Artificial Neural Network Implementation of Chemistry with pdf Simulation of H_2 / CO_2 Flames

F. C. CHRISTO,* A. R. MASRI, and E. M. NEBOT

Department of Mechanical and Mechatronic Engineering, University of Sydney, NSW 2006, Australia.

A novel approach using artificial neural networks for representing chemical reactions is developed and successfully implemented with a modeled velocity-scalar joint pdf transport equation for H_2/CO_2 turbulent jet diffusion flames. The chemical kinetics are represented using a three-step reduced mechanism, and the transport equation is solved by a Monte Carlo method. A detailed analysis of computational performance and a comparison between the neural network approach and other methods used to represent the chemistry, namely the look-up table, or the direct integration procedures, are presented. A multilayer perceptron architecture is chosen for the neural network. The training algorithm is based on a back-propagation supervised learning procedure with individual momentum terms and adaptive learning rate adjustment for the weights matrix. A new procedure for the selection of training samples using dynamic randomization is developed and is aimed at reducing the possibility of the network being trapped in a local minimum. This algorithm achieved an impressive acceleration in convergence compared with the use of a fixed set of selected training samples. The optimization process of the neural network is discussed in detail. The feasibility of using neural network models to represent highly nonlinear chemical reactions is successfully illustrated. The prediction of the flow field and flame characteristics using the neural network approach is in good agreement with those obtained using other methods, and is also in reasonable agreement with the experimental data. The computational benefits of the neural network approach over the look-up table and the direct integration methods, both in CPU time and RAM storage requirements are not great for a chemical mechanisms of less than three reactions. The neural network approach becomes superior, however, for more complex reaction schemes.

NOMENCLATURE

D_j	diameter of fuel jet (mm)
E_p	error function
L	number of hidden layers
N	number of input–output training samples
N_I	number of input variables
N_J	number of reactive scalars
N_p	number of Monte Carlo particles
N_t	number of time scales
N_ξ	number of discretized nodes in mixture fraction space
K_l	number of neurons in the lth hidden layer
T	temperature (K)
t	time (s)
u	axial velocity component (m/s)
x	distance from nozzle exit (mm)
Y	mass fraction
W	molecular weight

Greek Symbols

α	momentum factor
Γ	specific molar abundance ($\equiv Y/W$)
η	learning rate factor
ε	convergence rate
ξ	mixture fraction
ϕ	bias vector
ω	weight matrix

Subscript

i	value for species i
s	value at stoichiometric

Superscripts

-	conventional average

Other

ANN	artificial neural network
CPU	central processing unit
DI	direct integration
FLOP	floating point operations
LUT	look-up table
MC	Monte Carlo
MLP	multilayer perceptron
ODE	ordinary differential equation
RAM	random access memory

* Author to whom correspondence should be addressed.

0010-2180/96/$15.00
SSDI 0010-2180(95)00250-2

COMBUSTION AND FLAME 106:406–427 (1996)

Published by Elsevier Science Inc.

INTRODUCTION

Complex chemical kinetics and significant interactions between turbulence and chemistry are intrinsic to combustion processes which occur in many engineering applications. Understanding such issues and developing the capabilities to simulate them is essential for improving the thermal efficiency of power production, reducing combustion noise, pollution control of propulsion and industrial systems, fire prevention and safety, household and industrial heating [1, 2]. The steady and impressive improvement in the speed of computers and the increase of memory capacity has led to significant advances in numerical techniques for solving complex problems of nonreacting and reacting fluid flows [3]. Numerical solutions complement experimental and theoretical developments by providing an alternative cost-effective means for simulating real combustion conditions, and offer the means of testing theoretical advances for conditions unavailable or difficult to obtain by experimental methods. Despite the recent outburst in computing power, however, performing Direct Numerical Simulation (DNS) of reacting systems remains only a research tool limited to simple flows with low Reynolds numbers [4]. Modeling will, therefore, remain the only tool for closing the transport equations and allowing the numerical simulations of practical reacting systems.

The main difficulty in modeling chemically reacting turbulent flows using the Reynolds decomposition approach for the governing equations, stems from the closure problem associated with nonlinearities in the chemical source term. While various standard models (e.g., $\kappa-\epsilon$ and Reynolds stress) have been successful in achieving satisfactory closure for the turbulent terms in nonreacting flows, they fall short in their capabilities to obtain adequate closure for the species and the energy equations in chemically reacting flows. Achieving closure for the chemical source term in standard approaches which use the conventional (or Favre) averaged transport equations for species is practically impossible due to the high nonlinearity of the reaction rates with temperature.

A number of alternative approaches have been developed to overcome the closure problem of the chemical source term and allow the use of detailed chemistry. The laminar flamelet concept [5–8] is the most prominent and has been used in many situations for computing the structure of turbulent nonpremixed flames. It is based on the concept of describing turbulent flames as an ensemble of laminar flames (flamelets). The validity of such an approach, however, is limited to situations where the thickness of the reaction zone is smaller than a representative turbulence length scale, generally taken as 5–10 times the Kolmogorov length scale. Such conditions however, are not often encountered in combustion processes of practical interest. A more recent approach which allows the use of detailed chemistry is the Conditional Moment Closure (CMC) method, developed independently by Bilger [9] and Klimenko [10]. It is based on using moments conditional on the conserved scalar, mixture fraction, to eliminate the nonlinear dependence of the reaction rates. The main advantage of the CMC approach is its applicability to model systems of arbitrarily complex chemistry within realistic computational cost. The CMC method has been demonstrated for single scalar conditioning by Smith et al. [11]. The radical concentrations and NO_x formation in a turbulent jet diffusion flame of H_2–air mixture have been predicted successfully. This application assumes that the departures from conditional mean values are relatively small which implies that a singly conditioned form of the CMC method is applicable. To simulate lifted flames and flames near extinction, conditioning on mixture fraction as well as a reaction progress variable is necessary, but this method has not yet been numerically implemented [12].

