Numerical simulation of unsteady premixed laminar flames

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

RANA 97-06
July 1997

Numerical Simulation of Unsteady Premixed Laminar Flames

by

B. van 't Hof

Reports on Applied and Numerical Analysis
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
The Netherlands
ISSN: 0926-4507

Numerical Simulation of Unsteady Premixed Laminar Flames

B. van 't Hof

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract

Recently, a computer code for the numerical simulation of two-dimensional, steady laminar flames called LAMFLA2D was modified to allow for unsteady calculations. This paper describes why and how this was done and in what direction this research may continue.

1 Introduction: Purpose of the Study.

The natural gas-powered appliance industries, like any other industries, have to meet increasingly strict legal (environmental) requirements and customer expectations. Technology has to make a constant progress in order to keep up with these ever-increasing demands. Apart from experimental design, computer simulations have started to play a certain role in this effort, though possibilities are still rather modest as the amount of computational work involved with gas-flame simulation is immense. The steady increase of modern computers' power has enabled more and better simulations to be performed, and will continue to do so, turning computer simulation into a serious design tool. Present day research and programming effort should therefore always be done with the possibilities in mind that the near future will bring.

At Eindhoven University of Technology, the Department of Mechanical Engineering has developed several computer codes for gas-flame simulation. These include CHEM1D [1, 3], a code for the simulation of steady and unsteady one-dimensional laminar methane/air flames with a detailed description of the chemistry, CHEM2D, the two-dimensional equivalent of CHEM1D, which can, however, only handle the nonlinear steady state equations, and LAMFLA2D [4], a code very similar to CHEM2D, but used for the solution of laminar flames described by one-step chemistry. This paper describes how the LAMFLA2D code was modified to allow for simulations of unsteady flame behavior. When decisions are taken concerning the approach to such a modification, the kind of simulations the code is expected to do *now*, as well as what it will be expected to do in the (near) future have to be taken into account.

A code for unsteady simulations is, of course, meant to give insight into time-dependent behavior of the phenomenon studied. There are, however, different types of time-dependent behavior. It can be interesting to investigate the *stability* of steady solutions, a study for which only a very short time span needs to be simulated with very small time steps. Another type of time-dependent behavior is the oscillatory behavior a flame will display when fuel supply is insufficient, known as *quenching oscillations* [10]. Tests with CHEM1D have indicated that this type of behavior can be adequately studied by time span simulations of approximately one second, where the typical time step can be in the range of tenths of milliseconds for accuracy (first-order time integration). A third type of time-dependent behavior which will be studied with the new code is the *blow-off* phenomenon which occurs when gas supplies are too large for a stationary flame to develop. This type of behavior requires roughly the same simulated time span and time step that the quenching oscillations do. A fourth use of an unsteady code is to use time-stepping as a solver for the steady equations. A similar time span as for quenching oscillations and for blow-off behavior needs to be simulated for this purpose, but a larger time-step can be used if the code can do so without losing stability.

Initially, therefore, it is our goal to construct a code that can calculate several thousands of time steps on a fine enough grid for a combustion model that is so detailed that it can capture the essential behavior of a flame accurately enough. Required computing power for one simulation should not exceed, say, 16 hours on a Silicon Graphics workstation. Since this is a rather ambitious task, it is necessary to use a careful approach to building the code. We applied the computational techniques that best seemed to suit our needs rather than develop a lot ourselves. This lead us to believe that the multi grid approach as described in [2], with some minor modifications, should yield a powerful and dependable solver.

One of the extensions to the code which will be made soon is allowing a more complex chemical reaction scheme: from the 5 chemical species included in the current code we will include around 20. This is the reason for the focus on matrix-free solving techniques, because the matrices involved in linearizing the flame equations will need up to 20 or 30 Kbytes of storage per grid point for such a detailed problem, and this will definitely cause memory problems for today's computers. Another possible extension is to refine the physics of the problem to include sound waves.

The next section is devoted to the equations which are used to mathematically describe the mechanisms occurring in gas flames. In Section 3, we will describe the discretization of these equations. Section 4 describes the solvers used. Last, results will be shown and conclusions drawn in Section 5.

2 Mathematical Model

A laminar, two-dimensional flame is described in terms of the following variables:

$Y_i, i = 1, \cdots, n$	mass fraction of the chemical species $1, \cdots, n$,
T	temperature,
p	pressure,
ρ	mass density,
$v = (u, v)$	velocity vector.

The governing equations used by the LAMFLA2D code are the following [3].
Mass conservation for chemical species:

$$\frac{\partial \rho Y_i}{\partial t} + \frac{\partial}{\partial x}\left(\rho u Y_i - \frac{1}{Le_i}\frac{\lambda}{c_p}\frac{\partial Y_i}{\partial x}\right) + \frac{\partial}{\partial y}\left(\rho v Y_i - \frac{1}{Le_i}\frac{\lambda}{c_p}\frac{\partial Y_i}{\partial y}\right) = s_i, \tag{2.1}$$

Energy equation:

$$\frac{\partial \rho T}{\partial t} + \frac{\partial}{\partial x}\left(\rho u T - \frac{\lambda}{c_p}\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\rho v T - \frac{\lambda}{c_p}\frac{\partial T}{\partial y}\right) = s_T, \tag{2.2}$$

Equation of Motion:

$$\begin{aligned} &\frac{\partial \rho u}{\partial t} + \frac{\partial}{\partial x}\left(p + \rho u^2 - \mu\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{\partial v}{\partial y}\right)\right) + \frac{\partial}{\partial y}\left(\rho u v - \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right) = 0, \\ &\frac{\partial \rho v}{\partial t} + \frac{\partial}{\partial x}\left(\rho u v - \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right) + \frac{\partial}{\partial y}\left(p + gy\rho + \rho v^2 - \mu\left(\frac{4}{3}\frac{\partial v}{\partial y} - \frac{2}{3}\frac{\partial u}{\partial x}\right)\right) = 0, \end{aligned} \tag{2.3}$$

Continuity Equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0, \tag{2.4}$$

Equation of State:

$$p_0 = \frac{\rho R T}{M}. \tag{2.5}$$

The *chemical source term* s_T *for the temperature* and the *average molecular mass* M of the gas mixture are given by

$$s_T = -\frac{1}{c_p}\sum_{i=1}^{n} h_i s_i \quad \text{and} \quad \frac{1}{M} = \sum_{i=1}^{n} \frac{Y_i}{M_i},$$

respectively.

