SANDIA REPORT
SAND2000-8211
Unlimited Release
Printed July 2000

OPUS: A Fortran Program for Unsteady Opposed-Flowed Flames

H. G. Im, L. L. Raja, R. J. Kee, A. E. Lutz, L. R. Petzold

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

SAND2000-8211
Unlimited Release
Printed July 2000

OPUS: A FORTRAN PROGRAM FOR UNSTEADY OPPOSED-FLOW FLAMES

H. G. Im[1*], L. L. Raja[2], R. J. Kee[2], A. E. Lutz[1], and L. R. Petzold[4]

[1]Combustion Research Facility
Sandia National Laboratories
Livermore, CA 94551-0969

[2]Engineering Division
Colorado School of Mines
Golden, CO 80401

[3]Department of Mechanical and Environmental Engineering
University of California
Santa Barbara, CA 93106

ABSTRACT

OPUS is a Fortran program for computing unsteady combustion problems in an opposed-flow configuration using one-dimensional similarity coordinate. The code is an extension of the steady counterpart, OPPDIF, to transient problems by modifying the formulation to accommodate gasdynamic compressibility effects, allowing high-accuracy time integration with adaptive time stepping. Time integration of the differential-algebraic system of equations is performed by the DASPK software package, while the Chemkin packages are used to compute chemical reaction rates and thermodynamic/transport properties. This document describes the details of the mathematical formulation and instruction for using the code.

[*]Present address: Department of Mechanical Engineering, University of Michigan, 2250 G.G. Brown Bldg., 2350 Hayward St., Ann Arbor, MI 48109-2125

This page is intentionally left blank.

INTRODUCTION

This report documents OPUS (OPposed-flow Unsteady Strained flames), a Fortran program to compute unsteady ignition or flames established between two opposing nozzles in one-dimensional similarity coordinate.

The opposed-flow geometry serves as a convenient configuration to study the effects of flow straining on the behavior of laminar flames, thereby providing useful fundamental characteristics that represent turbulent combustion in the laminar flamelet regime. By adopting a similarity coordinate, the mathematical system can be reduced into a set of one-dimensional equations, hence facilitating computational analysis. Furthermore, experimental apparatus can be easily set up, allowing for direct quantitative comparison between model prediction and measured data. For these reasons, the opposed-flow flames have been extensively studied as a canonical system for validating various chemical kinetics and transport models.

As understanding of steady laminar flames improved for the last decades, the effects of flow unsteadiness has recently attracted more attention for its relevance in turbulent combustion, which exhibit a wide spectrum of length and time scales. However, robust numerical integration of fast transient combustion processes with spatial/temporal stiffness is difficult due to the associated compressible gas dynamic behavior. From a mathematical standpoint, the standard opposed-flow formulation with a constant pressure approximation results in a system of high-index differential-algebraic equations (DAE's), such that time integration with rigorous error and time-step control can be numerically unstable during periods of rapid change in the solution [1, 2]. Moreover, for problems with rapid transients such as ignition, adaptive time-step control is needed for efficient time-integration without sacrificing the overall accuracy of the solution.

OPUS is essentially an unsteady modification of the steady counterpart, OPPDIF [3], replacing the steady Newton solver, Twopnt [4], with solution package for a DAE system, DASPK [5], an improved version of DASSL [6]. DASPK incorporates a variable-order, variable-step backward-differentiation formula (BDF) to solve general index-1 DAE's, thereby allowing for a robust time integration of stiff transient problems. In addition to this, the mathematical formulation is also modified to alleviate numerical difficulty associated with the high-index nature of the DAE system, as will be discussed later.

In the next section, the incompressible flow formulation is first described following the formulations used in the steady opposed-flow problems, and the numerical difficulty of the high-index system is discussed. The modified, compressible formulation is then derived by partially relaxing the boundary layer approximation. Methods for index reduction are also discussed following Ref. [7]. Subsequently, the instructions for running the code are then described in detail, along with the list of keywords and execution examples.

INCOMPRESSIBLE-FLOW FORMULATION: HIGH-INDEX DAE'S

Figure 1 shows a schematic of the system configuration. Two opposing axisymmetric nozzles are separated by a distance of L. Fuel and oxidizer are supplied at $x = 0$ and $x = L$, respectively, thereby forming a diffusion flame in the vicinity of the stagnation plane. The code is also compatible with a premixed flame configuration, provided the product side boundary conditions are also given. A careful choice of grid system is required to capture the unsteady flame movement. The current version of the steady code provides several grid refinement options as described later.

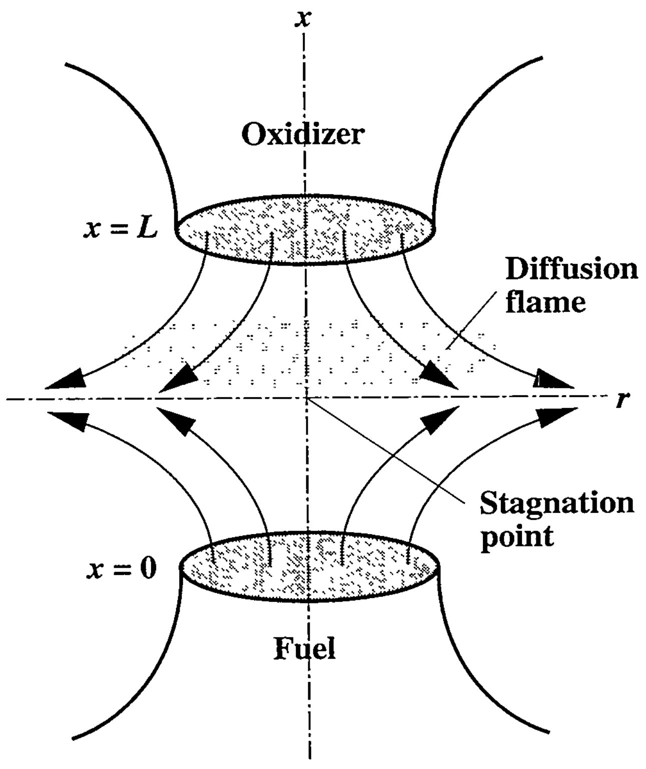

Figure 1: Schematic of the system configuration

The governing equations for the unsteady opposed-flow geometry follows following the formulation by Kee et al [8] derived for the finite-distance opposing nozzles. In contrast to the earlier potential-flow formulation [9] which has only one parameter, namely the strain rate, and hence conveniently used in many theoretical studies, the present formulation introduces an extra degree of freedom such that the velocities at both nozzle exit are given and the strain rate is computed as a solution.

Assuming similarity near the centerline, the axial velocity, radial velocity, temperature and species mass fractions are given as functions of time and the axial coordinate only:

$$u = u(t,x), \quad v/r = V(t,x), \quad T = T(t,x), \quad Y_k = Y_k(t,x). \tag{1}$$

For a system whose characteristic velocity scale is much smaller than the speed of sound, *i.e.* $Ma = u_c/a \ll 1$, the pressure, P, can be decomposed into the thermodynamic (p_0) and hydrodynamic (p) components by an asymptotic expansion in Ma. Furthermore, the boundary layer approximation leads to $\partial P/\partial x = 0$, such that:

$$P = p_0(t) + p(t, r) + o(Ma^2). \tag{2}$$

where $p/p_0 = O(Ma^2)$. Using the boundary layer approximation in the axial direction, the conservation equations become:

• Mass continuity:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) + 2\rho V = 0, \tag{3}$$

• Radial momentum:

$$\rho\frac{\partial V}{\partial t} + \rho u\frac{\partial V}{\partial x} + \rho V^2 - \frac{\partial}{\partial x}\left(\mu\frac{\partial V}{\partial x}\right) + \Lambda = 0, \tag{4}$$

• Energy conservation:

$$\rho c_p\frac{\partial T}{\partial t} + \rho c_p u\frac{\partial T}{\partial x} - \frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right) - \frac{\partial p_0}{\partial t} + \rho(\sum_k c_p Y_k V_k)\frac{\partial T}{\partial x} + \sum_k h_k W_k \omega_k = 0, \tag{5}$$

• Species conservation:

$$\rho\frac{\partial Y_k}{\partial t} + \rho u\frac{\partial Y_k}{\partial x} + \frac{\partial}{\partial x}(\rho Y_k V_k) - W_k\omega_k = 0, \quad k = 1, \cdots, K, \tag{6}$$

where c_p is the mixture specific heat, λ is the thermal conductivity of the mixture, h_k is the enthalpy of formation, W_k is the molecular weight of species k, $\bar{W}$ the mixture-averaged molecular weight, and ω_k is the molar reaction rate as defined in Chemkin [10]. Also,

$$\Lambda(t) = \frac{1}{r}\frac{\partial p}{\partial r} \tag{7}$$

is the eigenvalue of the system to be solved with other dependent variables. To maintain the banded structure of the iteration matrix, a trivial equation

$$\frac{\partial \Lambda}{\partial x} = 0 \tag{8}$$

is added.

Other constitutive relations include the equation of state

$$\rho = P\bar{W}/RT \tag{9}$$

and the diffusion velocity, V_k, given by either the multicomponent formulation

$$V_k = \frac{1}{X_k\bar{W}}\sum_{j=1}^{K} W_j D_{k,j}\frac{dX_j}{dx} - \frac{D_k^T}{\rho Y_k}\frac{1}{T}\frac{dT}{dx}, \tag{10}$$

or the mixture-averaged formulation

$$V_k = -\frac{1}{X_k} D_{km} \frac{dX_k}{dx} - \frac{D_k^T}{\rho Y_k} \frac{1}{T} \frac{dT}{dx} \quad \text{where} \quad D_{km} = \frac{1 - Y_k}{\sum_{j \neq k}^{K} X_j / \mathcal{D}_{jk}}. \tag{11}$$

where D_{kj}, D_{km}, $\mathcal{D}_{jk}$ and D_k^T are the multicomponent, binary and mixture-averaged and thermal-diffusion coefficients, respectively, and X_k is the mole fraction of species k.

The above system of equations are subject to boundary conditions as

$$\begin{aligned} x = 0: &\quad u = u_F(t), \quad V = V_F(t), \quad T = T_F(t), \quad Y_k = (Y_k)_F(t), \\ x = L: &\quad u = u_O(t), \quad V = V_O(t), \quad T = T_O(t), \quad Y_k = (Y_k)_O(t), \end{aligned} \tag{12}$$

where subscripts F and O denote the fuel and oxidizer streams, respectively. Note that, in the present formulation of finite nozzle spacing, both axial and radial velocity can be independently prescribed at the boundary, although $V = 0$ is the most common choice for practical cases. Obviously, when the boundary temperatures are varied, the density and pressure must be adjusted to satisfy the equation of state. Similarly, when species mass fractions are varied, it must be done so while satisfying the equation of state and mass conservation, $\sum Y_k = 1$. Therefore, when the mass fraction of a species is varied, another species (usually an inert species) in the same side of the boundary is also chosen to compensate for the variation of the species. To avoid excessive complexity in prescribing boundary conditions, the present code is set up such that only one variable, *i.e.* temperature or one specific species, can be varied in time at each boundary.