The joint velocity-composition probability density function (pdf) approach coupled with Monte Carlo (MC) methods has been extensively used for modeling turbulent reacting flows [13–15]. It has the main advantage of representing the chemistry and the convective terms in closed form, irrespective of the complexity and non-linearity of the reaction scheme. Consequently, the gradient transport assumption, which is used in standard finite

difference schemes, is avoided. This is an important gain because the validity of the gradient diffusion hypothesis is arguable [16, 17], especially for modeling turbulent combustion where such assumption may lead to erroneous prediction of stresses and fluxes in zones of intense chemical reaction [18]. The pdf approach may be used to simulate finite rate chemistry effects in applications where the flamelet assumption applies as well as in others where reaction zones are broad and distributed.

The use of detailed chemistry in turbulent combustion simulations is computer intensive, regardless of the numerical scheme that is used for solving the governing equations. The development of reduced chemical kinetic mechanisms [19] has greatly assisted in this regard by reducing the computational efforts while preserving a realistic representation and allowing the finite rate chemical kinetics to be considered. The conventional reduction methodology is based on systematically eliminating reactions that are very close to being in partial equilibrium and species that have steady-state concentrations [20]. More recently, Maas and Pope [21, 22] developed an automated general procedure for developing reduced chemical kinetic mechanisms, which is referred to as the Intrinsic Low-Dimensional Manifolds (ILDM) method. The ILDM approach is based on decoupling the reactions with short time scales from the slower reactions so that local equilibrium can be assumed for the fast reactions.

In simulating parabolic jet flows using the joint pdf/MC approach, the main computational cost lies in providing a realistic representation of the chemistry. This constraint necessitates the use of reduced kinetic mechanisms. This approach has been adopted in computing the structure of piloted diffusion flames [23]. The calculation of the composition changes due to chemical reaction is generally performed by one of two methods: (i) Direct integration over the reaction time for each composition. This technique requires extensive computer time and is well suited for parallel processing. (ii) Using look-up tables that store the incremental change in composition due to chemical reaction. This approach requires large computer memory and storage capabilities. Both approaches have significant limitations which are discussed in the following sections. Efforts are currently being invested to establish new methods for representing the chemistry adequately without imposing unrealistic computational penalties.

Recently, a new approach which is based on repro-modeling of the chemistry has been introduced. The idea of repro-modeling is to establish a functional relation, in the form of high-order multivariate orthonormal polynomials, between the compositions and the reaction rates or the incremental changes due to chemical reaction [24]. This technique has been successfully applied to describe the combustion of wet CO–air mixtures using simple chemistry models [25]. The full potential and limitations of this method, however, are yet to be determined and it is, therefore, not included in the present work.

Artificial Neural Network (ANN) modeling is one of the most rapidly expanding fields of research and application, attracting interest from a wide variety of disciplines. The engineering community is involved in applying this massively parallel distributed information technique to image processing, speech recognition, electronic noise filtering, telecommunication, traffic management, and chemical process control. The application is extended here to turbulent combustion. A neural network system is a highly parallel dynamic structure. The approach of neural modeling is to train the computing system to capture the guiding principles that underly the physical problem, and to generate a model. The model is aimed at producing an approximated version of the real system while retaining the same general behavior [26].

In this paper a novel approach using an artificial neural network to represent the chemistry in turbulent diffusion flames is introduced. It is aimed at developing a mathematical formulation capable of representing the changes in composition due to chemical reaction. The chemistry is described by three-step and five-step mechanisms. The objectives of this paper are: to demonstrate the use of neural networks in modeling chemical reactions;

to optimize the architecture of the network to manageable dimensions while maintaining accurate representation of the chemistry; and to implement the ANN model into a joint pdf/Monte-Carlo simulation of a pilot-stabilised turbulent jet diffusion flame. The potential savings in computer memory and in computation time requirements compared to the tabulation method and the direct integration approach is also illustrated.

STANDARD APPROACHES TO CHEMISTRY REPRESENTATION

In the following section, a brief review of the advantages and limitations of the existing methods used for representing the chemistry in pdf/Monte Carlo simulations of turbulent combustion is presented.

Direct Integration Method

In the direct integration (DI) approach the composition changes due to chemical reactions are calculated by solving a system of Ordinary Differential Equations (ODEs) which are deduced from the chemical kinetic mechanism. This method for representing the chemistry is the most accurate, and in principle its precision is limited only by the round-off error of the computing machine. Basically, the Random Access Memory (RAM) requirement to perform DI calculations is affordable and practically considered trivial. For a given reaction time Δt, the change in composition $\Delta\Gamma_j$, for an independent reactive scalar j, is determined by integrating the reaction rates over that specific time scale:

$$\Delta\Gamma_j = \int_0^{\Delta t} \frac{d\Gamma_j}{dt}\, dt, \qquad (1)$$

where Γ_j is the specific molar abundance of species j, defined as the ratio of mass fraction to the molecular weight of species j ($\Gamma_j \equiv Y_j/W_j$). The reaction rates $d\Gamma/dt$ of the reactive scalars are determined by the details of the chemical kinetics mechanism. The incremental changes, according to Eq. 1, usually produces a set of stiff ordinary differential equations that are computationally expensive to solve. The origin of stiffness lies in the fact that kinetic systems usually have a broad spectrum of time scales, which could be ranging from 10^{-9} to 10^{2} s. Numerical integration of a stiff system of ODEs is dominated by the smallest time scale (to ensure numerical stability), and consequently intensive computations are involved. The Central Processing Unit (CPU) time becomes intolerable, and hence imposes a severe limitation on the applicability of the DI scheme for modeling purposes.