The model is complete when expressions for

$Le_i, i = 1, \cdots, n$	Lewis numbers for the chemical species,
$s_i, i = 1, \cdots, n$	source terms for the chemical species,
$h_i, i = 1, \cdots, n$	enthalpy of the chemical species,
$M_i, i = 1, \cdots, n$	molecular mass of the chemical species,
λ	heat conductivity coefficient,
c_p	specific heat of the gas mixture,
μ	viscosity coefficient,
R	universal gas constant,
g	gravity constant,
p_0	atmospheric pressure

are given. These coefficients have been studied and measured by various scientists and more or less accurate estimates are available [5]. Much more complicated models than the one described here are available, especially where the energy equation and the conservation equations for chemical species are concerned. Smooke [6] and Somers [3], among others, use the energy equation (2.2). Equation (2.1) is obtained using what is called the generalized law of Fick [11].

These equations can of course only be solved if they are supplied with initial conditions and with boundary conditions. To see what kind of boundary conditions have been used, Figure 1 shows what the computational domain looks like and what types of boundaries there are.

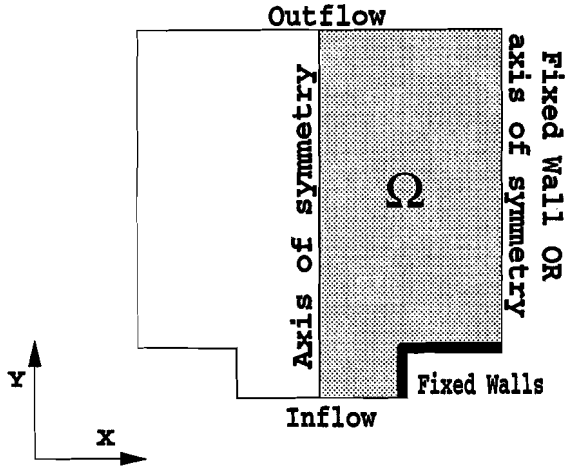

Figure 1: Computational Domain and Boundaries.

There are the following type of boundaries.

Inflow Boundary	$v = v_0(x),$	$u = 0,$	$Y_i = Y_{i0},$	$T = T_0,$
Outflow Boundary	$\frac{\partial v}{\partial y} = 0,$	$u = 0,$	$\frac{\partial Y_i}{\partial y} = 0,$	$\frac{\partial T}{\partial y} = 0,$
Symmetry Axis	$\frac{\partial v}{\partial x} = 0,$	$u = 0,$	$\frac{\partial Y_i}{\partial x} = 0,$	$\frac{\partial T}{\partial x} = 0,$
Cooled Fixed Wall	$v = 0,$	$u = 0,$	$\frac{\partial Y_i}{\partial n} = 0,$	$T = T_0,$
Insulated Fixed Wall	$v = 0,$	$u = 0,$	$\frac{\partial Y_i}{\partial n} = 0,$	$\frac{\partial T}{\partial n} = 0.$

With these boundary conditions given and initial conditions given also, the system can be solved, though the pressure p is only determined up to a constant.

This concludes the discussion of the model. However, it is possible to eliminate several equa-

tions. Obviously, using (2.5), one can eliminate ρ from the system. Then, we can eliminate one species' mass fraction from the system using the equation

$$\sum_{i=1}^{n} Y_i = 1.$$

Best results are obtained if a bulk species is eliminated in this way: typically, this is N_2. More equations can be eliminated by the idea of *Element Conservation*, which is well explained by Somers in [3].

Doing so, we are left with the *Equation of Motion* (2.3), the *Continuity Equation* (2.4), the *Energy Equation* (2.2) and the *Mass Conservation Laws* (2.1) for a reduced number (which we will call n_s) of chemical species. How these are to be discretized in space and time is described in the next section.

3 Discretization of the Equations

The discretization was done through the so-called *method of lines*, which means that space discretization and time integration have been studied separately. This section will therefore consist of two parts: one about the space discretization, the other about the time integration.

3.1 Space Discretization

Before looking at the discretized equations, let us have a look at the grid. We use a standard staggered grid. Figures 2 and 3 show what the grid looks like and the notations used to denote grid points.

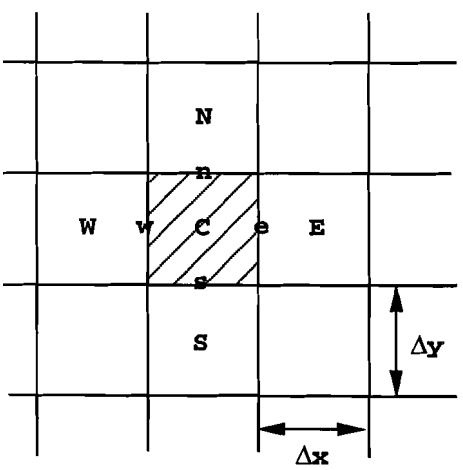

Figure 2: Grid around the point C.

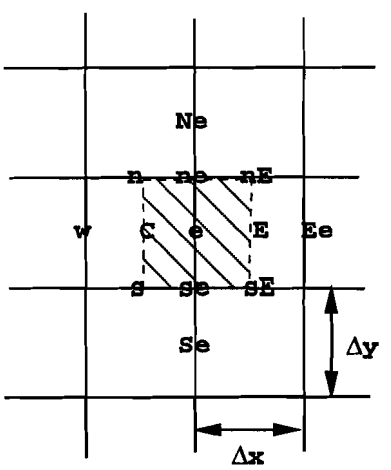

Figure 3: Grid around the point e.

All the variables, except the velocity components u and v, are approximated at the *cell-centers*, i.e. grid points denoted by capital letters C, E etcetera. The horizontal velocity component u is approximated at the *vertical cell-faces*: grid points with names like e, w and Ee. The vertical velocity component v, at last, is approximated at the horizontal cell-faces: grid points with names

like n, s nE, sE. The equations (2.1), (2.2), (2.3) and (2.4) all have the same generic form

$$\frac{\partial \rho\phi}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = s, \tag{3.1}$$

where F and G are the horizontal and vertical components of some flux vector. Note that, for the continuity equation, $\phi = 1$. All the equations except the equations of motion are approximated at the cell-centers. Therefore, all the conserved quantities (i.e. ρT, ρY_i, ρ) except ρu and ρv, as well as the pressure p are approximated at the cell-centers, whereas their fluxes are approximated at the cell-faces. The approximation of (3.1) then reads:

$$\frac{\partial \rho_C \phi_C}{\partial t} + \frac{F_e - F_w}{\Delta x} + \frac{G_n - G_s}{\Delta y} = s_C,$$

where, for any variable H and grid point g, H_g denotes the approximation of H at grid point g. For all these equations, the fluxes F and G have the following form:

$$F = \rho u\phi - \Gamma\frac{\partial \phi}{\partial x}, \qquad\qquad G = \rho v\phi - \Gamma\frac{\partial \phi}{\partial y},$$

though for the continuity equation, $\Gamma = 0$. The fluxes F and G are then approximated by central differences:

$$\begin{aligned} F_e &= \frac{\rho_C + \rho_E}{2} u_e \frac{\phi_C + \phi_E}{2} - \frac{\Gamma_C + \Gamma_E}{2}\frac{\phi_E - \phi_C}{\Delta x} \quad , \\ G_n &= \frac{\rho_C + \rho_N}{2} v_n \frac{\phi_C + \phi_N}{2} - \frac{\Gamma_C + \Gamma_N}{2}\frac{\phi_N - \phi_C}{\Delta y}. \end{aligned}$$

This concludes the description of the space discretization of all equations except the equation of motion.