From a mathematical standpoint, Eqs. (3) - (6) and (8) after spatial discretization are viewed as a system of differential-algebraic equations with a solution vector $[u_j, V_j, T_j, (Y_k)_j, \Lambda_j]^T$, where the subscript j denotes grid points that range from 1 to J. In this case, u_j and Λ_j (and the boundary nodes for V, T and Y_k) constitute the algebraic variables due to the absence of $\partial u/\partial t$ and $\partial \Lambda/\partial t$ terms in the corresponding equations. These dependent variables can be grouped into three different kinds of solution vectors as follows:

$$\begin{aligned} \mathbf{x} &= [V_2, T_2, (Y_k)_2, \cdots, V_{J-1}, T_{J-1}, (Y_k)_{J-1}]^T \\ \mathbf{y} &= [u_1, \cdots, u_J, \Lambda_1, \cdots, \Lambda_{J-1}, V_1, V_J, T_1, T_J, (Y_k)_1, (Y_k)_J]^T \\ \mathbf{z} &= [\Lambda_J] \end{aligned} \tag{13}$$

then the DAE system of the present problem can be written as:

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{z}) \\ 0 &= \mathbf{g}(\mathbf{x}, \mathbf{y}) \\ 0 &= \mathbf{h}(\mathbf{y}). \end{aligned} \tag{14}$$

Equation (14) reveals the distinction between the two algebraic variables, $\mathbf{y}$ and $\mathbf{z}$. In this problem, the Jacobian $\mathbf{g_y}$ is nonsingular and all the entries in $\mathbf{y}$ are index-1 variables [1, 2]. On the other hand, the variable $\mathbf{z}$, which represents Λ_J, has an index higher than two because the Jacobian $\mathbf{h_x f_z}$ becomes singular. This is due to the fact that no explicit equation can be found to solve for the eigenvalue, $\Lambda(t)$; hence a boundary condition such as $u = u_O(t)$ is used for the algebraic equation $0 = \mathbf{h}(\mathbf{y})$. Consequently, numerical integration of the high-index system, Eqs. (3) - (6) and (8), encounters difficulties for stiff problems since DASPK is an index-1 solver. In OPUS, these difficulties are alleviated by the modification of the formulation as described in the next section.

COMPRESSIBLE-FLOW FORMULATION WITH INDEX REDUCTION

Mathematically, the system has higher index because Λ and u are not closely coupled as a result of the boundary layer approximation. Physically, the stiffness problem can be interpreted as fast transients associated with rapid gas-dynamic response, which cannot be properly captured with complete elimination of the hydrodynamic pressure from the system. That is, some spatial pressure distribution must be considered in addition to the indirect pressure effect represented by the eigenvalue, Λ. These observations suggest that more direct coupling between Λ and u can be achieved by relaxing the boundary layer approximation and re-introducing the pressure component. Similarity assumption is still needed, however, to retain one-dimensional formulation. Therefore, the pressure expression is modified as:

$$P = p_0(t) + p(t,x) + \frac{1}{2}\Lambda(t)r^2 + o(Ma^2) \tag{15}$$

such that p is introduced as an additional dependent variable, where $p/p_0 = O(Ma^2)$. The axial momentum equation is also retrieved to solve for the additional variable. By differentiating the equation of state,

$$\frac{\partial\rho}{\partial t} = \frac{\rho}{P}\frac{\partial p}{\partial t} - \frac{\rho}{T}\frac{\partial T}{\partial t} - \rho\bar{W}\sum_k \frac{1}{W_k}\frac{\partial Y_k}{\partial t}. \tag{16}$$

and substituting the $\partial\rho/\partial t$ term in the continuity equation (3) of the incompressible-flow formulation, the conservation equations can be rewritten as [7]:

- Mass continuity:

$$\frac{\rho}{p_{tot}}\frac{\partial p}{\partial t} - \frac{\rho}{T}\frac{\partial T}{\partial t} - \rho\bar{W}\sum_k \frac{1}{W_k}\frac{\partial Y_k}{\partial t} + \frac{\partial}{\partial x}(\rho u) + 2\rho V = 0. \tag{17}$$

- Axial momentum:

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} - 2\mu\frac{\partial V}{\partial x} - \frac{4}{3}\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right) + \frac{4}{3}\frac{\partial}{\partial x}(\mu V) = 0. \tag{18}$$

- Radial momentum:

$$\rho\frac{\partial V}{\partial t} + \rho u\frac{\partial V}{\partial x} + \rho V^2 - \frac{\partial}{\partial x}\left(\mu\frac{\partial V}{\partial x}\right) + \Lambda = 0. \tag{19}$$

- Energy conservation:

$$\rho c_p\frac{\partial T}{\partial t} + \rho c_p u\frac{\partial T}{\partial x} - \frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right) - \frac{\partial p_0}{\partial t} - u\frac{\partial p}{\partial x} + \rho(\sum_k c_p Y_k V_k)\frac{\partial T}{\partial x} + \sum_k h_k W_k\omega_k = 0. \tag{20}$$

• Species conservation:

$$\rho\frac{\partial Y_k}{\partial t} + \rho u\frac{\partial Y_k}{\partial x} + \frac{\partial}{\partial x}(\rho Y_k V_k) - W_k\omega_k = 0, \quad k = 1, \cdots, K. \tag{21}$$

In this new formulation, u at the interior nodes ($j = 2, \cdots, J-1$) are changed to differential variables. Furthermore, the boundary condition $u = u_O$ is now used in the JJ-th node of the axial momentum equation, and the high-index algebraic equation for Λ is replaced by another condition. Since the pressure, p, is introduced as a new dependent variable, a pressure boundary condition is also required. While there is no unique way to identify a physical boundary condition, we adopt a Bernoulli's equation at the nozzle exit at $x = L$,

$$p_J + \frac{1}{2}\rho_J u_J^2 = \text{constant}. \tag{22}$$

Note that, if Eq. (22) is used as the algebraic equation for Λ, differentiating it twice yields a differential equation form for Λ, recognizing the relation between Λ and p via Eqs. (17) and (19). Therefore, the index of Λ is reduced to 2. Further index reduction can be achieved by a simple substitution

$$\frac{d\varphi}{dt} = \Lambda(t) \tag{23}$$

with any arbitrary initial condition for ϕ, since only $d\varphi/dt$ appears in the system of equations.

In summary, Eqs. (17) - (21) have become an index-1 system through the following procedure: the algebraic equation (22) can be differentiated once with respect to t to yield an equation for $\partial V/\partial t$, which is further substituted by $d\phi/dt$ via the radial momentum equation.

NUMERICAL METHODS

Numerical integration of the index-1 DAE system, Eqs. (17) - (21), is performed using DASPK. Since DASPK requires that the initial condition must satisfy all the equations in the DAE system, a fully converged steady solution field is used as the initial condition. Due to the modified grid structure as described in the following, a modified version of OPPDIF, called OPPST, is used to obtain the initial condition.

To fit the modified formulation, OPUS employs a staggered grid system as shown in Fig. 2. The grid stencil and boundary conditions for individual dependent variables are shown in separate columns. All dependent variables are represented at the control-volume center nodes, except the axial velocity which is represented at the control-volume faces. The grid indices are shown on the left and the face indices on the right. The right-facing protuberance on the stencils indicates where the time derivative is evaluated. For the pressure-eigenvalue equation there is no time derivative, hence indicated by an unfilled protuberance.

Spatial discretization uses finite differencing for non-uniform grid system. For the species, energy, and radial momentum equations, a second-order central differencing is used for diffusive terms, and either first-order upwind or second-order central differencing is used for the convective terms, by choosing the keyword `WDIF` or `CDIF`, respectively, as was done in OPPDIF [3]. The continuity equation is spatially first order, using a central difference formulation with u at the cell surfaces. The axial momentum equation is second order in velocity and first order in pressure, utilizing the staggered-grid system.

Because the central differencing on the continuity equation is only neutrally stable, an artificial damping term is introduced to maintain numerical stability. A first-order damping term of the form $\sigma(\Delta x)(\partial^2 p/\partial x^2)$ is added in the continuity equation, where a sufficiently small value for σ is used to ensure that the solution is not affected. From our experience, $\sigma \approx 10^{-3}$ appears to be acceptable without noticeably affecting the final solution.