Tabulation Method

The most common approach used for representing the chemical source term is the look-up table (LUT) method. The benefit of using a look-up table is to reduce the computation by preventing repeated integration of the same set of ODEs. The incremental changes in composition due to chemical reaction over a certain time are first computed by solving the set of ODEs (Eq. 1), which are then stored in a look-up table that can be accessed during the calculations.

The look-up table for the reaction increments should contain the composition changes covering the full range of compositions that are accessible by the Monte Carlo simulation. It should also represent these changes (for each mixture) over a range of reaction time scales. The tabulation procedure is described in details elsewhere [23].

The resolution of the table of composition increments is restricted by the available computer memory, due to its strong dependence on the number of reactive scalars considered. The required memory (RAM) becomes extremely large as the number of reactions exceeds five, even for a modest table resolution. It should be mentioned here that, in practice, some compositions in the table are never accessed during the simulation, though these compositions satisfy the atom balance equations. Therefore, the "useful" domain of the reaction table which is accessed during the simulation is usually far smaller than the actual size of the tabulated space. This problem emerges due to the lack of a priori knowledge

about the most accessible compositions. Norris [27] attempted to produce a tabulation space with a nonuniform grid that is denser in regions where reaction rates are likely to be high. The method has been successfully used for simple chemistry and achieved an impressive reduction in the table size. However, it came at the expense of significant increase in the computational time due to the iterative nature of the procedure involved in structuring the mesh. Storage requirements remain, however, very high if sufficient resolution is to be achieved.

ARTIFICIAL NEURAL NETWORKS

An optimum numerical scheme for representing the chemistry should offer the advantages of both the direct integration scheme and the look-up table approach. That is, achieving accurate representation of the chemistry within an affordable memory framework (as in the DI scheme) and within a realistic computational time (as in the LUT method). In the following sections, we introduce and explore the capabilities of the artificial neural network approach to approximate chemical reactions and then demonstrate its applicability in turbulent combustion modeling.

The term artificial neural network (ANN) refers to any computing architecture that consists of massively distributed interconnected simple units and processors. It is a highly parallel dynamic structure. It is originated from the attempts to emulate the biological nervous system in the human brain. The approach of neural computing is to train the computing system to capture the guiding principle that underly the physical problem, and to produce a model. The model aimed at producing a simplified version of the real system while retaining the same general behavior [26, 28, 29]. Basically, the neural network model can be thought of as a "black box," which internally consists of simple units of one and two dimensional matrices with massive interconnections of the elements of sequential units.

The most important factors involved in developing a neural network for any application include the choice of the architecture for the network, the selection of the learning algorithm, and establishing the initial training samples. Several architectures and teaching algorithms are presented by Lippmann [28]. In principle, it has been proven that any continuous function can be uniformly approximated by a neural network model with only one internal (hidden) layer [29, 30]. Unfortunately, there are no definite guidelines for optimizing the size of the network. Determining the appropriate number of layers and the number of neurons (elements) in each layer is effectively an iterative trial and error procedure. The inherent capabilities of ANN to approximate nonlinearfunctions, makes it an attractive choice for modeling chemical reactions that are highly nonlinear functions of temperature and species concentrations.

Multilayer Perceptron Architecture

For multilayer perceptron (MLP) networks, the inputs propagated through the network, and the output of each neuron is evaluated according to:

$$y_i^l = f\left(\sum_{j=1}^{K_{(l-1)}} w_{ij}^l y_j^{l-1} + \phi_i^l \right)_{i=1\ldots K_l,\, l=1\ldots L}, \tag{2}$$

where y_i^l is the output of the ith neuron of the lth layer, (w_{ij}^l) is the weight value of connection between the jth node of the $(l-1)$ layer and the ith neuron of lth layer, ϕ_i^l is the bias value of the ith neuron of the lth layer. The nonlinear transfer (activation) function, $f(\cdot)$, should be differentiable and strictly have a positive first derivative. A sigmoidal or hyperbolic-tangent function is commonly used as a transfer function.

Throughout this work, a MLP architecture with two intermediate (hidden) layers has been used, as shown in Fig. 1, each consisting of an equal number of neurons, either 8 or 10. A hyperbolic-tangent (tanh) function has been used as a transfer function.

The Learning Algorithm

The learning algorithm provides the method for adjusting the weights (w_{ij}^l) in the network, and so continually reducing the value of the

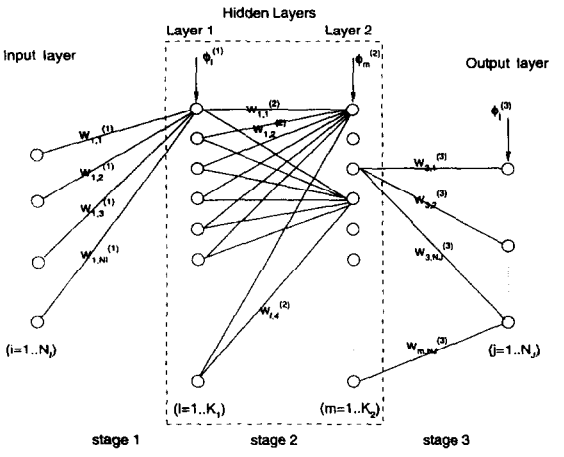

Fig. 1. Structure of multilayer neural network architecture.

error function as the network learns. In general, the error function, E_p is defined as

$$E_p = \sum_i (d_i - y_i)^2, \tag{3}$$

where y_i and d_i are, respectively, the actual network model output and the desired output over the sample space. Since the learning process is guided by knowing the desired output, it is called a "supervised learning" process.