The horizontal equation of motion is approximated at the vertical cell-faces. This approximation reads

$$\frac{1}{2}\frac{\partial(\rho_C + \rho_E)u_e}{\partial t} + \frac{F_E - F_C}{\Delta x} + \frac{G_{ne} - G_{se}}{\Delta y} = 0,$$

where the fluxes are given by

$$F = p + \rho u^2 - \mu\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{\partial v}{\partial y}\right) \quad , \quad G = \rho uv - \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right),$$

and approximated by

$$\begin{aligned} F_C &= p_C + \rho_C\left(\frac{u_e + u_w}{2}\right)^2 - \mu_C\left(\frac{4}{3}\frac{u_e - u_w}{\Delta x} - \frac{2}{3}\frac{v_n - v_s}{\Delta y}\right), \\ G_{ne} &= \frac{\rho_C + \rho_N + \rho_E + \rho_{NE}}{4}\frac{u_e + u_{Ne}}{2}\frac{v_n + v_{nE}}{2} - \\ &\quad \frac{\mu_C + \mu_N + \mu_E + \mu_{NE}}{4}\left(\frac{u_{Ne} - u_e}{\Delta y} + \frac{v_{nE} - v_n}{\Delta x}\right). \end{aligned}$$

The vertical equation of motion is approximated in a totally analogous way.

Note that this space-discretization is a standard, central-difference based finite volume discretization on a standard staggered grid. Large local Peclet numbers or cell-Reynolds numbers may cause non-physical wiggles in the solution. This problem can be dealt with by adding just so much artificial (stream line) diffusion that the well-known exponential, Patankar [7], or Il'in discretization schemes are obtained.

This discretization can be easily generalized to rectangular grids with variable Δx and Δy ('tensor-product grids').

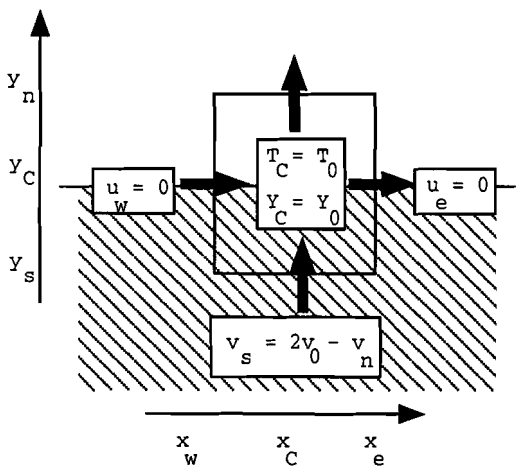

Figure 4: Discrete boundary conditions at the inflow boundary

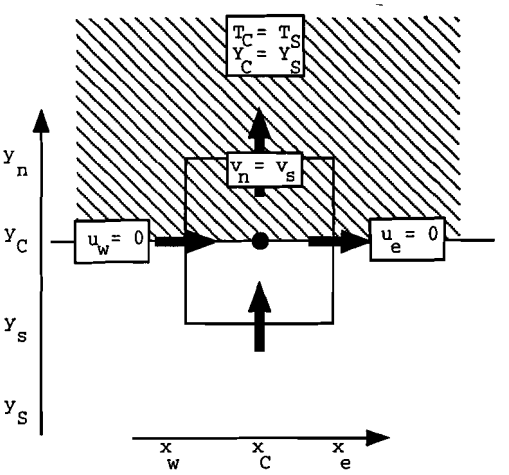

Figure 5: Discrete boundary conditions at the outflow boundary.

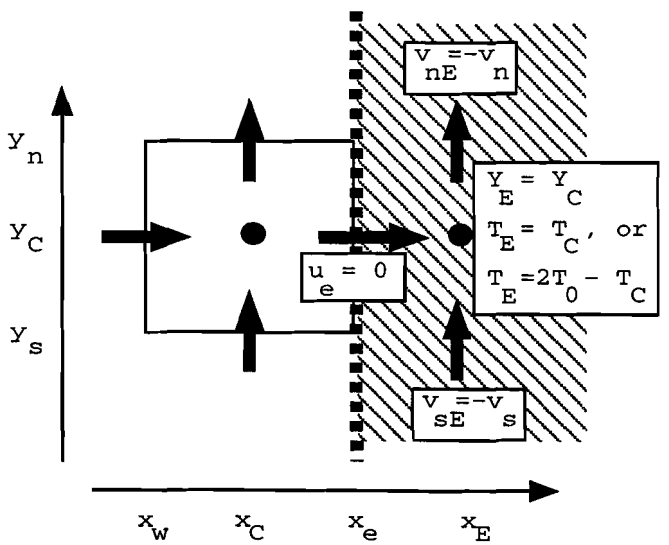

Figure 6: Discrete boundary conditions at a fixed wall.

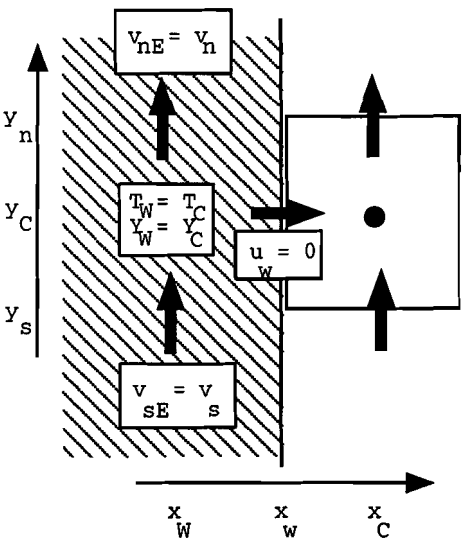

Figure 7: Discrete boundary conditions at a symmetry boundary.