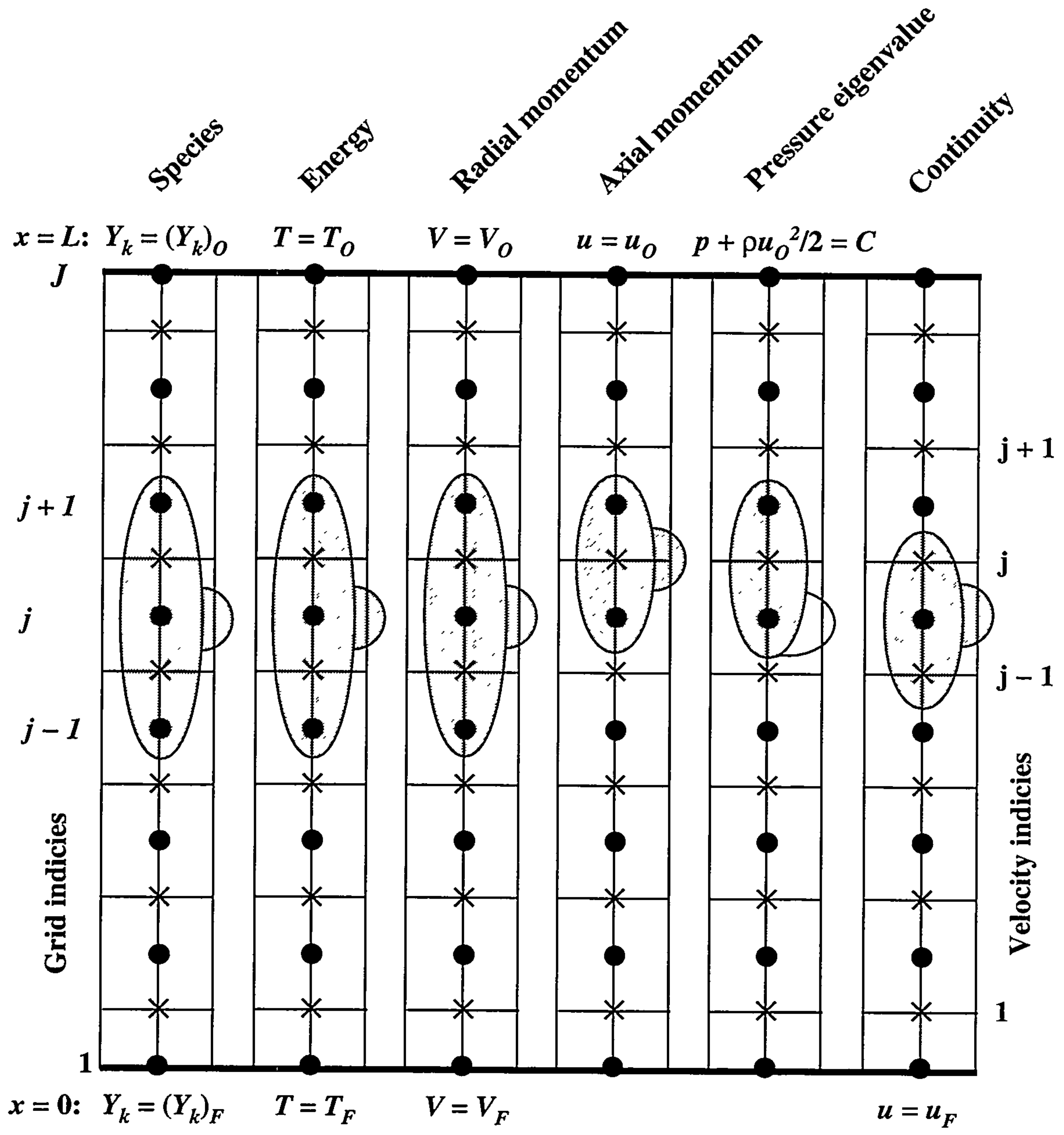
Species
Energy
Radial momentum
Axial momentum
Pressure eigenvalue
Continuity
$x = L$: $Y_k = (Y_k)_O$ $T = T_O$ $V = V_O$ $u = u_O$ $p + \rho u_O^2/2 = C$
J
$j+1$
j
$j-1$
1
Grid indicies
j + 1
j
j − 1
1
Velocity indicies
$x = 0$: $Y_k = (Y_k)_F$ $T = T_F$ $V = V_F$ $u = u_F$

Figure 2: Schematic of the grid configuration using a finite-volume, staggered-grid spatial-difference stencil.

INSTRUCTION FOR RUNNING OPUS

The current version of OPUS requires subroutines from the Chemkin-II [10]. Transport [11]. and DASPK [5] software packages. In addition. a steady solution obtained from OPPST is required as the initial field: OPUS only allows restart mode from a fully-converged solution. OPPST is a modified version of OPPDIF to be compatible with the new discretization scheme of OPUS. It also provides an option for zonal mesh refinement in order to capture the flame movement during the unsteady calculation. As with OPPDIF. OPPST also requires the Twopnt [4] package. The calculation procedure is described as follows:

1. Execute the Chemkin interpreter. which reads the reaction mechanism and the thermodynamic database. and writes the Chemkin link file.
2. Execute the transport interpreter. which reads the Chemkin link file and the transport database. and writes the transport link file.
3. Compile and link OPPST with the Chemkin. Transport and Twopnt libraries.
4. Execute OPPST. which reads the Chemkin and Transport link files and a text input file. and writes both text and binary output files.
5. Compile and link OPUS with the Chemkin. Transport and DASPK libraries.
6. Execute OPUS. which reads the Chemkin and Transport link files. a steady solution field and a text input file. and writes both text and binary output files.

Figure 3 illustrates the relationships between the routines and files. Sharp-edged rectangles represent executable programs. and round-edged rectangles denote files. The arrows indicate the input/output direction. while the lines without arrows imply links between the executable files. The following are further remarks on various elements of the package:

Chemkin and Transport Package

The first step is to execute the Chemkin interpreter (chem.exe). which reads a user-supplied reaction mechanism with element. species and reaction constant information (chem.inp). The thermodynamic properties of the species are also extracted from the database (therm.dat). All this information is stored in the Chemkin link file (chem.bin). The Transport interpreter (tran.exe) reads the Chemkin link file and the transport database (tran.dat) and creates the Transport link file (tran.bin). Both chem.bin and tran.bin must be read by OPPST or OPUS when they start execution. More details are documented in the Chemkin [10] and Transport package [11] manuals.

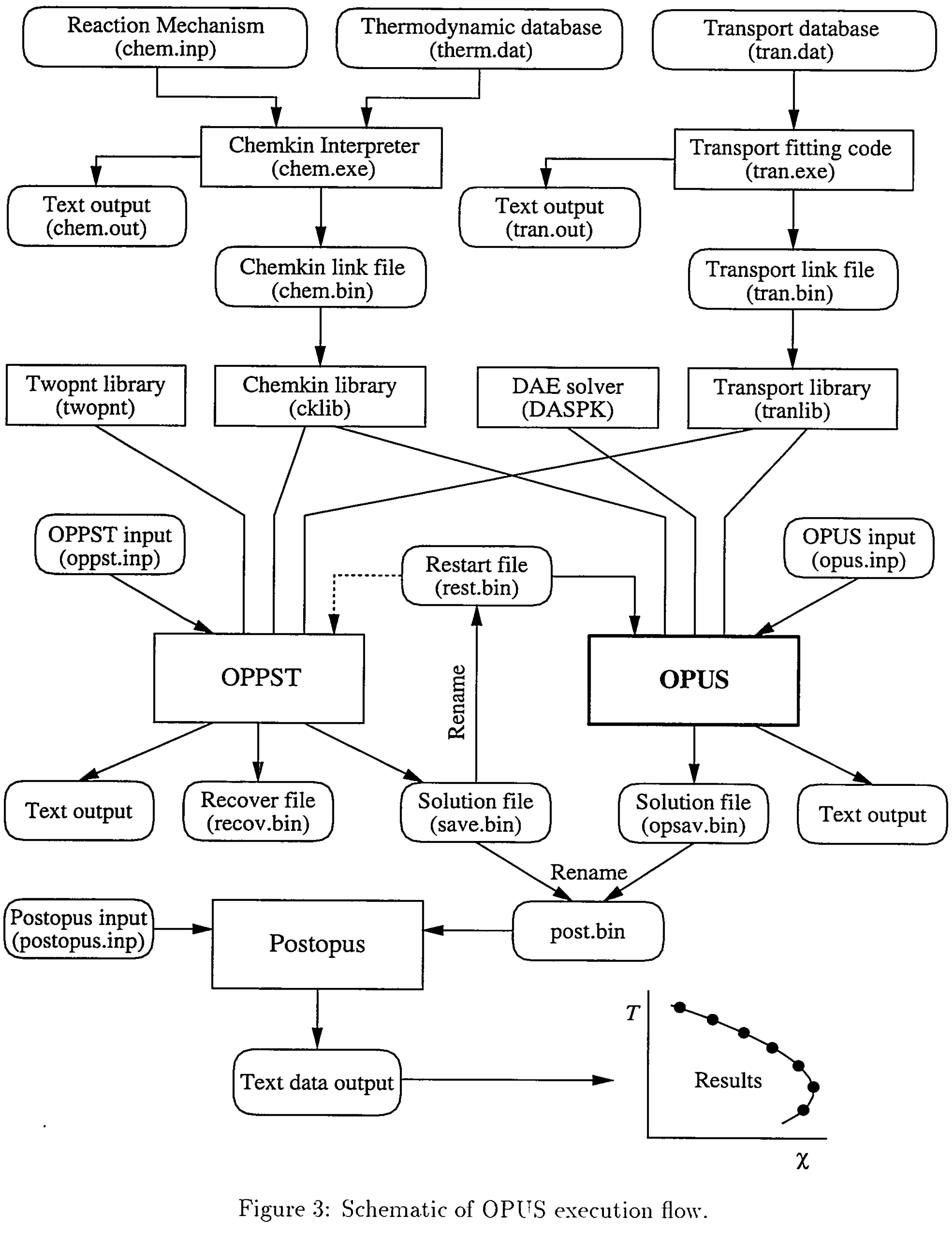

Figure 3: Schematic of OPUS execution flow.

Input and Output

In addition to the Chemkin and Transport link files, both OPPST and OPUS require a separate text input file (called oppst.inp and opus.inp, respectively) in which various options and parametric conditions are specified. These files are written using a keyword format as described in the next section.

Both OPPST and OPUS write a text output displaying a summary of execution progress. The amount of information in the text output varies depending on the setting for the keyword PRNT (see the next section).

OPPST writes two types of binary files. The Save file (save.bin) contains the last successful solution field, and the Recover file (recov.bin) contains the latest unsuccessful solution field for restart purpose. The difference between the Save and Recover files is that the Recover file is updated by OPPST during iteration, whereas the Save file is only written when a converged steady-state solution is obtained. Both Save and Recover files are written in the following Fortran format:

```
WRITE (LSAVE) 'SOLUTION'
WRITE (LSAVE) NATJ, JJ, P
WRITE (LSAVE) VFUEL, VOXID
WRITE (LSAVE) ( X(J), J = 1, JJ )
WRITE (LSAVE) ( ( S(N,J), N = 1, NATJ), J = 1, JJ )
```

OPUS writes a binary file, opsav.bin, containing the collection of solution fields in a given interval and increment. It is written in the following Fortran format:

```
WRITE (LSAVE) 'UNSTEADY'
WRITE (LSAVE) TIME
WRITE (LSAVE) NATJ, JJ, P
WRITE (LSAVE) VFUEL, VOXID
WRITE (LSAVE) ( X(J), J = 1, JJ )
WRITE (LSAVE) ( ( S(N,J), N = 1, NATJ), J = 1, JJ )
```

As time integration is not an iterative procedure, OPUS does not write a Recover file.

Grid Refinement

OPUS runs in restart mode only, and requires a fully-converged steady solution obtained from OPPST. Furthermore, OPUS does not provide any grid adaptation during time integration, and the grid structure used in the steady solution is directly used. The user should have an *a priori* knowledge of the solution response, *e.g.* the spatial range of the flame movement, and accommodate it in the grid refinement while obtaining the steady solution, as described in the following.