The most popular and successful learning algorithm used to train multilayer neural networks is currently the back-propagation scheme. It is a generalized form of the Widrow–Hoff delta rule [31], which is basically a gradient descent algorithm designed to minimize the error function (E_p) in the weights space (w_{ij}). The Widrow–Hoff delta rule is a self-normalized algorithm, which makes it independent of the magnitude of the training samples. The normalization space for the input and output samples used here is the interval $[-0.9, +0.9]$, which offers faster convergence over the common [0, 1] normalization space, due to its wider dynamic range.

The need for a nonlinear activation function is imposed by virtue of using the back-propagation scheme. The hidden layers cause the actual inputs to be masked off beyond the first input layer, and prevents the network from "learning" because it loses contacts with the original inputs. Therefore, there is no indication on how the weights should be adjusted as the scheme is back-propagating. The back-propagation algorithm adjusts the weights according to the following equation [26]:

$$w_{ij}^l(p+1) = w_{ij}^l(p) + \eta\delta_j y_i + \alpha\left[w_{ij}^l(p) - w_{ij}^l(p-1)\right], \quad 0 < \eta, \alpha < 1, \tag{4}$$

where p is the iteration number and η and α are variable factors that can be adjusted to speed up the convergence of the algorithm. The function δ_j, represent the change in the error function of the jth node with respect to the network inputs. That is,

$$\delta_j \equiv -\frac{\partial E_p}{\partial \Sigma_i w_{ij}^l y_i}. \tag{5}$$

The explicit form of the function δ_j depends on the activation function, and on whether j is an internal or an output node. In general, the functional form of δ_j is given by

$$\delta_j = \begin{cases} f_j'(d_j - y_j) & \text{if } j \text{ is an output node} \\ f_j' \Sigma_i \delta_i w_{ij}^l & \text{if } j \text{ is an internal hidden node.} \end{cases}$$

The function f_j', denotes the derivative of the activation function of the jth node with respect to the total net inputs. That is,

$$f_j' = \frac{\partial y_j}{\partial \Sigma_i w_{ij} y_i}. \tag{6}$$

By virtue of using a gradient descent approach in back-propagation algorithm, the weights are continually adjusted towards a minimum value of errors. This means that the network can be trapped in a suboptimal local minimum far from the global minimum, as shown schematically in Fig. 2, hence resulting in an unsatisfactory model. In general, the use of appropriate momentum and learning rate factors (α and η, respectively) in Eq. 4, reduces the possibility of the algorithm being trapped in local minima, and assists the algorithm to escape out of local minima traps. In practice, selecting a small fixed learning rate factor, η, requires a large number of iterations, while a large factor may cause instability and oscillations. However,

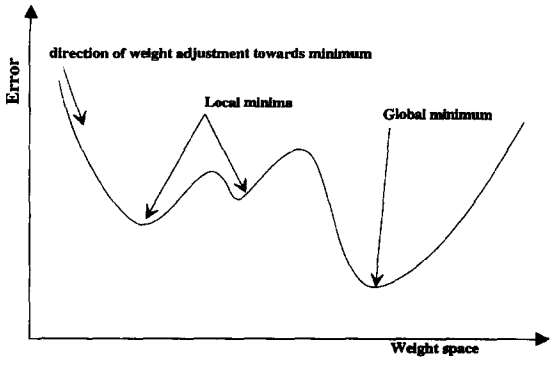

Fig. 2. Schematic illustration of error adjustment in weight space towards minimizing the error value.

continuous adjustment of η usually enables the network to bypass local minima and settle in some deeper minima without large oscillations. Hence, an adaptive learning rate approach seems an appropriate choice. This algorithm adjusts the learning rate for each individual weight (after the overall error is computed) according to the following guidelines:

1. If the overall error is decreasing, then the learning rate (η) is increased by a factor which is greater than one.
2. However, if the overall error shows an increase by more than few percent, then the learning rate (η), is reduced by a factor less than one and the current weight update is rejected.

The factor α in Eq. 4, usually called momentum factor, is useful to get the algorithm out of local minima and is commonly believed to assist in smoothing out oscillations due to a large learning rate [32]. The momentum term adds an extra expression into the weight adaption equation (Eq. 4), which produces weight changes proportional to the current magnitude of the weight's gradient. That is, large changes in the weight will occur if the gradients are currently large, decreasing as the gradients become smaller. Therefore, this term causes the weights to adjust in the same direction as the gradient, hence the name—momentum. However, in some cases, increasing the momentum factor is not enough and the network model obtained is not appropriate. This problem may be overcome by preprocessing the training data set.

Dynamic Randomization Algorithm

The training stage requires samples that are representative of the overall expected working range of the model, which is often difficult to obtain. From the N input–output data samples available for training and testing, R samples ($R < N$) are selected randomly for training. Selecting fixed intervals that are too long, however, may cause the network to reach a local minimum. To reduce the possibility of being trapped in local minima a new algorithm is developed in which the network is trained over fixed intervals with subsets S, of the R training samples ($S < R$), as shown schematically in Fig. 3. In this case the training samples are slightly changed giving the algorithm new information to escape from local minima. The subset S, of the training samples, is generated randomly during the learning process. To achieve smooth behavior of the algorithm, the size of the S-subset should be 70%–80% of the size of the training set (R). This continuous (dynamic) randomization of the subset of the training samples is found to give superior performance, in terms of convergence rate, over similar approach which uses fixed set of preselected training samples.