Dirichlet boundary conditions are trivially discretized for grid points on the boundary. All other boundary conditions are discretized by means of so-called virtual grid points: a grid point is defined outside the domain, and an equation is derived from the boundary conditions. A Dirichlet boundary condition

$$\phi|_{\delta\Omega} = \phi_0$$

then becomes

$$\phi_{virtual} = 2\phi_0 - \phi_C,$$

and a homogeneous Neumann boundary condition becomes

$$\phi_{virtual} = \phi_C.$$

This is, again, a standard procedure. The resulting discrete boundary equations are shown in detail by the Figures 4, 5, 6 and 7.

The semi-discrete equations now have to be integrated in time in order to obtain numerical results.

3.2 Time Integration

For the time integration of the semi-discretized equations, we have to make a choice between explicit and implicit time integrators, or we have to use a partly implicit, partly explicit time integrator. In the equations we study, a number of effects play a role. Let us make an estimation what the maximum time step will be for an explicit time integrator for all of these terms.

First, we have chemical reactions. Maas and Pope [12] found that chemical reactions in premixed methane/air combustion happen at time scales ranging from 10^{-2} to 10^{-10} seconds. Therefore, unless we are willing to take millions of time steps, chemical source terms have to be taken implicitly.

Next, there are diffusive and viscous effects. The following rule of thumb gives clear insight in the maximum time step allowed:

$$\Delta t_{max} = \mathcal{O}\left(\frac{\Gamma}{\Delta x^2} + \frac{\Gamma}{\Delta y^2}\right),$$

in which Γ stands for the diffusion or viscosity coefficient. In typical problems, this maximum time step will be in the order of 10^{-6} to 10^{-8}, so again an implicit approach is necessary.

Then, there are convective phenomena. Here, we have the following rule of thumb:

$$\Delta t_{max} = \mathcal{O}\left(\frac{|u|}{\Delta x} + \frac{|v|}{\Delta y}\right),$$

which indicates that Δt_{max} will be something in the order 10^{-5} to 10^{-6} seconds. Again, these are time step sizes we are not very happy to be held to.

Last, there are pressure wave effects. The propagation velocity of these waves depends on the Mach number which, because of the special form of the equation of state (2.5), is zero. This implies that a fully explicit time stepping algorithm is doomed unless we introduce some sort of artificial compressibility. Many pressure correction algorithms have been developed to minimize the number of implicit terms.

From these considerations we may conclude that hundreds of thousands up to billions of time steps have to be taken for one second of simulated time, unless a fully implicit time stepping algorithm is used (Euler Backward, BDF2). We chose the Euler Backward time integration. This required us to have a very powerful solver. We think we have found this in the Multi grid approach due to [2], as described in Section 4.

4 Matrix Free Nonlinear Solvers

Many solvers for nonlinear equations are based on Newton iteration: an (approximate) linearization of the system is built, after which the resulting linear system is solved (approximately). We are, however, dealing with a very large system of nonlinear equations. Also, in any of these equations, many variables occur. A linearization matrix will consequently not be very sparse, and will require (too) much computer memory.

4.1 Broyden Iteration

A method which avoids the numerical approximation of a linearization matrix is Broyden iteration [13]. It does not, however, avoid working with a full matrix, so it is not suited for very large systems (though sparse modifications exist). For non-sparse problems, the advantage of Broyden Iteration as compared to a modified Newton iteration with a numerically approximated linearization matrix can be summarized by stating that with the computational work necessary to *start* the Newton process, one can do as many Broyden iterations as the dimension of the system.

Broyden iteration works in the following way for the solution of the system

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{o}.$$

- choose a cheap, crude approximation $\boldsymbol{B}_0$ of the inverse linearization matrix $\boldsymbol{J}_f^{-1}$.
- choose an initial approximation $\boldsymbol{x}_0$.
- repeat for $i = 0, 1, \cdots$:
  - Calculate the update $\delta\boldsymbol{x}_i := \boldsymbol{B}_i\boldsymbol{f}(\boldsymbol{x}_i) =: \boldsymbol{B}_i\boldsymbol{f}_i$, just like you would do if you were doing Newton iteration and $\boldsymbol{B}_i$ were equal to the $\boldsymbol{J}_f^{-1}$.
  - Find a relaxation parameter λ_i such that

    $$\|\boldsymbol{f}(\boldsymbol{x}_i - \lambda_i\delta\boldsymbol{x}_i)\| < \|\boldsymbol{f}_i\|.$$

    Update the solution

    $$\boldsymbol{x}_{i+1} := \boldsymbol{x}_i - \delta\boldsymbol{x}_i.$$

  - to construct a matrix $\boldsymbol{B}_{i+1}$ which is a better approximation of the inverse linearization matrix, we set

    $$\boldsymbol{f}_{i+1} - \boldsymbol{f}_i = \boldsymbol{B}_{i+1}^{-1}(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i),$$

    or, equivalently,

    $$\boldsymbol{B}_{i+1}(\boldsymbol{f}_i - \boldsymbol{f}_{i+1}) = \lambda_i\delta\boldsymbol{x}_i.$$

    This is an under determined system, so we have many options in constructing $\boldsymbol{B}_{i+1}$. The following choices are widely used:

    * $\boldsymbol{B}_{i+1} := \boldsymbol{B}_i\left(\boldsymbol{I} + \frac{(\lambda_i-1)\boldsymbol{f}_i+\boldsymbol{f}_{i+1}}{\|\boldsymbol{f}_i-\boldsymbol{f}_{i+1}\|^2}(\boldsymbol{f}_i - \boldsymbol{f}_{i+1})^T\right)$, because it minimizes $\|\boldsymbol{B}_{i+1} - \boldsymbol{B}_i\|_2$
    * $\boldsymbol{B}_{i+1} := \left(\boldsymbol{I} + \boldsymbol{B}_i\frac{(\lambda_i-1)\boldsymbol{f}_i+\boldsymbol{f}_{i+1}}{\delta\boldsymbol{x}_i^T\boldsymbol{B}_i(\boldsymbol{f}_{i+1}-\boldsymbol{f}_i)}\delta\boldsymbol{x}_i^T\right)\boldsymbol{B}_i$, because it minimizes $\|\boldsymbol{B}_{i+1}^{-1} - \boldsymbol{B}_i^{-1}\|_2$.
- until $\|\boldsymbol{f}_i\| < tol$.

4.2 Symmetrically Coupled Gauss Seidel (SCGS) Iteration

We are dealing with a very large system of nonlinear equations. It is impossible to solve all of these equations at once. However, if we take a small number of these equations, and the same number of variables, it is possible to solve this subset by changing the selected variables only, if the equations and variables are chosen wisely.