First, a grid redistribution by a weighting function is made using the keyword JJRG. In its default form, temperature is used as the gauge variable and the grid redistribution uses a transformation from the physical coordinate x to a new coordinate η.

$$\frac{dx}{d\eta} W(x,T) = C \tag{24}$$

with the weighting function

$$W(x,T) = 1 + b_1 \left| \frac{dT}{dx} \right| + b_2 \left| \frac{d^2T}{dx^2} \right| . \tag{25}$$

The constant C is defined by the integration over the entire domain:

$$C = \frac{1}{N-1} \int_0^L W(x,T) dx, \tag{26}$$

where N is the total number of grid points. Integrating over a portion of the domain gives an expression for the locations in the η-coordinate space:

$$\eta = 1 + \frac{1}{C} \int_0^x W(x,T) dx. \tag{27}$$

The new grid locations, x, are obtained by interpolation between the computed values of η defined using the old mesh, onto a uniform mesh in the η-space. Since $d\eta$ is constant on this uniform mesh, the solution to Eq. (24) states that $W(x,T) \cdot x$ is constant, so that the new values of x are concentrated where the weighting function is large.

As an example, the following set of keywords can be input at restart:

```
JJRG 20
PCAD 0.6
RGTC 1.0
```

This sequence creates a new solution field mapped onto 20 points, of which 60 percent is devoted to resolving gradients, with equal weighting of gradient and curvature in the temperature profile. From experience, the RGTC value of greater than 1 is recommended. Depending on the resolution of the existing solution, PCAD should be in the neighborhood of 0.5. Note that PCAD equal to zero generates a uniform mesh.

In addition to the above grid redistribution, OPPST provides the further grid refinement option. If the initial steady solution has a relatively smooth temperature profile, but a large gradient is expected later in time (such as in an ignition problem), a dummy temperature profile can be prescribed by the following keywords, *in addition to the above keywords*:

NURG a_1 a_2

which sets a temporary temperature profile of the form:

$$T(x) = \exp\left[-4\left(\frac{x-a_2}{a_1}\right)^2\right]. \tag{28}$$

which is a Gaussian shape with a width of a_2, centered at $x = a_1$. The grid refinement is then performed based on this temperature profile. Note that the artificial temperature profile created by the NURG option is used temporarily for the purpose of the grid refinement only, and does not replace the actual temperature field read from the restart file.

For the unsteady problems with reaction zones fluctuating in space, it is necessary to extend the fine grid sizes to a broader range in x. For this purpose, OPUS provides the following option. After a steady solution is converged with full grid adaptation to the degree specified by GRAD and CURV (see the next section for their detailed description), the following keyword is used on restart:

USWD a_1 a_2

The code then identifies the minimum grid interval and its location, x_m, and subsequently creates a uniform grid points with the minimum size within the band $-a_1 \cdot a_2 < x - x_m < a_1(1 - a_2)$. That is, a_1 is the band width and a_2 is the weighting of the band in each side of x_m.

User-Supplied Program

Both OPPST and OPUS are written as a subroutine for which the user must write a small "driver" program that allocates the working array, opens input/output files, and calls the subroutine. An example of the driver programs are included in the execution shell script files shown in the Examples section.

To use OPUS, the user must provide a subroutine bcfun.f in which the desired unsteady boundary conditions are specified as a function format. The following boundary conditions are available in the software distribution as shown in the Examples section:

FO_VBC, FL_VBC: prescribe $u_F(t)$ and $u_O(t)$, respectively. See keyword VBCT.
FO_TBC, FL_TBC: prescribe $T_F(t)$ and $T_O(t)$, respectively. See keyword TBCT.
FO_FBC, FL_OBC: prescribe $(Y_k)_F(t)$ and $(Y_k)_O(t)$, respectively. See keyword FBCT, FBCS, OBCT, and OBCS.

INPUT KEYWORDS

Both OPPST and OPUS read input in a keyword format. On each line of the text input file, the keyword must appear first. Some keywords require only the keyword itself, while others need additional information followed by the keyword. Some keywords have default values as indicated in the detailed description. Some keywords can be changed in continuation or restart runs; otherwise, they retain their previous values. Keywords may appear in any order. The following syntax rules apply for the keyword input:

1. The first four columns of the line are reserved for the keyword, which must begin in the first column.
2. Any further input associated with the keyword can appear anywhere in columns 5 through 80.
3. When more than one piece of information is required, the order in which the information appears is important, and the pieces are delimited by one or more blank spaces.
4. Numbers may be stated in either integer, floating point or E format. The program converts the numbers to the proper type, including conversion to double precision if the variable is declared double precision in the code.
5. Species names must appear exactly as they are specified in the Chemkin input.
6. If more information is input than required, then the last read inputs are used. For example, if contradictory keywords are encountered, the last one read is taken.
7. A comment line can be inserted by placing either a period (.), a slash (/) or an exclamation mark (!) in the first column. Such a line is ignored by the code, but it is echoed back in the printed output. In addition, on any keyword line, any input that follows the required input and is enclosed in parentheses is taken as a comment.
8. The keyword `END` must be the last line.

Detailed description of keywords and their use is given in the following. Some keywords are followed by real, integer or character input, which are denoted by a, n and c in the list.

KEYWORDS FOR OPPST

Solution Method Option

`TGIV` - Energy equation is not included. User specifies a temperature profile using the `TEMP` keyword.

`ENRG` - Energy equation is included.

`NOFT` - Skip the fixed temperature problem and include the energy equation on the first attempt.

`ATOL` a - Absolute convergence criteria for Newton iteration.
Default: 1.E-9

`RTOL` a - Relative convergence criteria for Newton iteration.
Default: 1.E-4

`ATIM` a - Absolute convergence criteria for time stepping.
Default: 1.E-9

`RTIM` a - Relative convergence criteria for time stepping.
Default: 1.E-4

`TIME` n a - Number of steps and time step for time stepping for the starting procedure.
Default: 50. 1.E-6

`TIM2` n a - Number of steps and time step for time stepping after adding the energy equation.
Default: 50. 1.E-6

`UFAC` a - Time step increase when time integration proceeds without changing the solution rapidly.
Default: 2.

`DFAC` a - Time step decrease when time integration experiences difficulties.
Default: 2.2

`DTMN` a - Minimum allowable time step.
Default: 1.E-10

`DTMX` a - Maximum allowable time step.
Default: 1.E-4

`WDIF` - Use windward differencing on convective terms in the equations.
Default: `WDIF`

`CDIF` - Use central differencing on convective terms in the equations.

`DAMP` - The coefficient, σ, used in the damping term in the continuity equation. Default: `WDIF`

`SFLR` a - Floor, or minimum value for the species mass fractions.
Default: -1.E-4

`GFAC` a - Multiply all reaction rates by a.
Default: 1.

`IRET` n - Retirement period for a given time step, or number of time steps prior to increasing the step size.
Default: 50

`ISTP` n - Directs Twopnt to take n time steps before attempting the Newton search for the steady-state solution.
Default: 0

`NJAC` n - Retirement age for the Jacobian during the Newton search.
Default: 20

`NJAC` n - Retirement age for the Jacobian during the time stepping.
Default: 20

Transport Options

`MULT` - Use the multicomponent formula for diffusion velocities, Eq. 10.
Default: `MIX`

`MIX` - Use the mixture-averaged formula for diffusion velocities, Eq. 11.
Default: `MIX`

`TDIF` - Include thermal diffusion in the diffusion velocities.
Default: inactive

Grid Parameters

`NPTS` n - Number of initial grid points. This specification is overwritten by GRID input.
Default: 6

`GRID` a - Location of an initial grid point for manual gridding.
Unit: cm

`GRAD` a - Parameter that controls grid adaptation based on the first gradient in the solution.
Default: 0.1

`CURV` a - Parameter that controls grid adaptation based on the second gradient, or curvature, in the solution.
Default: 0.5

`NADP` n - Number of grid points that Twopnt can add during each grid refinement.
Default: 3

`XEND` a - Physical length of the domain of computation, L.
Unit: cm

`JJRG` n - Perform a regrid operation on the restart solution prior to attempting to solve the new problem. Use n points for the new mesh. The new grid is adapted to the temperature profile from the restart file or prescribed by `NURG` option according to the weighting parameters `PCAD` and `RTGC`.
Default: 40

`PCAD` a - Specifies the fraction of available grid points to be used for adapting the mesh to the temperature profile in a regrid operation. Requires $0 \leq a \leq 1$. If a is zero, then uniform grids will be produced. See previous section for details.
Default: 0.6

`RGTC` a - Specifies the weighting between the first and second gradients in the temperature profile for grid adaptation. Although there is no theoretical limit, values $a \geq 1$ are recommended, because adaptation to gradient is usually more important than to curvature. See previous section for details.
Default: 1.

`NURG` a_1 a_2 - Used together with `JJRG` to prescribe a temporary Gaussian temperature profile with a width of a_1 centered at $x = a_2$ for grid adaptation. See previous section for details.
Units: cm, cm

`USWD` a_1 a_2 - On restart, the code identifies the minimum mesh size and its location, and extend it within the range $a_1 \cdot a_2 < x < a_1(1 - a_2)$. Requires $0 \leq a_2 \leq 1$. Units: cm, none

Boundary Conditions and Physical Parameters

`VFUE` a - Axial velocity at the fuel inlet, $x = 0$.
Unit: cm/s

VOXI a - Axial velocity at the oxidizer inlet, x = XEND.
Unit: cm/s

AFUE a - Radial velocity gradient, V, at the fuel inlet, $x = 0$.
Unit: cm/s
Default: 0.

AOXI a - Radial velocity gradient, V, at the oxidizer inlet, x = XEND.
Unit: cm/s
Default: 0.

TFUE a - Temperature at the fuel inlet, $x = 0$.
Unit: Kelvin
Default: 300.

TOXI a - Temperature at the oxidizer inlet, x = XEND.
Unit: Kelvin
Default: 300.

FUEL c a - Species name, c, and moles of the species at the fuel inlet, $x = 0$. The mole fractions of the species will be computed by normalizing with the sum of the input moles, so the absolute magnitude of the input quantities are unimportant.
Unit: moles or mole fraction
Default: Zero for unspecified species. At least one species input is required, however.

OXID c a - Species name, c, and moles of the species at the oxidizer inlet, x = XEND. The mole fractions of the species will be computed by normalizing with the sum of the input moles, so the absolute magnitude of the input quantities are unimportant.
Unit: moles or mole fraction
Default: Zero for unspecified species. At least one species input is required, however.

PRES a - The initial pressure of the gas mixture.
Unit: atmospheres
Default: 1.

Initialization and I/O

RSTR - Read a solution from the restart file.

SKIP n - On restart, skip n solution fields and read $n + 1$-th field from the restart file.