Network Optimization and Convergence Rate

The optimal performance of back-propagating networks is not severely influenced by the network topology, namely the layering of the neurons. However, networks with two hidden layers are easier to train than single layer

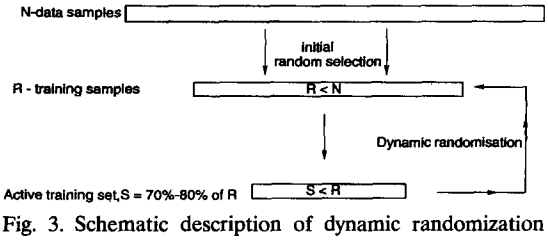

Fig. 3. Schematic description of dynamic randomization procedure for selecting the training samples in neural network.

networks if the number of neurons in the internal layers are balanced [33]. The number of neurons in each hidden layer is determined by an iterative trial and error process until an appropriate convergence of the algorithm is obtained. The following numerical experiments are carried out to estimate the convergence rate sensitivity to the number and to the distribution of the neurons in the hidden layers. The convergence rate, ε is defined as a normalized error function, as follows:

$$\varepsilon \equiv \frac{E_p}{E_{p,0}}, \tag{7}$$

where $E_{p,0}$ is the value of the error function (Eq. 3) at the end of the first ten iterations. Two sets of sensitivity experiments are performed. In the first set, four input and three output variables are used, while the second set uses six input and five output variables. The training sets are generated, respectively, from a three-step and five-step chemical kinetic mechanism for a H_2/CO_2 fuel mixture. Such selection is aimed at firstly, examining the ability of neural networks to model different and complex chemical mechanisms, and secondly to evaluate, conclusively, the effects of the neural network structure on the convergence rate and on the network's dimensions.

In the first set of experiments, the sensitivity of the network to the number of neurons in a two-hidden-layers configuration, is examined. A total of twelve neurons arrangements are considered. In the first configuration, an equal number of neurons are assigned in each hidden layer. Four different clusters with 6, 8, 10, and 12 neurons in each layer, are examined. In general, it is found that the first 200–300 iterations usually give a very good indication about the convergence of the algorithm (or the lack of it). Figure 4 shows the convergence rate versus the number of iterations for the four different configurations. It is clear that the number of neurons does not have a significant effect on the convergence rate. In the second configuration, the effect of inequality in the number of neurons in the hidden layers on the convergence rate is evaluated. The results, shown in Fig. 5, indicate that for networks with a balanced number of neurons, the convergence rate is almost insensitive to the number of neurons. It is also insensitive to the ordering of the clusters in the hidden layers. The exception occurs if the number of neurons are significantly different between the intermediate layers, e.g. clusters (1, 12) and (12, 1). This situation causes a "bottleneck" junction of information that severely affects the convergence of the algorithm. It should be mentioned that the transition to such a condition occurs gradually and is noticeable during the training stage.

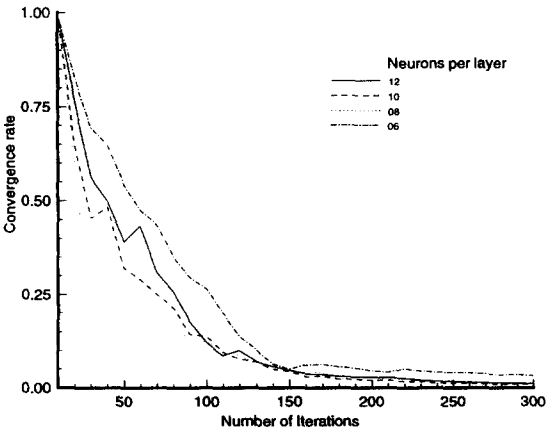

Fig. 4. Convergence rate of MLP network with balanced number of neurons in the hidden layers, for different layers dimensions.

The same sensitivity analysis described above is repeated here, but for a larger number of variables. The neural network is trained using six inputs and five output variables which corresponds to a five-step chemical kinetic mechanism. The effect of the number of neurons and their arrangements, on the convergence rate is investigated and the results are shown in Fig. 6. These results reinforce the observation stated earlier regarding the systematic and gradual drift of the convergence toward an obstructed (bottleneck) situation as the number of neurons in the hidden layers becomes largely unbalanced. The optimal configuration that leads to the fastest convergence in both the three variables, and five variables cases is where the number of neurons is equal in both layers.

It becomes clear during the optimization process of the network's architecture that special emphasis should be placed on the quality

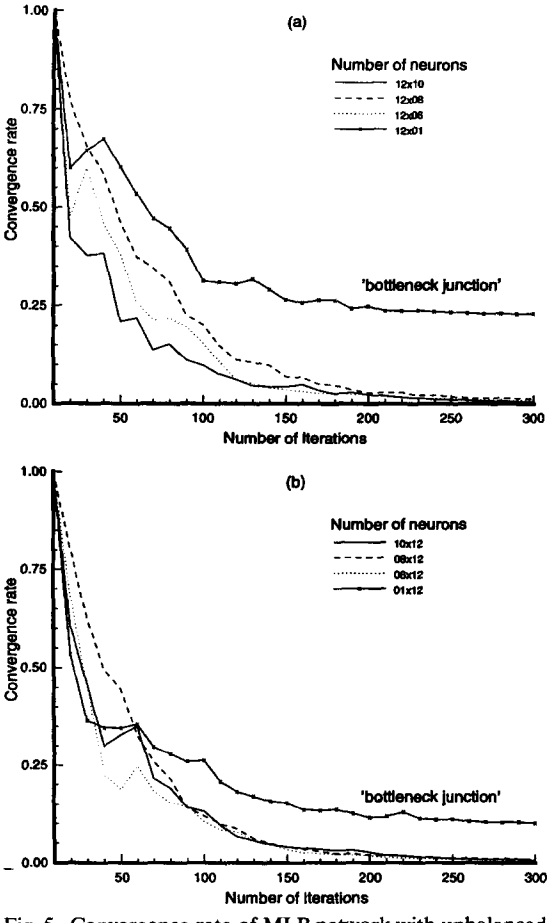

Fig. 5. Convergence rate of MLP network with unbalanced number of neurons in the hidden layers: (a) various layer sizes (b) reversed order of the layers' dimensions (chemistry is represented using three-step reduced kinetic mechanism).

