The parallelization of each follows the techniques just described. The preconditioner is local to each processor, not global. Hence, the total number of iterations of our parallel solvers is not always the same as the serial computer equivalents.

The nonlinear residual calculation uses the same vector decomposition used in the matrix–vector and Jacobian routines. Hence, no data needs to be transferred between processors to start this procedure.

One aspect of the Jacobians having so many nonzeroes is that we can afford to store the complete solution vector on a single processor without using much extra memory. In our case, our parallel computers had from 128 Mb to 2 Gb of local memory on each node. Hence, if some operation is quick and does not parallelize well (or at all), we can actually gather all of the data, do the operation on a single node, and then scatter data back to the remaining processors. This was particularly useful while parallelizing the code one major operation at a time and for debugging purposes.

The general computing methodology used was that there was one processor that directed all of the processors to do various computing tasks. These included having processors do their part of the parallel iterative procedures, compute a Jacobian, or evaluate a nonlinear residual. Large tasks are the typical operation, not small tasks like requiring all processors to do an inner product.

The communications library we used, EUIH [6], allowed us to produce one executable for all of the following IBM computers: SP1, SP2, and clusters of IBM RS-6000's. No relinking or recompiling was necessary in order to use the fast switches. Hence, only one copy of the source files and one executable were maintained for all of these environments.

NUMERICAL EXPERIMENTS

The flame sheet model provides a test problem of a moderate computational cost with which to probe the efficiency of solution algorithms that can then be used to tackle detailed chemistry diffusion problems. Such a study is presented in [2] [3]. The flame sheet that is considered is adequately resolved on a grid with up to 2×10^4 nodes, with 20% of the mesh nodes clustered in a region covering 0.1% of the computational domain. This requires up to 100 megabytes of work space. The adaptive mesh multigrid procedure outlined above is particularly efficient for solving flame sheet problems. In comparison to traditional solution procedures, the total execution times drop by a factor of 10. Speedups as high as 166 in the time relaxation phase have been seen on a single processor. For three dimensional problems, speedups much greater than 10 should be obtained. At the beginning of this study, we spent as much as 96 minutes on the flame sheet phase using an IBM RISC System/6000 model 560 workstation. Recently, we spent as little as 45 seconds using an eight node SP2.

Solving a finite rate chemistry flame problem instead of a flame sheet problem is a quite challenging step which involves a dramatic increase in the difficulty of the problem. This is mostly attributable to the much larger number of dependent unknowns, the nonlinear fluid dynamics-thermochemistry coupling, and the disparate length scales that must be resolved in the computed solution. In particular, excellent resolution of a flame sheet problem requires up to 128 megabytes, while a reasonable solution of a diffusion flame with 50 chemical species may require up to a gigabyte. Traditional solvers for two-dimensional laminar diffusion flames used to require more than a hundred CPU hours on a supercomputer to get a solution. Thanks to recent developments of computational algorithms and computer capabilities, this is no longer the case.

The flame configuration is a methane-air lifted laminar diffusion flame with a triple flame structure at its base, as illustrated in Figure 1. Although extremely difficult to compute numerically, this flame configuration is chosen in the present study since experimental and numerical results are already available for these flames [8]. The numerical solution that is considered first includes 16 chemical species engaged in a C1-chain re-

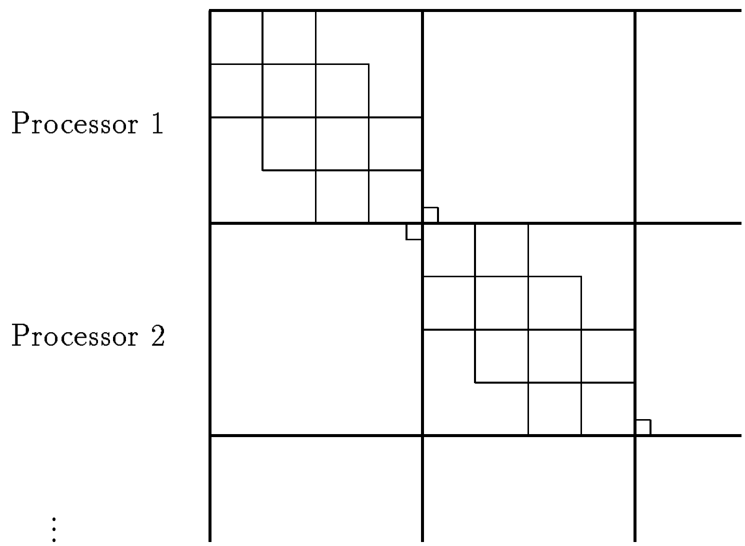

FIG. 2. *Upper left corner of the Jacobian matrix block structure*

action mechanism, i.e., only molecules with at most *one* carbon atom are considered. In addition to the 16 chemical species, 4 unknowns are associated with each mesh point. Hence, there are a total of 20 unknowns per mesh point. The flame is appropriately resolved using 5×10^3 mesh nodes, and very good agreement with previous experimental (2.5×10^5 data points) and numerical data is obtained.

Consider a one way nonlinear multigrid with Bi–CGSTAB/GS as the solver on all levels. On the finest level, these are the (wall clock) times we see to converge on the 89×85 fine grid.

Machine	Processors	Minutes
RISC System/6000–580	1	602.59
SP1	4	137.21
	8	70.37
	12	58.89
	16	53.85
SP2	4	47.41
	6	30.05

Peak floating point is about 130 Mflops for the SP1 and RISC System/6000–580, but about 266 for the SP2.

The chemical mechanism just considered yields adequate resolution for the temperature and several chemical species. More detailed investigation of the flame structure requires additional chemical species in the model. For instance, the study of pollutant formation, such as nitric oxide, requires up to 50 chemical species. While the memory requirements are too large for current serial computers, such problems can be adequately solved on parallel computers with a large amount of memory per node. The diffusion flame just described can then be conveniently used to initialize diffusion flame calculations with more detailed chemistry submodels. Such work is in progress.

REFERENCES

[1] C. C. Douglas, *Implementing abstract multigrid or multilevel methods*, in Sixth Copper Mountain Conference on Multigrid Methods, N. D. Melson, T. A. Manteuffel, and S. F. McCormick, eds., vol. CP 3224, Hampton, VA, 1993, NASA, pp. 127–141.

[2] C. C. Douglas and A. Ern, *Numerical solution of flame sheet problems with and without multigrid methods*, in Sixth Copper Mountain Conference on Multigrid Methods, N. D. Melson, T. A. Manteuffel, and S. F. McCormick, eds., vol. CP 3224, Hampton, VA, 1993, NASA, pp. 143–157.

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

[4] A. Ern, C. C. Douglas, and M. D. Smooke, *Detailed chemistry modeling of laminar diffusion flames on parallel computers*. In preparation, 1994.

[5] A. Ern and V. Giovangigli, *Multicomponent Transport Algorithms*, vol. m 24 of New series monographs, Springer–Verlag, Heidelberg, 1994.

[6] P. Hochschild, *EUIH: An experimental EUI implementation*. Unpublished report, IBM Reseach Division, P. O. Box 218, Yorktown Heights, NY 10598, 1993.

[7] M. D. Smooke, R. E. Mitchell, and D. E. Keyes, *Numerical solution of two-dimensional axisymmetric laminar diffusion flames*, Combust. Sci. and Tech., 67 (1989), pp. 85–122.

[8] Y. Xu, M. D. Smooke, P. Lin, and M. B. Long, *Primitive variable modeling of multidimensional laminar flames*, Combust. Sci. and Tech., 90 (1993), pp. 289–313.