LINE - Linear profile used to set up initial solution.
Default: PLAT

PLAT - Plateau profile used to set up initial solution
Default: PLAT

XCEN a - Center of the mixing region used in defining the initial profile with the LINE or PLAT options.
Unit: cm

WMIX a - Width of the mixing region used in defining the initial profile with the LINE or PLAT options.
Unit: cm

TMAX a - Maximum temperature for the initial temperature profile with the LINE or PLAT options.
Unit: Kelvin
Default: 2200.

PROD c a - Species name, c, and moles of the species in the products, to be used in the initial profiles using the LINE or PLAT keywords. The mole fractions of the species will be computed by normalizing with the sum of the input moles, so the absolute magnitude of the input quantities are unimportant.
Unit: moles or mole fraction
Default: Zero for unspecified species.

TEMP a_1 a_2 - Specifies initial temperature profile in pairs of grid location and temperature, (x, T).
Unit: cm, Kelvin

USTG - On a restart, use the given temperature profile instead of the temperature solution from the restart file.

PRINT n - Integer controls the level of printing and diagnostics from Twopnt. $n = 0, 1$, and 2 provide increasing amount of output.

KOUT $c_1, c_2, \cdots$ - Species names for text output.
Default: All species printed.

DELS - Arc-length continuation parameter, ΔS. See Kee *et al.*[8] for detailed description.

CNTN - Continuation run will follow.

END - This keyword must appear at the end of the input data.

KEYWORDS FOR OPUS

*Note: "Check consistency" means that this keyword setting must be consistent with that used for the initial field obtained by OPPST.

Solution Method Option

TGIV - Energy equation is not included. User specifies a temperature profile using the TEMP keyword.

ENRG - Energy equation is included.

ATOL *a* - Absolute convergence criteria for DASPK.
Default: 1.E-9

RTOL *a* - Relative convergence criteria for DASPK.
Default: 1.E-4

WDIF - Use windward differencing on convective terms in the equations. Check consistency.
Default: WDIF

CDIF - Use central differencing on convective terms in the equations. Check consistency.

DAMP - The coefficient, σ, used in the damping term in the continuity equation. Check consistency. Default: WDIF

SFLR *a* - Floor, or minimum value for the species mass fractions.
Default: -1.E-4

GFAC *a* - Multiply all reaction rates by *a*.
Default: 1.

TEND - Final time of integration.
Unit: seconds

DT - Time step for saving solution for both text and binary outputs.
Unit: seconds

`SVNT` c a - An output option. In addition to saving solution fields at every DT, intermediate fields are saved when the spatial maximum value for the variable specified by the character string c changes by more than a compared to the last time at which the solution was saved. The input c must be a variable that has a spatial maximum, such as temperature or minor species. If c is a species, then the the unit for a is the mole fraction. If c = `T` or unassigned, then temperature is monitored and the unit for a is Kelvin.
Default: inactive

`MORD` - Maximum order of integration used in DASPK. Must be between 1 and 5. See DASPK manual [5] for details.
Default: 5

`HO` - Initial time step size for DASPK. Useful if the DAE system suffers from severe scaling difficulties on the first time step. The problem can be alleviated by specifying the step size if the user have a prior knowledge about the scaling of the problem.
Default: 0.

Transport Options

`MULT` - Use the multicomponent formula for diffusion velocities, Eq. 10. Check consistency.
Default: `MIX`

`MIX` - Use the mixture-averaged formula for diffusion velocities, Eq. 11. Check consistency.
Default: `MIX`

`TDIF` - Include thermal diffusion in the diffusion velocities. Check consistency.
Default: inactive

Grid Parameters

`XEND` a - Physical length of the domain of computation, L. Check consistency.
Unit: cm

Initial Boundary Conditions and Physical Parameters

`VFUE` a - Initial axial velocity at the fuel inlet, $x = 0$. Check consistency.
Unit: cm/s

VOXI a - Initial axial velocity at the oxidizer inlet, x = XEND. Check consistency.
Unit: cm/s

AFUE a - Radial velocity gradient, V, at the fuel inlet, $x = 0$. Check consistency.
Unit: cm/s
Default: 0.

AOXI a - Radial velocity gradient, V, at the oxidizer inlet, x = XEND. Check consistency.
Unit: cm/s
Default: 0.

TFUE a - Initial temperature at the fuel inlet, $x = 0$. Check consistency.
Unit: Kelvin
Default: 300.

TOXI a - Initial temperature at the oxidizer inlet, x = XEND. Check consistency.
Unit: Kelvin
Default: 300.

FUEL c a - Species name, c, and moles of the species at the fuel inlet, $x = 0$, as the initial condition. The mole fractions of the species will be computed by normalizing with the sum of the input moles, so the absolute magnitude of the input quantities are unimportant. Check consistency.
Unit: moles or mole fraction
Default: Zero for unspecified species. At least one species input is required, however.

OXID c a - Species name, c, and moles of the species at the oxidizer inlet, x = XEND, as the initial condition. The mole fractions of the species will be computed by normalizing with the sum of the input moles, so the absolute magnitude of the input quantities are unimportant. Check consistency.
Unit: moles or mole fraction
Default: Zero for unspecified species. At least one species input is required, however.

PRES a - The initial pressure of the gas mixture. Check consistency.
Unit: atmospheres
Default: 1.

Time-dependent Boundary Conditions

VBCT a_1 a_2 a_3 - Unsteady boundary condition for the axial velocity, $u_F(t)$ = VFUE $\times$ FO_VBC(t, a_1, a_2, a_3) and $u_O(t)$ = VOXI $\times$ FL_VBC(t, a_1, a_2, a_3). Specific functional form is customized in the subroutine `bcfun.f` using three parameters a_1, a_2 and a_3. Note that the three parameters are shared by the two boundary functions, FO_VBC and FL_VBC.

TBCT a_1 a_2 a_3 - Unsteady boundary condition for the temperature. $T_F(t) = \text{TFUE} \times \text{FO\_TBC}(t, a_1, a_2, a_3)$ and $T_O(t) = \text{TOXI} \times \text{FL\_TBC}(t, a_1, a_2, a_3)$. Specific functional form is customized in the subroutine `bcfun.f` using three parameters a_1, a_2 and a_3. Note that the three parameters are shared by the two boundary functions. FO_TBC and FL_TBC.

FBCT a_1 a_2 a_3 - Unsteady boundary condition for the fuel at $x = 0$. $(Y_k)_F(t) = \text{FUEL}(k) \times \text{FO\_FBC}(t, a_1, a_2, a_3)$ Specific functional form is customized in the subroutine `bcfun.f` using three parameters a_1, a_2 and a_3. The time-varying species Y_k is specified by the keyword FBCS.

FBCS c_1 c_2 - Species to be varied by the unsteady boundary condition FBCT. Note that c_1 is the species to be varied, and c_2 is a second inert species to compensate the variation by c_1 in order to satisfy the conservation of mass.

OBCT a_1 a_2 a_3 - Unsteady boundary condition for the oxidizer at $x = L$. $(Y_k)_O(t) = \text{OXID}(k) \times \text{FO\_OBC}(t, a_1, a_2, a_3)$ Specific functional form is customized in the subroutine `bcfun.f` using three parameters a_1, a_2 and a_3. The time-varying species Y_k is specified by the keyword OBCS.

OBCS c_1 c_2 - Species to be varied by the unsteady boundary condition OBCT. Note that c_1 is the species to be varied, and c_2 is a second inert species to compensate the variation by c_1 in order to satisfy the conservation of mass.

Initialization and I/O

RSTR - Read a solution from the restart file. Must be active for all OPUS runs.

SKIP n - On restart, skip n solution fields and read $n+1$-th field from the restart file.

PRINT n - Integer controls the level of printing and diagnostics from Twopnt. $n = 0, 1,$ and 2 provide increasing amount of output.

KOUT $c_1, c_2, \cdots$ - Species names for text output.
Default: All species printed.

DELS - Arc-length continuation parameter, ΔS. See Kee *et al.*[8] for detailed description.

CNTN - Continuation run will follow.

END - This keyword must appear at the end of the input data.

APPLICATION EXAMPLE

The application example described in this section is for a diffusion flame in a hydrogen-air system. First, a steady diffusion flame is solved for the boundary conditions of $u_F = u_O = 100$ cm/sec, $T_F = T_O = 300$K, and the species boundary conditions are $X_{H_2} = 0.5, X_{N_2} = 0.5$ at $x = 0$ and $X_{O_2} = 0.21, X_{N_2} = 0.79$ at $x = L$. Pressure p_0 is assigned a constant value of 1 atm. Once the steady condition is obtained with OPPST, the unsteady solution is computed using the time-dependent boundary condition for the axial velocity:

$$u_F(t) = u_O(t) = 100 \times [1 + A\{1 - \cos(2\pi f t)\}], \tag{29}$$

such that the velocity varies from 100 cm/sec to (100+2A) cm/sec, at a frequency of f Hz. In this test run, $A = 0.1$ and $f = 100$ Hz.

The following set of source codes and data files are needed:

- Chemkin source codes: `ckinterp.f cklib.f`
- Transport source codes: `tranfit.f tranlib.f`
- Software packages: `twopnt.f ddaspk.f`
- Application codes: `oppst.f driver.f opus.f opdriv.f`
- DASPK utility codes: `daux.f dlinpk.f mpi_dummy.f adf_dummy.f`
- Other utility codes: `eqlib.f stanlib.f math.f dmach.f cputim.f`
- Data files: `chem.inp therm.dat tran.dat oppst.inp opus.inp`

Self-contained execution shell script for steady (oppst.sh) and unsteady (opus.sh) problems are also shown in this section. The execution sequence will be as follows:

1. Execute steady code using the shell script **oppst.sh**.
2. Rename the binary file **save.bin** to **rest.bin**.
3. Execute unsteady code using the shell script **opus.sh**.