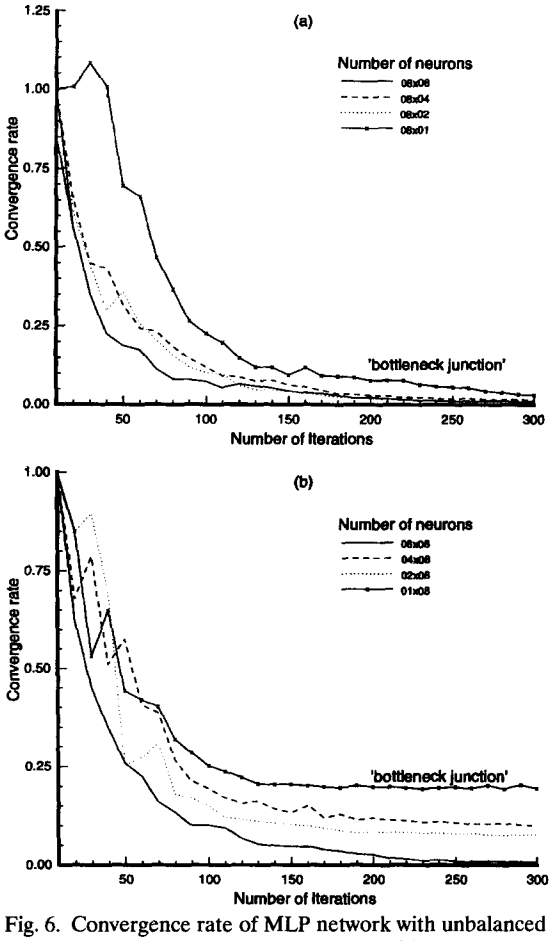

Fig. 6. Convergence rate of MLP network with unbalanced number of neurons in the hidden layers: (a) various layer sizes (b) reversed order of the layers dimensions (chemistry is represented using five-step reduced kinetic mechanism).

of the training samples. The sample's quality, according to our definition, is a measure of the sensitivity of an output to a small variation in one of the corresponding input variables. A highly sensitive sample is characterized by a spike-like output of distinctly large magnitude in comparison to the other samples, and is considered a low-quality sample. The existence of a large number of low quality samples in the training set causes large oscillations and instabilities, and makes it difficult and sometimes impossible to obtain satisfactory convergence. The origin of low-quality samples may be attributed to having physically unrealistic compositions (hence unrealistic incremental changes in the reaction space) in the training set. It may also be due to the highly nonlinear nature of chemical reactions. At present, there is no effective algorithm to ensure that an unrealistic mixture is not included in the training set. A new technique which will be reported later is currently being investigated to eliminate unrealistic samples, and to produce a more representative training set.

FLOATING POINT OPERATIONS AND STORAGE REQUIREMENTS

A numerically optimized scheme is one that results in minimum storage and CPU time requirements. The neural network model requires very little memory storage, and hence the efforts in optimizing the performance of

the neural network model is to focus on reducing CPU time, and hence on reducing the number of FLoating point OPerations (FLOP) that are needed to compute the changes in a given composition. The following is a first-order estimate of the number of FLOP which is required by the LUT and ANN methods.

For the LUT method, the number of FLOP, F_{LUT}, is the number of arithmetical operations required to perform multidimensional linear interpolation in the look-up table. This is usually an algorithm dependent value. For the algorithm used here, a simple but rough estimate has been empirically derived for F_{LUT} as

$$F_{\mathrm{LUT}} \approx 10 \times 2.5^{N_J}, \tag{8}$$

where N_J is the number of input variables, which represent the number of reactive scalars. Equation 8 indicates that the number of arithmetical operations increase sharply as the number of reactive species increases beyond say, 10 scalars.

The number of FLOP required by the neural network model depends also on the dimensions of the network. In our case, it is assumed that any basic arithmetical operation (i.e., addition, subtraction, multiplication or division), as well as evaluation of the nonlinear activation function (tanh), requires a single floating point operation. The CPU time that is required to perform these operations has been estimated by numerical experimentation. It is found that the basic operations of addition, subtraction, multiplication, and division require equal CPU time while the hyperbolic-tangent requires about 3–4 times the CPU time of a basic operation.

The overall number of FLOP required to perform neural model calculations, F_{ANN}, is then approximated as

$$F_{\mathrm{ANN}} \approx (2N_I + 2)K_1 + (2K_L + 1)N_J + \sum_{l=1}^{L-1} (2K_l + 2)K_{l+1}, \tag{9}$$

where, N_I, N_J are the number of input and output variables, respectively. K_l is the number of neurons in the lth intermediate layer, and L is the number of hidden layers ($L \geq 2$).

To examine the ratio, F_r, between the number of FLOP that is required to carry out the LUT calculations and that required in the neural network modeling, we considered a MLP network with two hidden layers ($L = 2$), each consisting of an equal number of neurons K (i.e., $K_l = K$, $\forall l = 1 \ldots L$). It is also known that $N_I = N_J + 1$. Combined with Eqs. 8 and 9, this gives:

$$F_r \equiv \frac{F_{\mathrm{ANN}}}{F_{\mathrm{LUT}}} = \left(\frac{1}{5 \times 2.5^{N_J}}\right)K^2 + \left(\frac{2N_J + 3}{5 \times 2.5^{N_J}}\right)K + \frac{N_J}{10 \times 2.5^{N_J}}. \tag{10}$$