In Symmetrically Coupled Gauss Seidel (SCGS) Iteration, we associate equations and variables with grid cells (see Figure 2). These equations and variables are:

- $Y_1, \cdots, Y_{n_s}$ at point C, with equation (2.1), integrated around the point C,
- T at point C with equation (2.2), integrated around the point C,
- p at point C with equation (2.4), integrated around the point C,
- u at points e and w with equation (2.3), integrated around those points,
- v at points n and s with equation (2.3), integrated around those points.

The resulting system in point C is then a system of $n_s + 6$ variables and equations. Let us write this equation formally as

$$f(\phi_C) = \mathbf{o}.$$

We chose Broyden iteration to solve this system.

In this way, the equations associated with one grid cell can be solved changing only the variables associated with the same cell. After the variables associated with one grid cell have been updated in this way, the variables of the next grid cell can be updated and so on, until all the grid cells have been visited once. This we call one SCGS-sweep. The SCGS solver simply makes SCGS-sweeps until the equations are solved accurately enough.

SCGS Iteration is not a very efficient solver. In the following subsections it will be discussed how it can be accelerated.

4.3 Local Re-Iteration

When SCGS Iteration is applied, it is found that the residuals vanish much more quickly in most of the domain, and only remain in a small number of grid points which we call the reaction zone. This is the zone where chemical source terms are dominant. It makes sense to iterate more in this small area where the equations appear to be more difficult than elsewhere. This is done in the following way. First, the difficult points are located. These are defined as those points, where the norm of the residual exceeds a certain multiple of the average residual norm. The points at which the residual is large after iteration number n are then denoted by $\{d_1^n, \cdots, d_{m(n)}^n\}$. An SCGS-sweep can be made more effective by first visiting the difficult points a number of times (e.g. 3 times), and then the rest.

4.4 Multi grid

There are many ways in which multi grid can be used to solve equations efficiently. An overview is given in [9]. The specific form in which our solver is put is very close to the approach of [2] and is discussed in this subsection.

Assume that the set of equations which have to be solved every time step is

$$\frac{\phi - \phi^{n-1}}{\Delta t} + \boldsymbol{F}_h(\phi) = \mathbf{o}. \tag{4.1}$$

In order to find ϕ, we apply a number of (Locally Re-iterated) SCGS-sweeps. We call this number of sweeps k. We then find the approximation ϕ_k, for which

$$\frac{\phi_k - \phi^{n-1}}{\Delta t} + \boldsymbol{F}_h(\phi_k) = \boldsymbol{r}_k. \tag{4.2}$$

It is interesting to investigate how the residual r_k depends on k, the number of SCGS sweeps. Since SCGS is a solver, the residual will be small if k is large. However, since SCGS is an inefficient smoother, this will only be so if k is very large, and it is not good to wait that long. For small k, r_k will not be substantially smaller than r_0. It will, however, be substantially *smoother*. What is meant by this is illustrated by Figure 8. Figure 8 shows the plot of what could be r_0 on the left, and what could be r_4 on the right. The two residuals are approximately equally large. The plot of r_0, however, has a lot more detail than the one of r_4. Therefore, to represent the residual r_4 well, one only needs a very coarse grid. It is this notion of smoothing that leads to the multi grid

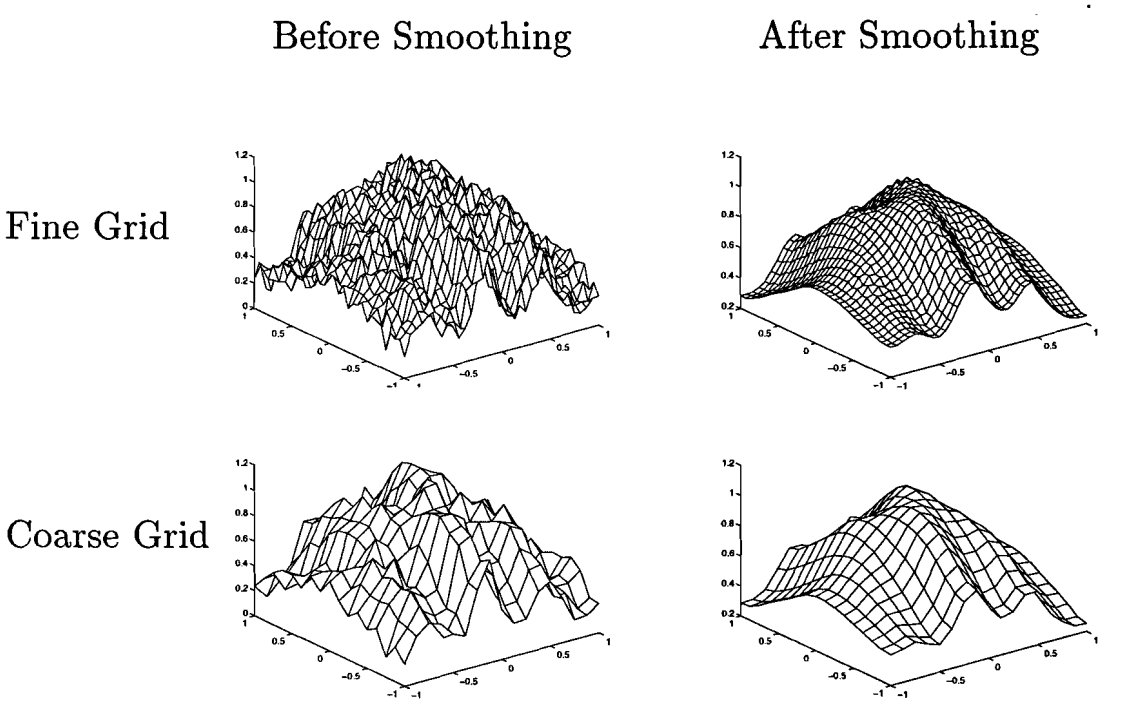

Figure 8: Plots of residual, before and after smoothing, on coarse and fine grids.

approach. If we write $\delta\phi := \phi - \phi_k$, then we can combine (4.1) and (4.2) to obtain

$$\frac{\delta\phi}{\Delta t} + \boldsymbol{F}_h(\phi_k + \delta\phi) = \boldsymbol{F}_h(\phi_k) - r_k. \tag{4.3}$$