Example Input to Chemkin Interpreter: chem.inp [12]

```
ELEMENTS
H O N
END
SPECIES
H2 O2 O OH H2O H HO2 H2O2 N2
END
REACTIONS
H+O2=O+OH 1.915E+14 0.00 1.644E+04
O+H2=H+OH 5.080E+04 2.67 6.290E+03
H2+OH=H2O+H 2.160E+08 1.51 3.430E+03
OH+OH=O+H2O 1.230E+04 2.62 -1.880E+03
H2+M=H+H+M 4.577E+19 -1.40 1.044E+05
H2/2.5/ H2O/12/
O+O+M=O2+M 6.165E+15 -0.50 0.000E+00
H2/2.5/ H2O/12/
O+H+M=OH+M 4.714E+18 -1.00 0.000E+00
H2/2.5/ H2O/12/
H+OH+M=H2O+M 2.240E+22 -2.00 0.000E+00
H2/2.5/ H2O/6.3/
H+O2+M=HO2+M 6.170E+19 -1.42 0.000E+00
H2/2.5/ H2O/12/
HO2+H=H2+O2 6.630E+13 0.00 2.130E+03
HO2+H=OH+OH 1.690E+14 0.00 8.740E+02
HO2+O=O2+OH 1.810E+13 0.00 -4.000E+02
HO2+OH=H2O+O2 1.450E+16 -1.00 0.000E+00
HO2+HO2=H2O2+O2 3.020E+12 0.00 1.390E+03
H2O2+M=OH+OH+M 1.202E+17 0.00 4.550E+04
H2/2.5/ H2O/12/
H2O2+H=H2O+OH 1.000E+13 0.00 3.590E+03
H2O2+H=HO2+H2 4.820E+13 0.00 7.950E+03
H2O2+O=OH+HO2 9.550E+06 2.00 3.970E+03
H2O2+OH=HO2+H2O 7.000E+12 0.00 1.430E+03
END
```

Example OPPST Input: oppst.inp

```
MIX
ENRG
DAMP 0.005
AFUE 0
AOXI 0
VFUE 100
VOXI 100
TFUE 300
TOXI 300
TMAX 2100
NPTS 20
XEND 1.0
XCEN 0.5
WMIX 0.3
PRES 1.0
IRET 20
UFAC 2.
PRNT 1
TIME 200 1.E-6
TIM2 200 1.E-6
DTMX 1.E-2
GRAD 0.1
CURV 0.5
FUEL H2 0.5
FUEL N2 0.5
OXID O2 0.21
OXID N2 0.79
PROD H2O 0.075
PROD N2 0.925
KOUT H2 O2 H2O O H OH
RTOL 1.E-9
ATOL 1.E-11
ATIM 1.E-6
RTIM 1.E-3
END
```

Example OPUS Input: opus.inp

```
RSTR
MIX
ENRG
DAMP 0.005
AFUE 0
AOXI 0
VFUE 100
VOXI 100
TFUE 300
TOXI 300
XEND 1.0
PRES 1.0
TEND 0.05
DT 0.0005
MORD 5
VBCT 0.1 100.0 0.0
FUEL H2 0.5
FUEL N2 0.5
OXID O2 0.21
OXID N2 0.79
KOUT H2 O2 H2O O H OH
RTOL 1.E-4
ATOL 1.E-11
SVNT T 100.
END
```

Example driver program for OPPST: driver.f

```
      PROGRAM DRIVER
C*****precision > double
      IMPLICIT DOUBLE PRECISION (A-H, O-Z), INTEGER (I-N)
C*****END precision > double
C*****precision > single
C     IMPLICIT REAL (A-H, O-Z), INTEGER (I-N)
C*****END precision > single
C
      PARAMETER (LENRWK=4000000, LENIWK=20000, LENLWK=500, LENCWK=100,
     1           LIN=5, LOUT=6, LINKCK=25, LINKMC=35, LRIN=14,
     2           LROUT=15, LRCRVR=16)
      DIMENSION RWORK(LENRWK), IWORK(LENIWK), LWORK(LENLWK)
      CHARACTER*16 CWORK(LENCWK)
C
      NMAX = 400
C
      OPEN (LINKCK, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1       FILE='chem.bin')
      OPEN (LINKMC, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1       FILE='tran.bin')
      OPEN (LRCRVR, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1       FILE='recov.bin')
      OPEN (LROUT, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1       FILE='save.bin')
      OPEN (LRIN, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1       FILE='rest.bin')
      CALL OPPDIF (NMAX, LIN, LOUT, LINKCK, LINKMC, LRIN, LROUT,
     1             LRCRVR, LENLWK, LWORK, LENIWK, IWORK,
     2             LENRWK, RWORK, LENCWK, CWORK)
C
      STOP
      END
```

Example driver program for OPUS: opdriv.f

```
      PROGRAM DRIVER
C
C*****precision > double
      IMPLICIT DOUBLE PRECISION (A-H, O-Z), INTEGER (I-N)
C*****END precision > double
C*****precision > single
C     IMPLICIT REAL (A-H, O-Z), INTEGER (I-N)
C*****END precision > single
C
      PARAMETER (LENRWK=7000000, LENIWK=35000, LENLWK=200, LENCWK=200,
     1           LIN=5, LOUT=6, LINKCK=25, LINKMC=35, LREST=14,
     2           LSAVE=15, LRCRVR=16)
      DIMENSION RWORK(LENRWK), IWORK(LENIWK), LWORK(LENLWK)
      CHARACTER*16 CWORK(LENCWK)
C
      NMAX = 601
C
      OPEN (LINKCK, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1      FILE='chem.bin')
      OPEN (LINKMC, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1      FILE='tran.bin')
      OPEN (LRCRVR, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1      FILE='oprec.bin')
      OPEN (LSAVE, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1      FILE='opsav.bin')
      OPEN (LREST, FORM='UNFORMATTED', STATUS='UNKNOWN',
     1      FILE='rest.bin')
      CALL OPUS (NMAX, LIN, LOUT, LINKCK, LINKMC, LREST, LSAVE,
     1           LRCRVR, LENLWK, LWORK, LENIWK, IWORK,
     2           LENRWK, RWORK, LENCWK, CWORK)
C
      STOP
      END
```

Example execution shell script for OPPST: oppst.sh

```
cat << EOF > makefile

#*****compiler > sun, sgi
FFLAGS = -static -O2
F77 = f77 -static -O2 -c
LINK = f77 -o
MACH = dmach.o
#*****END compiler > sun, sgi

HOMDIR = .
BINDIR = $(HOMDIR)/bin/
SRCDIR = $(HOMDIR)/sources/
OBJDIR = $(HOMDIR)/objects/
OBJS = $(OBJDIR)driver.o $(OBJDIR)oppst.o $(OBJDIR)eqlib.o
        $(OBJDIR)stanlib.o $(OBJDIR)cklib.o $(OBJDIR)tranlib.o
        $(OBJDIR)twopnt.o $(OBJDIR)cputim.o $(OBJDIR)math.o $(OBJDIR)$(MACH)

chem.exe :  $(OBJDIR)ckinterp.o
        $(LINK) chem.exe $(OBJDIR)ckinterp.o

tran.exe :  $(OBJDIR)tranfit.o $(OBJDIR)cklib.o $(OBJDIR)math.o
                $(OBJDIR)$(MACH)
       $(LINK) tran.exe $(OBJDIR)tranfit.o $(OBJDIR)cklib.o
                $(OBJDIR)math.o $(OBJDIR)$(MACH)

oppst.exe :  $(OBJS)
       $(LINK) oppst.exe $(OBJS)

$(OBJDIR)ckinterp.o:  $(SRCDIR)ckinterp.f
       cd $(OBJDIR); $(F77) $(SRCDIR)ckinterp.f
$(OBJDIR)cklib.o:  $(SRCDIR)cklib.f
       cd $(OBJDIR); $(F77) $(SRCDIR)cklib.f
$(OBJDIR)tranfit.o:  $(SRCDIR)tranfit.f
       cd $(OBJDIR); $(F77) $(SRCDIR)tranfit.f
```

```
$(OBJDIR)tranlib.o:  $(SRCDIR)tranlib.f
       cd $(OBJDIR); $(F77) $(SRCDIR)tranlib.f
$(OBJDIR)eqlib.o:  $(SRCDIR)eqlib.f
       cd $(OBJDIR); $(F77) $(SRCDIR)eqlib.f
$(OBJDIR)stanlib.o:  $(SRCDIR)stanlib.f
       cd $(OBJDIR); $(F77) $(SRCDIR)stanlib.f
$(OBJDIR)math.o:  $(SRCDIR)math.f
       cd $(OBJDIR); $(F77) $(SRCDIR)math.f
$(OBJDIR)dmach.o:  $(SRCDIR)dmach.f
       cd $(OBJDIR); $(F77) $(SRCDIR)dmach.f
$(OBJDIR)driver.o:  $(SRCDIR)driver.f
       cd $(OBJDIR); $(F77) $(SRCDIR)driver.f
$(OBJDIR)oppst.o:  $(SRCDIR)oppst.f
       cd $(OBJDIR); $(F77) $(SRCDIR)oppst.f
$(OBJDIR)twopnt.o:  $(SRCDIR)twopnt.f
       cd $(OBJDIR); $(F77) $(SRCDIR)twopnt.f
$(OBJDIR)cputim.o:  $(SRCDIR)cputim.f
       cd $(OBJDIR); $(F77) $(SRCDIR)cputim.f
EOF

touch makefile; make chem.exe tran.exe oppst.exe
chem.exe; tran.exe > tran.out; oppst.exe < oppst.inp > oppst.out

ENDSH
```