Figure 7 shows the FLOP ratio, F_r (given by Eq. (10), against the number of neurons in each of the hidden layers, for different number of variables which corresponds to a different number of chemical reactions. An interesting behavior of the FLOP ratio is observed in Fig. 7, showing a systematic reduction in F_r as the number of reactive scalars (or reaction steps) in the chemical mechanism increases, regardless of the number of neurons. Another important outcome that can be drawn from the functional form of Eq. 10, is its finite asymptotic behavior, i.e., as $N_J \to \infty$, the FLOP ratio, $F_r \to 0$. The results in Fig. 7 can also be used as a parameter to obtain optimum dimensions for the network that ensure that $F_r \leq 1$. For

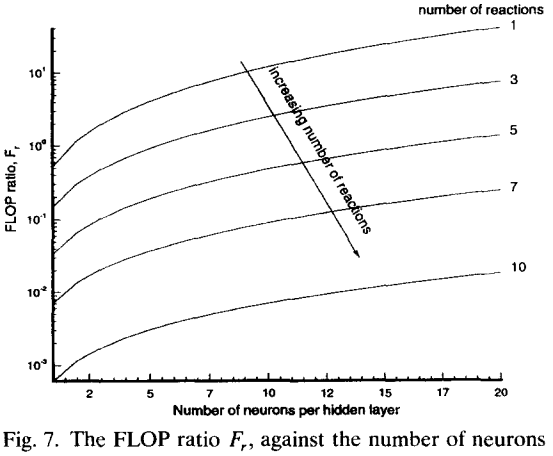

Fig. 7. The FLOP ratio F_r, against the number of neurons in a two hidden layers network, for different reduced mechanisms.

example, in modeling chemical kinetic mechanisms that include more than five reactions, any number of neurons less than 20 (per hidden layer) would confirm that F_r is smaller than unity. It is also interesting to note that for a given network size, the rate in which the FLOP ratio F_r is decreasing becomes higher, as the number of reactions becomes large (say > 5), and consequently less neurons may be used.

Memory Allocation Ratio

The required memory storage for the reaction table in single precision accuracy, S_{LUT} (bytes), is given by

$$S_{\mathrm{LUT}} = 4N_J N_\xi N_t \left[\prod_{i=1}^{N_J} N_i\right], \tag{11}$$

where N_J is the number of reactive scalars, while N_ξ, N_i, $(i = 1 \ldots N_J)$ are the number of discretised nodes of the mixture fraction, and the tabulated scalars, respectively. N_t is the number of reaction time intervals. The mixture fraction space is usually divided into a large number of nonuniformly distributed nodes, with more nodes assigned around the stoichiometric mixture fraction, ξ_s. This is because the reaction rates are very sensitive to composition changes in this vicinity and a fine grid may assist in capturing the reaction details.

The required computer storage for the neural network model, S_{ANN} (bytes), is given by

$$S_{\mathrm{ANN}} \approx 4N_t\left[(N_I + 1)K_1 + \sum_{l=1}^{L-1} K_{l+1}(K_l + 1) + (N_J + 1)K_L\right], \tag{12}$$

where K_l is the number of neurons in the lth hidden layer, and N_t is the number of modeled reaction time intervals.

The memory ratio M_r ($\equiv S_{\mathrm{ANN}}/S_{\mathrm{LUT}}$), is a measure of the storage requirement (for the chemistry term) of the ANN models relative to that required by the LUT method. Using Eqs. 11–12, and inserting $N_I = N_J + 1$, the ratio M_r, is given by

$$M_r \equiv \frac{S_{\mathrm{ANN}}}{S_{\mathrm{LUT}}} = \frac{(N_I + 1)K_1 + \sum_{l=1}^{L-1} K_{l+1}(K_l + 1) + (N_J + 1)K_L}{N_J N_\xi \left[\prod_{i=1}^{N_J} N_i\right]}. \tag{13}$$

To examine the asymptotic behavior of the ratio M_r, as the number of reactive scalars N_J increases, we assumed uniform composition grid dimensions and time intervals, i.e., $N_\xi = N_i = n$, $\forall i = 1 \ldots N_J$, and considered an equal number of neurons, $K_l = K$, $\forall l = 1 \ldots L$. Equation 13, then simply becomes

$$M_r = \frac{K[2N_J + 3 + (1 + K)(L - 1)]}{N_J n^{N_J + 1}} \tag{14}$$

$$\rightsquigarrow \lim_{N_J \to \infty} M_r \to 0.$$

Equations 13 and 14 demonstrate the impressive memory savings of the neural model approach in comparison to the look-up table method. For a five-step reaction mechanism, $N_J = 5$, and assuming a modest table resolution, $n = 10$, and considering a two-hidden-layers network configuration ($L = 2$) with $K = 10$, the memory ratio, $M_r = 5 \times 10^{-5}$. This implies that the storage requirement for the look-up table method is 20,000 times that of the neural network approach.

The overall memory allocation for the pdf/Monte Carlo simulation is determined by the storage requirements of the chemistry term, and by the amount of core memory, M_{MC}. The core memory is used to store the flow field parameters, species concentrations, thermodynamics properties, etc... (13 variables altogether), and simply approximated as:

$$M_{\mathrm{MC}} \approx 4N_p(13 + N_J), \tag{15}$$

where N_p is the number of Monte Carlo particles. The ratio between the core memory, M_{MC}, and the memory requirement for the chemistry term of the LUT method, R_{LUT}, and the ANN

model, R_{ANN}, is easily obtained (using Eqs. 11 and 12):

$$R_{\mathrm{LUT}} \equiv \frac{S_{\mathrm{LUT}}}{M_{\mathrm{MC}}} = \frac{N_\xi N_t N_J \left[\prod_{i=1}^{N_J} N_i\right]}{N_p(13 + N_J)}. \tag{16}$$

$$R_{\mathrm{ANN}} \equiv \frac{S_{\mathrm{ANN}}}{M_{\mathrm{MC}}} = \frac{N_t\left[(N_I + 1)K_1 + \sum_{l=1}^{L-1} K_{l+1}(K_l + 1) + (N_J + 1)K_L\right]}{N_p(13 + N_J)}. \tag{17}$$