If we can find $\delta\phi$, we can set $\phi = \phi_k + \delta\phi$ and (4.1) will be solved. To find an approximation to $\delta\phi$, we proceed as follows. First, represent ϕ_k and r_k on a coarse grid. The resulting coarse grid functions are called ϕ^H and r^H. The operator $\boldsymbol{F}_h$ is the discretization of a differential operator $\mathcal{F}$ on the fine grid. The discretization of the same operator on the coarse grid will be $\boldsymbol{F}_H$. The second step is to solve

$$\frac{\delta\phi^H}{\Delta t} + \boldsymbol{F}_H(\phi^H + \delta\phi^H) = \boldsymbol{F}_H(\phi^H) - r^H. \tag{4.4}$$

This produces $\delta\phi^H$, a good approximation of $\delta\phi$, but it is defined on the coarse grid. It is therefore necessary to represent $\delta\phi^H$ on the original fine grid. The resulting fine grid function is called $\delta\phi^h$. This is a good approximation of $\delta\phi$, so $\left(\phi_k + \delta\phi^h\right)$ will be very close to ϕ. If this new approximation of ϕ is not good enough, we redo the whole routine: first we do a small number of SCGS sweeps, then represent solution and residual on the coarse grid, solve the coarse grid equations, and represent the correction on the fine grid.

In this description of the solution procedure, mention was made of 'representing ϕ_k and r_k on the coarse grid', of 'representing $\delta\phi^H$ on the fine grid', and of 'solving the coarse grid equation'. The first two of these actions are called 'restriction' and 'prolongation'. In order to restrict, prolong

and solve, one must of course first *have* a coarse grid. In Figure 9 it is shown how the coarse grid is obtained from the fine grid. The coarse grid equation is solved using multi grid recursively, so we obtain what is called a 'W'-cycle [9]. We used 4 grids. The equation on the very coarsest grid available is solved by many (40) SCGS iterations. In Sections 4.3 and 4.4, the restriction and prolongation operators are discussed.

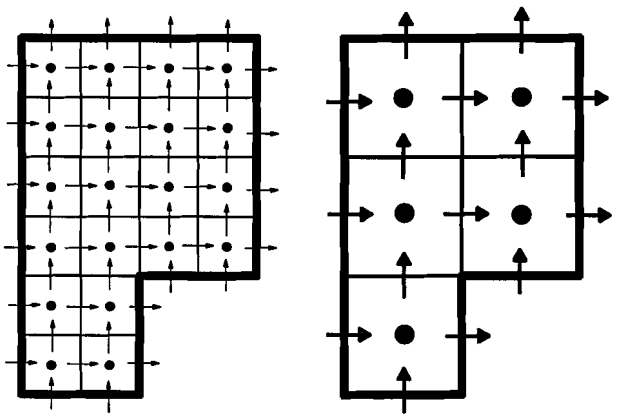

Figure 9: Fine and coarse grids.

4.4.1 The Restriction Operator

The restriction operator must make it possible to place fine grid functions on the coarse grid. As was seen in Section 3, some grid functions live on the cell-centers, while others live on the cell-faces. These will have to be restricted in different ways.

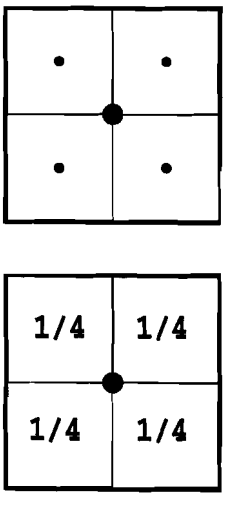

Figure 10: Cell-Centers.

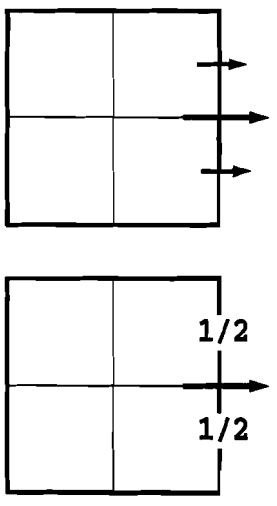

Figure 11: velocities.

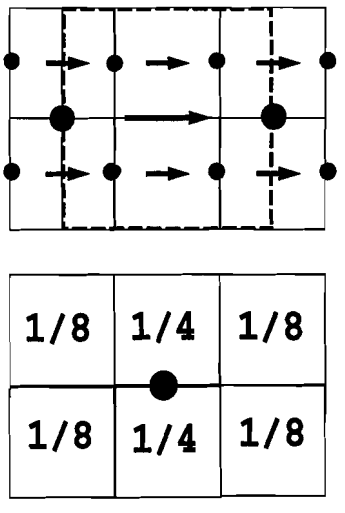

Figure 12: motion residuals.

As is seen in Figure 10, coarse grid cell-centers lie right between four fine grid cell-centers. The restricted value of a function in the coarse grid cell-center is found by taking the mean of these four fine grid cell-centers, as is indicated in Figure 10.

For the velocity grid functions, we see that a coarse grid cell face center lies right between two fine grid cell face centers. The restricted value of a velocity component on the coarse grid cell-face

is found by taking the mean of these two fine grid cell-face velocities, as is indicated in Figure 11.

Last, we must restrict the residuals of the equations of motion (2.3). This is done by area-weighting: a control-volume for the equation of motion on the coarse grid partly covers six control-volumes for the equation of motion on the fine grid. The restricted residual is then a weighted average of these six residuals, where the weighting is done according to the area of the fine grid control volumes that is inside the coarse grid control volume, as is indicated in Figure 12.

This concludes the discussion of the restriction operator we used.

4.4.2 The Prolongation Operator

Inversely to the restriction operator, the prolongation operator must make it possible to place coarse grid functions on the fine grid. However, no residuals have to be prolongated, just *corrections* to concentrations, temperatures, pressures and velocities. A fine-grid cell center is always between four coarse grid cell-centers, though some are farther away than others. Bilinear interpolation is used to find fine grid values, as Figure 13 indicates.

Some fine grid cell face centers are located between two, and others between four coarse grid cell face centers. Again, (bi)linear interpolation is used to obtain weights for constructing a weighted average, as Figure 14 indicates.

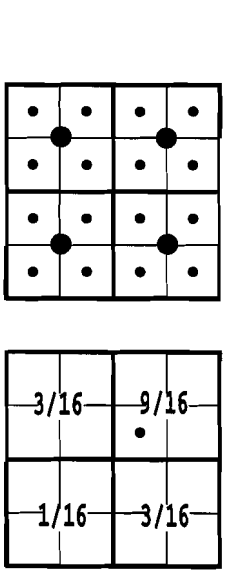

Figure 13: Cell-Centers.