Example execution shell script for OPUS: opus.sh

```
cat << EOF > makefile

#*****compiler > sun, sgi
FFLAGS = -static -O2
F77 = f77 -static -O2 -c
LINK = f77 -o
MACH = dmach.o
#*****END compiler > sun, sgi

HOMDIR = .
BINDIR = $(HOMDIR)/bin/
SRCDIR = $(HOMDIR)/sources/
OBJDIR = $(HOMDIR)/objects/
OBJS = $(OBJDIR)opdriv.o $(OBJDIR)opus.o $(OBJDIR)cklib.o
        $(OBJDIR)tranlib.o $(OBJDIR)cputim.o $(OBJDIR)ddaspk.o
        $(OBJDIR)daux.o $(OBJDIR)dlinpk.o $(OBJDIR)mpi_dummy.o
        $(OBJDIR)adf_dummy.o $(OBJDIR)math.o $(OBJDIR)bcfun.o $(OBJDIR)$(MACH)

chem.exe :  $(OBJDIR)ckinterp.o
        $(LINK) chem.exe $(OBJDIR)ckinterp.o

tran.exe :  $(OBJDIR)tranfit.o $(OBJDIR)cklib.o $(OBJDIR)math.o
                $(OBJDIR)$(MACH)
       $(LINK) tran.exe $(OBJDIR)tranfit.o $(OBJDIR)cklib.o
                $(OBJDIR)math.o $(OBJDIR)$(MACH)

opus.exe :  $(OBJS)
       $(LINK) opus.exe $(OBJS)

$(OBJDIR)ckinterp.o:  $(SRCDIR)ckinterp.f
       cd $(OBJDIR); $(F77) $(SRCDIR)ckinterp.f
$(OBJDIR)cklib.o:  $(SRCDIR)cklib.f
       cd $(OBJDIR); $(F77) $(SRCDIR)cklib.f
$(OBJDIR)tranfit.o:  $(SRCDIR)tranfit.f
       cd $(OBJDIR); $(F77) $(SRCDIR)tranfit.f
```

```
$(OBJDIR)tranlib.o:  $(SRCDIR)tranlib.f
       cd $(OBJDIR); $(F77) $(SRCDIR)tranlib.f
$(OBJDIR)math.o:  $(SRCDIR)math.f
       cd $(OBJDIR); $(F77) $(SRCDIR)math.f
$(OBJDIR)dmach.o:  $(SRCDIR)dmach.f
       cd $(OBJDIR); $(F77) $(SRCDIR)dmach.f
$(OBJDIR)opdriv.o:  $(SRCDIR)opdriv.f
       cd $(OBJDIR); $(F77) $(SRCDIR)opdriv.f
$(OBJDIR)opus.o:  $(SRCDIR)opus.f
       cd $(OBJDIR); $(F77) $(SRCDIR)opus.f
$(OBJDIR)bcfun.o:  $(SRCDIR)bcfun.f
       cd $(OBJDIR); $(F77) $(SRCDIR)bcfun.f
$(OBJDIR)ddaspk.o:  $(SRCDIR)ddaspk.f
       cd $(OBJDIR); $(F77) $(SRCDIR)ddaspk.f
$(OBJDIR)daux.o:  $(SRCDIR)daux.f
       cd $(OBJDIR); $(F77) $(SRCDIR)daux.f
$(OBJDIR)dlinpk.o:  $(SRCDIR)dlinpk.f
       cd $(OBJDIR); $(F77) $(SRCDIR)dlinpk.f
$(OBJDIR)mpi_dummy.o:  $(SRCDIR)mpi_dummy.f
       cd $(OBJDIR); $(F77) $(SRCDIR)mpi_dummy.f
$(OBJDIR)adf_dummy.o:  $(SRCDIR)adf_dummy.f
       cd $(OBJDIR); $(F77) $(SRCDIR)adf_dummy.f
$(OBJDIR)cputim.o:  $(SRCDIR)cputim.f
       cd $(OBJDIR); $(F77) $(SRCDIR)cputim.f
EOF

touch makefile; make chem.exe tran.exe opus.exe
chem.exe; tran.exe > tran.out; opus.exe < opus.inp > opus.out

ENDSH
```

POST-PROCESSING

The post-processing programs consist of two different routines: POST (using `post.f`) and PROF (using `prof.f`). Both programs must be linked to Chemkin libraries and requires the Chemkin and Transport binary files. They read the binary solution fields (either **save.bin** or **opsav.bin** renamed to **post.bin**) and write the text output to be used for graphics.

POST

POST outputs the spatial maximum values for the series of solution fields contained in **post.bin**, thereby allowing parametric presentations such as the maximum temperature vs. scalar dissipation rate. The code requires the following text input file named **post.in**:

```
MFILE     TFILE       FREQ
1         0.0         1000.0
KOUT H2 O2 H O OH HO2 H2O2
```

where `MFILE` specifies the maximum number of steady solution fields to be read from **post.bin** created by a continuation runs of OPPST, and `TFILE` specifies the latest time for the unsteady solution fields to be read from **post.bin** created by OPUS. `FREQ` is an input parameter used only for the problems with sinusoidal oscillation, where it specifies the frequency of imposed oscillation to extract phase information. It must be consistent with the value used to obtain the unsteady solution. Finally, `KOUT` specifies the list of species that are written in the output. POST writes the following text output files:

`post1.tec`: Basic quantities such as velocity, strain rate, maximum temperature, etc.
`post2.tec`: Maximum mole fraction of the species specified by `KOUT`.
`post3.tec`: Maximum reaction rates for various species specified by `KOUT`.

PROF

PROF outputs the spatial profile information for a specific solution field out of **post.bin**. The code requires the following text input file named **prof.in**:

```
MFILE     TFILE       FREQ
3         0.1         1000.0
KOUT H2 O2 H O OH HO2 H2O2
```

where `MFILE` = 3 specifies that the third solution field in the steady continuation result is read (provided there exist at least 3 continuation runs), more than specifies the maximum number of steady solution fields to be while TFILE specifies that the code reads the solution field at the closest time that exceeds the TFILE value. FREQ is an input parameter used only for the problems with sinusoidal oscillation, where it specifies the frequency of imposed oscillation to extract phase information. It must be consistent with the value used to obtain the unsteady solution. Finally, KOUT specifies the list of species that are written in the output. PROF writes the text output files **prof.tec**, in which the solution variables are written as a function of space variable.

Figures 4 - 7 show the graphic results of the example runs shown in the previous section. The spatial profiles for the steady diffusion flame used as the initial condition are shown in Figs. 4 and 5. Figures 6 and 7 show the temporal oscillations of various dependent variables, where $\chi_{st} = 2D(\partial\xi/\partial x)^2$ is the scalar dissipation rate which is represents the characteristic flow time scale in the diffusion flame.

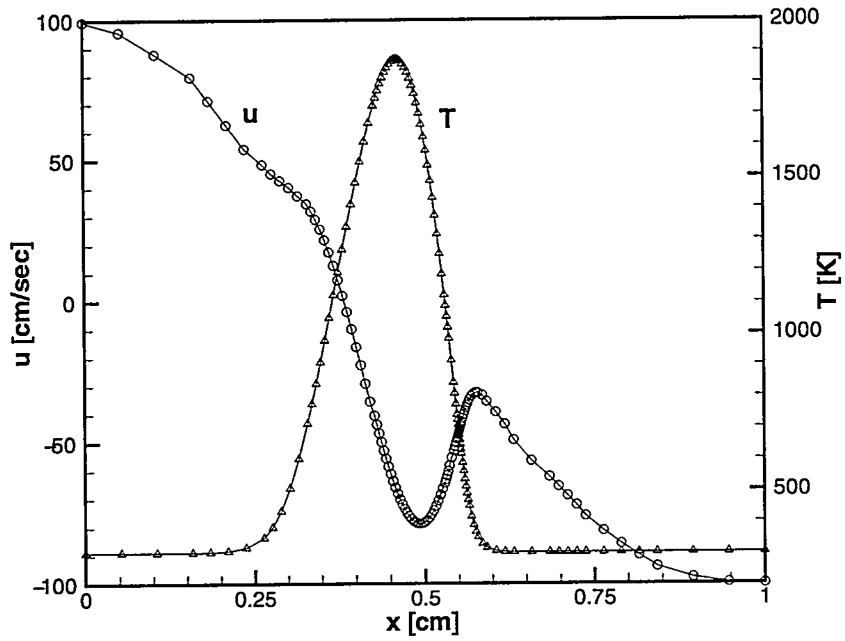

Figure 4: Axial velocity and temperature profile for the steady diffusion flame obtained from the OPPST example run described in the previous section.

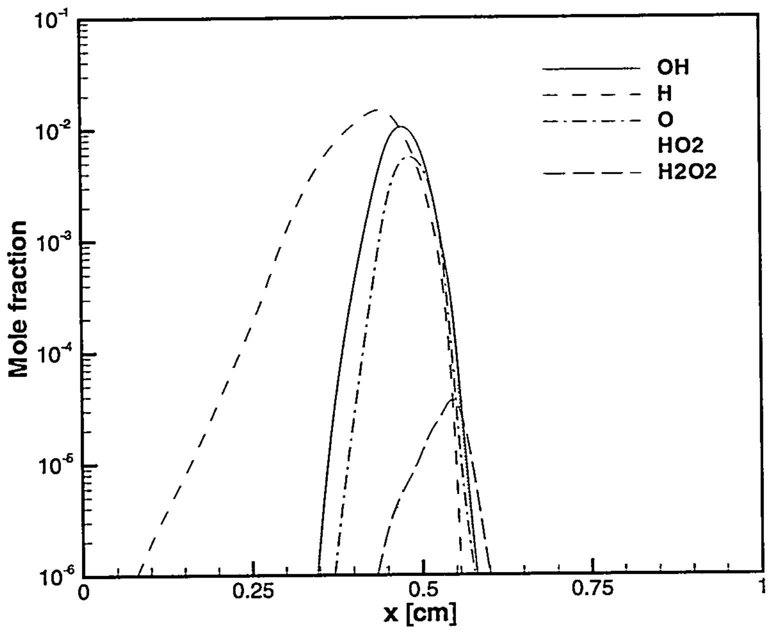

Figure 5: Profiles of mole fraction of minor species for the steady diffusion flame obtained from the OPPST example run described in the previous section.

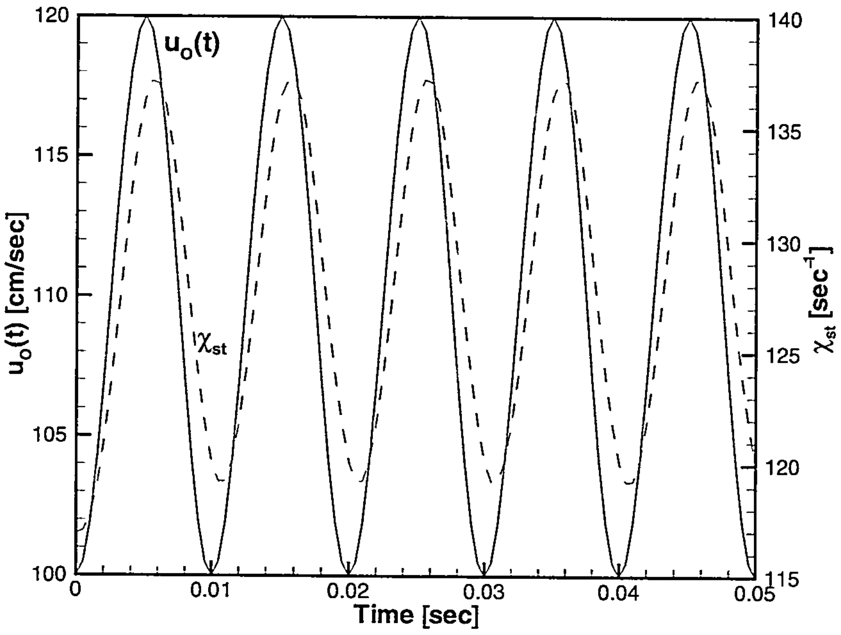

Figure 6: Imposed axial velocity oscillation and the response of scalar dissipation rate as a function of time for the unsteady diffusion flame obtained from the OPUS example run described in the previous section.