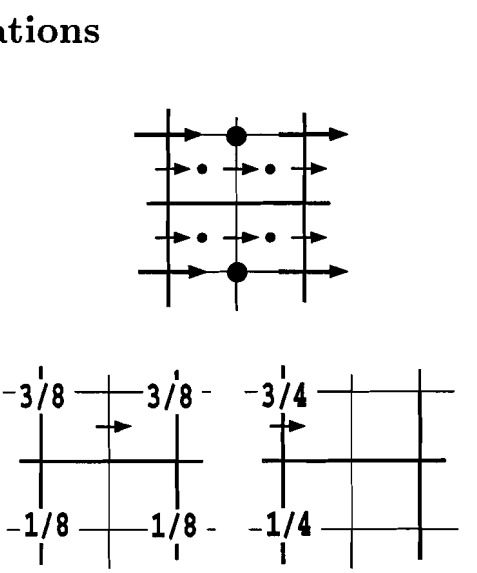

Figure 14: velocities.

4.5 Linear Subspace Minimization

Let us assume that we have a certain iterative method, $\phi_{n+1} = \boldsymbol{U}(\phi_n)$. The iteration function $\boldsymbol{U}$ can be, for instance, a W-cycle from the multi grid algorithm. Furthermore, we have a residual function $\boldsymbol{R}(\phi)$, which is properly scaled in the sense that it is a good measure of the accuracy of ϕ as a solution of the equation $\boldsymbol{R}(\phi) = \mathbf{o}$.

If the computational work involved with one iteration is large, then this work can be used more fully. This is done by storing the update v_n:

$$v_n := \phi_{n+1} - \phi_n,$$

and the effect that this update had on the residual:

$$w_n := \boldsymbol{R}(\phi_{n+1}) - \boldsymbol{R}(\phi_n).$$

Now, in first order approximation we have:

$$\boldsymbol{R}\left(\boldsymbol{\phi}_{n+1} + (\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n)\boldsymbol{\alpha}\right) \doteq \boldsymbol{R}(\boldsymbol{\phi}_{n+1}) + (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_n)\boldsymbol{\alpha},$$

for any n-vector $\boldsymbol{\alpha}$. The vector $\boldsymbol{\alpha}$ which minimizes $\|\boldsymbol{R}\|_2$ is the the solution of the (small) least squares system

$$\boldsymbol{W}\boldsymbol{\alpha} = -\boldsymbol{R}(\boldsymbol{\phi}_{n+1}).$$

This linear subspace minimization is very similar to Krylov subspace methods like GMRES or BiCGStab [14]. It takes more work and storage per iteration and is therefore less efficient than those in certain cases. In our case it has the definite advantage that it has a clear meaning for non-linear problems and that it is more flexible in (not) using certain search vectors available, for instance because the minimization procedure fails and only increases the residual. We found that this subspace minimization increased robustness of the method.

Figure 15 gives a schematical overview of the method discussed in this section.

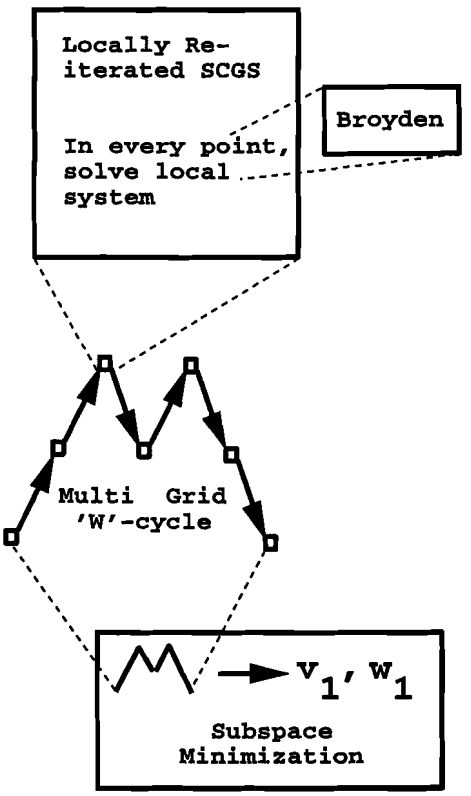

Figure 15: Schematical overview of the method.

5 Results

To test our code, we made a simulation in which rather abrupt changes occur. This would give us insight in the robustness and correctness of the method.

We did this in the following manner. On a grid of approximately 2100 points, we started from a steady solution, with a cooling (Dirichlet) condition at the lower wall and an insulation (Neumann) condition for the temperature at the right hand side of the domain. At $t = 0$ these boundary conditions were changed: cooling at the right hand side and insulation at the bottom.

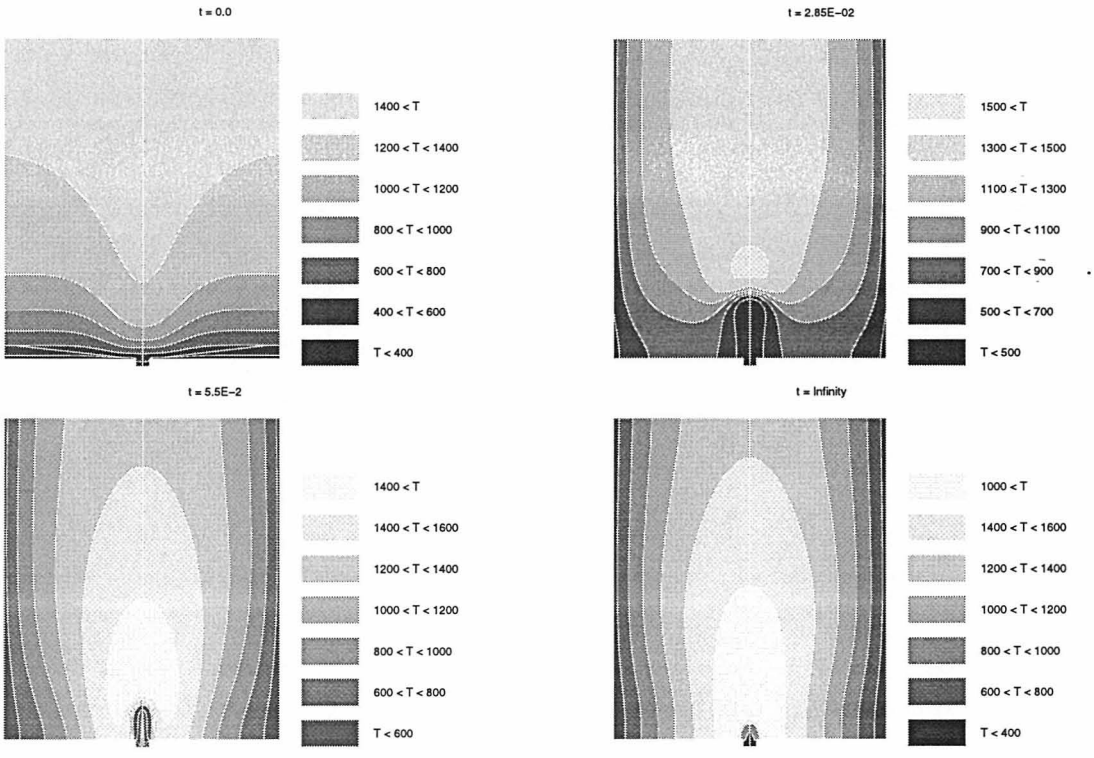

Figure 16: Temperatures.