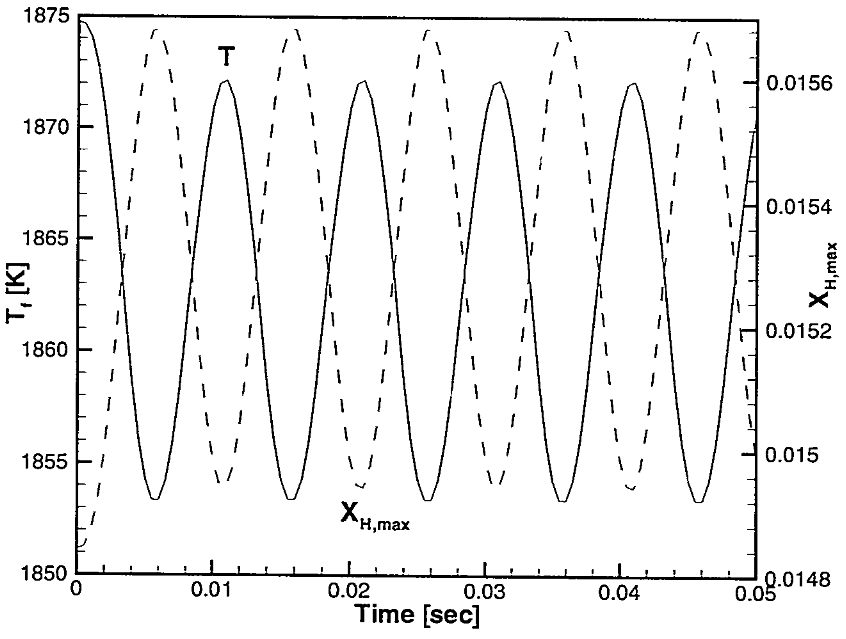

Figure 7: Unsteady response of the flame temperature and the maximum mole fraction of H atom as a function of time for the unsteady diffusion flame obtained from the OPUS example run described in the previous section.

Acknowlegdment

This work was mainly supported by the U. S. Department of Energy, Office of Basic Energy Sciences, Chemical Sciences Division, while the authors at Colorado School of Mines were supported in part by NASA through the CCACS program.

References

[1] Brenan, K. E., Campbell, S. L. and Petzold, L. R., *Numerical Solution of Initial-Value Problems in Differential Algebraic Equations*. 2nd ed., SIAM, Philadelphia, PA, 1996.

[2] Ascher, U. M. and Petzold, L. R., *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, PA, 1998.

[3] Lutz, A. E., Kee, R. J., Grcar, J. F. and Rupley, F. M. (1996). OPPDIF: A Fortran Program for Computing Opposed-flow Diffusion Flames, *Sandia Report* SAND96-8243, May 1996.

[4] Grcar, J. F. (1992). The Twopnt Program for Boundary Value Problems, *Sandia Report* SAND91-8230, April 1992.

[5] Li, S. and Petzold, L. R. (1999). Design of New DASPK for Sensitivity Analysis. *Technical Report of Computer Science Department* (TRCS99-23), University of California, Santa Barbara.

[6] Petzold, L. R. (1982). A Description of DASSL: A Differential/Algebraic System Solver. *Sandia Report* SAND82-8637, Sept. 1982.

[7] Raja, L. L., Kee, R., J. and Petzold, L. R. (1998). Simulation of the Transient, Compressible, Gas-Dynamic, Behavior of Catalytic-Combustion Ignition in Stagnation Flows, *Twenty-Seventh Symposium (International) on Combustion*, The Combustion Institute, pp. 2249-2257.

[8] Kee, R. J., Miller, J. A., Evans, G. H. and Dixon-Lewis, G. (1988). A Computational Model of the Structure and Extinction of Strained, Opposed Flow, Premixed Methane-Air Flames, *Twenty-Second Symposium (International) on Combustion*, The Combustion Institute, pp. 1479-1494.

[9] Dixon-Lewis, G., David, T., Gaskell, P. H., Fukutani, S., Jinno, H., Miller, J. A., Kee, R. J., Smooke, M. D., Peters, N., Effelsberg, E., Warnatz, J. and Behrendt, F. (1984). Calculation of the Structure and Extinction Limit of a Methane-Air Counterflow Diffusion Flame in the Forward Stagnation Region of a Porous Cylinder. *Twentieth Symposium (International) on Combustion*, The Combustion Institute, Pittsburgh, PA, pp. 1893-1904, p. 95.

[10] Kee, R. J., Rupley, F. M. and Miller, J. A. (1991). Chemkin-II: A Fortran Chemical Kinetics Package for the Analysis of Gas-Phase Chemical Kinetics. *Sandia Report* SAND89-8009B, November 1991.

[11] Kee, R. J., Dixon-Lewis, G., Warnatz, J., Coltrin, M. E. and Miller, J. A. (1986). A Fortran Computer Code Package for the Evaluation of Gas-Phase Multicomponent Transport Properties. *Sandia Report* SAND86-8246, December 1986.

[12] Yetter, R. A., Dryer, F. L. and Rabitz, H. (1991). A Comprehensive Reaction Mechanism for Carbon Monoxide/Hydrogen/Oxygen Kinetics. *Combust. Sci. Tech.*, 79:97-128.

DISTRIBUTION:

Im, Hong G. (30)
Mechanical Engineering and Applied Mechanics
Univ of Michigan CoE / 2250 GG Brown Lab
2350 Hayward St
Ann Arbor, MI 48109-2125

Raja, L. L. (10)
Engineering Division
Colorado School of Mines
Golden, CO 80401-1887

Kee, R. J. (10)
Engineering Division
Colorado School of Mines
Golden, CO 80401-1887

Petzold, L. R.
Department of Mechanical and Environmental Engineering
University of California
Santa Barbara, CA 93106

Klipstein, David H.
Reaction Design
11436 Sorrento Valley Road
San Diego, CA 92121

Meeks, Ellen
Reaction Design
6500 Dublin Blvd., Suite 214
Dublin, CA 94568

Atreya, Arvind
Mechanical Engineering and Applied Mechanics
Univ of Michigan CoE / 2250 GG Brown Lab
2350 Hayward St
Ann Arbor, MI 48109-2125

Wooldridge, Margaret S.
Mechanical Engineering and Applied Mechanics
Univ of Michigan CoE / 2250 GG Brown Lab
2350 Hayward St
Ann Arbor, MI 48109-2125

Assanis, Dennis N.
Mechanical Engineering and Applied Mechanics
Univ of Michigan CoE / 2043 WE Lay Auto Lab
1231 Beal Ave
Ann Arbor MI 48109-2121

Sick, Volker
Mechanical Engineering and Applied Mechanics
Univ of Michigan CoE / 2043 WE Lay Auto Lab
1231 Beal Ave
Ann Arbor MI 48109-2121

Dahm, Werner J. A.
Laboratory for Turbulence \& Combustion (LTC)
Department of Aerospace Engineering
The University of Michigan
Ann Arbor, MI 48109-2118

Driscoll, J. F.
Department of Aerospace Engineering
3004 FXB
The University of Michigan
Ann Arbor, MI 48109-2140

Faeth, G. M.
Department of Aerospace Engineering
3000 FXB
The University of Michigan
Ann Arbor, MI 48109-2140

Law, C. K.
Department of Mechanical and Aerospace Engineering
Princeton University
Princeton, NJ 08544-5263

Dryer, F. L.
Department of Mechanical and Aerospace Engineering
Princeton University
Princeton, NJ 08544-5263

Sung, C. J.
Department of Mechanical and Aerospace Engineering
The Case School of Engineering
Cleveland, Ohio 44106-7222

Moin, P.
Center for Turbulence Research
Building 500
Stanford University
Stanford, CA 94305-3030

Ferziger, J. H.
Mechanical Engineering Department
Stanford University
Building 530
Stanford, California 94305-3030

DISTRIBUTION: (continued)

Egolfopoulos, F. N.
Department of Mechanical Engineering
University of Southern California
3650 McClintock Avenue, OHE 430
Los Angeles, CA 90089-1453

Ronney, P. D.
Department of Aerospace and Mechanical Engineering
University of Southern California
3650 McClintock Avenue, OHE 430
Los Angeles, CA 90089-1453

Williams, F. A.
Center for Energy and Combustion Research (0411)
University of California, San Diego
La Jolla, CA 92093-0411

Bechtold, J. K.
Department of Mathematics
New Jersey Institute of Technology
Newark, NJ 07102

Dibble, Robert
Department of Mechanical Engineering
University of California at Berkeley
Berkeley, CA 94720-1740

Chen, J.-Y.
Department of Mechanical Engineering
University of California at Berkeley
Berkeley, CA 94720-1740

Gore, Jay
School of Mechanical Engineering
Purdue Univeristy
West Lafayette, IN 47907-1288

Pitz, William J.
L-014
Lawrence Livermore National Laboratories
Livermore, CA 94551

Westbrook, Charlie
L-014
Lawrence Livermore National Laboratories
Livermore, CA 94551

Pitz, Robert W.
Vanderbilt University
Box 54, Station B
Nashville, Tennessee 37235

Blevins, Linda G.
National Institute of Standards & Technology
Building & Fire Research Laboratory
Bldg. 224/Rm B360
Gaithersburg, MD 20899-0001

Takeno, Tadao
Department of Mechanical Engineering
Nagoya University Chikusa-ku
Nagoya
464-01
JAPAN

Mahalingam, Shankar
Department of Mechanical Engineering
University of Colorado at Boulder
Boulder, CO 80309-0427

MS9054	W. J. McLean, 8300
MS9055	J. A. Miller, 8353
MS9042	G. H. Evans, 8345
MS9051	J. F. Grcar, 8345
MS9051	A. E. Lutz, 8345 (10)
MS9051	L. A. Rahn, 8351
MS9051	W. T. Ashurst, 8351
MS9051	R. S. Barlow, 8351
MS9051	J. H. Chen, 8351
MS9051	T. Echekki, 8351
MS9051	J. H. Frank, 8351
MS9051	J. C. Hewson, 8351
MS9051	C. A. Kennedy, 8351
MS9051	A. R. Kerstein, 8351
MS9051	W. Kollmann, 8351
MS9051	H. N. Najm, 8351
MS9051	P. H. Paul, 8351
MS9051	R. W. Schefer, 8351

3	MS 9018	Central Technical Files, 8940-2
1	MS 0899	Technical Library, 4916
1	MS 9021	Technical Communications Department, 8528/Technical Library, MS 0899, 4916
1	MS 9021	Technical Communications Department, 8528 For DOE/OSTI
1	MS 0161	Patent and Licensing Office, 11500