The resulting behavior of the flame in time is shown in Figure 16. First, the temperature drops at the right. At the same time, the flame is lifted because not enough fuel is burnt. After about 0.03 seconds, burning intensity has increased so much that a hot area appears in the middle of the flame. This hot area expands and curls around the burner mouth, resulting in a new steady solution which is completely different from the initial conditions.

The behavior of the gas flow is illustrated in Figure 17. At first glance, it looks as if this solution can by no means be correct. Although the initial flow looks plausible, we see that over a very short time span, a flow develops which seems almost impossible: streak lines running into fixed walls.

At second sight, however, there appears to be an explanation for this outrageous behavior.

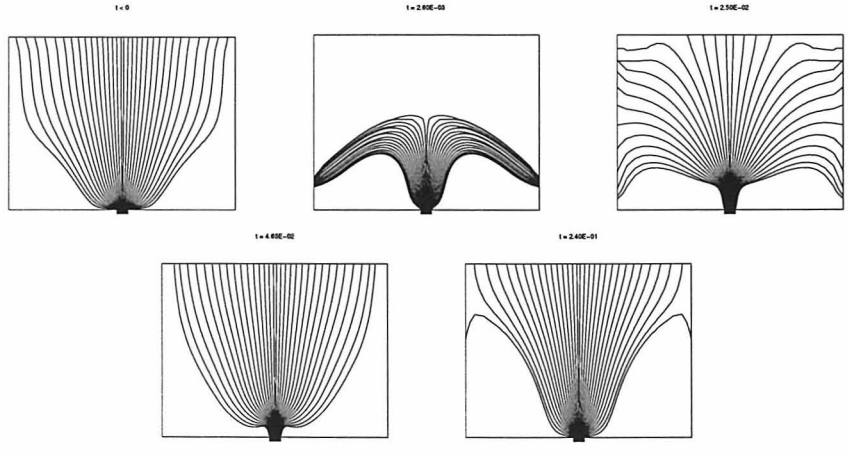

Figure 17: Streak Lines.

The side walls are cooled down extremely quickly: in no time at all, the temperature drops from 1600K to 300K. This results in an increase of the density, which in turn attracts the streak lines. After a while the flow fans out and assumes a more natural pattern, although long-time oscillating phenomena have not died out after the first 0.2 seconds.

This example shows the robustness of the code: all the abrupt changes which occur have been simulated without the method's breaking down.

References

[1] H.C. De Lange, *Modelling of Premixed Laminar Flames*, Ph.D. Thesis, Eindhoven University of Technology, 1992.

[2] C.W. Oosterlee, *Robust Multi grid Methods for the Steady and Unsteady Incompressible Navier-Stokes Equations in General Coordinates*, Ph.D. Thesis, Delft University of Technology, 1993.

[3] L.M.T. Somers, *The Simulation of Flat Flames with Detailed and Reduced Chemical Methods*, Ph.D. Thesis, Eindhoven University of Technology, 1994.

[4] R.L.G.M. Eggels, *Modelling of Combustion Processes and NO formation with Reduced Reaction Mechanisms*, Ph.D. Thesis, Eindhoven University of Technology, 1995.

[5] R.J. Kee, F.M. Rupley and J.A. Miller, *The Chemkin Thermodynamic Data Base* SANDIA REPORT SAND87-8215, Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94551.

[6] M.D. Smooke ed., *Reduced Kinetic Mechanisms and Asymptotic Approximations for Methane-Air Flames*, Lecture Notes in Physics, Springer-Verlag, Berlin Heidelberg , 1991.

[7] S.V. Patankar, *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, New York 1980.

[8] G.D. Thiart, *Improved Finite-Difference Scheme for the Solution of Convection-Diffusion Problems with the SIMPLEN algorithm*, Numer. Heat Transfer, Part B 18 (1990), pp 81-95.

[9] P. Wesseling, *An Introduction to Multi grid Algorithms*, John Wiley & Sons Ltd., Chichester, England, 1992.

[10] S.B. Margolis, *Bifurcation Phenomena in Burner Stabilized Premixed Flames*, Combustion Science & Technology, Vol 22 (1980), pp 143-169.

[11] F.A. Williams, *Combustion Theory, The Fundamental Theory of Chemically Reacting Flow Systems*, 2nd Edition, Addison-Wesley, Redwood City, 1985.

[12] U. Maas and S.B. Pope, *Simplifying Chemical Kinetics: Intrinsic Low-Dimensional Manifolds in Composition Space*, Combustion and Flame 88: 239-264 (1992). Redwood City, 1985.

[13] J.L. Mohamed and J.E. Walsh *Numerical algorithms*, Oxford, Clarendon, 1986.

[14] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Company, 1995.

PREVIOUS PUBLICATIONS IN THIS SERIES:

Number	Author(s)	Title	Month
96-25	J. de Graaf	Evolution equations	December '96
96-26	J. de Graaf	Spaces of harmonic functions and evolution equations in them	December '96
96-27	R.M.M. Mattheij S.J. Wright	Parallel algorithms for parameter identification in odes	December '96
97-01	J.H.P.A. Martens J.C. Reijenga J.H.M. ten Thije Boonkkamp R.M.M. Mattheij F.M. Everaerts	Transient modelling of capillary electrophoresis. Isotachophoresis	January '97
97-02	A.A. Reusken	Approximate cyclic reduction preconditioning	February '97
97-03	J.J.A.M. Brands	Asymptotics of non-Laplacian integrals	February '97
97-04	M.J. Noot A.C. Telea J.K.M. Jansen R.M.M. Mattheij	Real Time Numerical Simulation and Visualization of Electrochemical Drilling	March '97
97-05	H.J.C. Huijberts	Characterization of static feedback realizable transfer functions for non-linear control systems	May '97
97-06	B. van 't Hof	Numerical Simulation of Unsteady Premixed Laminar Flames	July '97