Equation 16 shows that as the number of reactive species increases, the ratio R_{LUT} continues to increase indefinitely implying that the chemistry term totally dominates the memory requirement for high number of variables (N_J):

$$\lim_{N_J \to \infty} R_{\mathrm{LUT}} \to \infty. \tag{18}$$

However, for the neural network case (Eq. 17), the behavior of the ratio R_{ANN} is constrained and approaches a finite limit as the number of reactive species increases:

$$\lim_{N_J \to \infty} R_{\mathrm{ANN}} = \frac{N_t(K_1 + K_L)}{N_p}. \tag{19}$$

For typical values of N_t, K_l, and N_p, which are used in actual simulations, the ratio R_{LUT} (according to Eq. 16) is less than unity only for cases with a small number of variables (reactive scalars), but then it increases sharply as the number exceeds three. The ratio R_{ANN}, however, remains practically far less than unity, irrespective to the number of reactive species implying that storage requirements due to chemical term remain small regardless of the number of variables.

Performance Ratio

A performance analysis is carried out to evaluate the FLOP and CPU time requirements of the ANN and LUT methods, and also to examine the accuracy of the mathematical formulation of the FLOP ratio term, F_r (Eq. 10). The computational time (CPU) is related to the number of FLOP, but it is not possible to determine the explicit functional form of such dependence. This is a machine-dependent relation and, therefore, the CPU time ratio is determined empirically. The CPU time, memory, and FLOP demands for each method required to compute the composition changes of a large number of fluid samples of different compositions ($> 45{,}000$) over a fixed reaction time, are recorded and compared with theoretical values. The analysis is repeated for three-step, four-step, and five-step chemical mechanisms.

The actual number of FLOP required by the ANN is similar to the predicted value, i.e. using Eq. 9. The FLOP formula (Eq. 8) for the LUT method, however, is less accurate and shows an absolute relative error of about 20% when compared with the actual number of FLOP.

Table 1 shows the actual FLOP, memory, and CPU time ratios of both methods for different chemical kinetic mechanisms. It also lists the CPU time ratio between the ANN and DI scheme. The results are also illustrated in Fig. 8, and show that all ratios continue to decrease as the number of variables increase. That is, as the number of chemical reactions increase, the ANN approach becomes compu-

TABLE 1

Performance Ratio of the Number of Floating Point Operations, Memory Size, and CPU Time Required by Various Methods, for Different Number of Variables (or Reactions): ANN = Artificial Neural Network, LUT = Look-up table, DI = Direct Integration

Number of Variables (or Reactions)	FLOP Ratio (ANN/LUT)	Memory Ratio (ANN/LUT)	CPU Time Ratio (ANN/LUT)	CPU Time Ratio (ANN/DI)
3	1.95	8.74×10^{-4}	1.25	0.0625
4	0.74	8.32×10^{-4}	0.44	0.0355
5	0.22	5.63×10^{-4}	0.14	0.0095

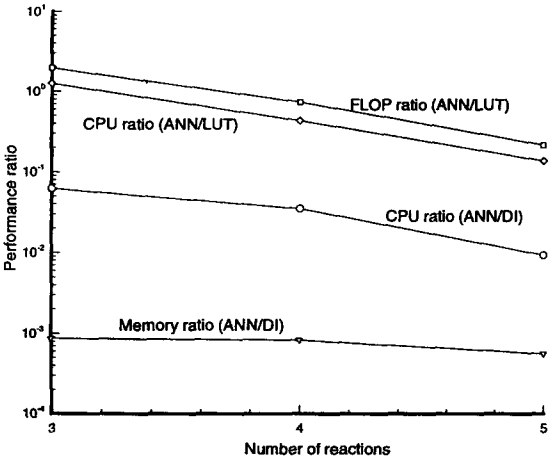

Fig. 8. The FLOP, memory, and CPU time ratios against the number of steps in the kinetic mechanism (ANN = artificial neural network, LUT = look-up table, DI = direct integration).

tationally more efficient than the LUT method in both memory storage and CPU time requirements. Furthermore, the rate by which the FLOP and the CPU time ratios are decreasing is similar. The results also show a strong, and almost constant correlation (of ~ 1.5) between both ratios. The figure clearly demonstrates that for a five variables system the saving in computation time of the ANN approach is about two orders of magnitude over the DI scheme.

IMPLEMENTATION OF NEURAL NETWORKS WITH PDF

The velocity-scalars joint pdf Eulerian transport equation has been derived for inhomogeneous, variable-density, low Mach number gaseous flows by Pope [34]. Stochastic models are used to handle the viscous and viscous-diffusive stress tensors, the fluctuating pressure, and the molecular mixing. The details of these models and other deterministic models used for solving the joint pdf equation are well documented in Refs. 35–41.

Chemical Kinetics

The fuel used is a mixture of 53.5% H_2 and 46.5% CO_2 (by volume) which gives a stoichiometric mixture fraction $\xi_s = 0.37$. The piloted burner (developed at The University of Sydney) [42] has a jet nozzle diameter of 7.2 mm and an annulus diameter of 18 mm. The co-flow air is maintained at 15 m/s and the burned pilot gas velocity is 11 m/s. These values are kept unchanged throughout the computations. The detailed chemistry of H_2/CO_2 has been systematically reduced to the following global three-step reactions [43]:

$$CO + H_2O \rightleftharpoons CO_2 + H_2, \quad \text{(I)}$$

$$2H + M \rightleftharpoons H_2 + M, \quad \text{(II)}$$

$$O_2 + 3H_2 \rightleftharpoons 2H + 2H_2O, \quad \text{(III)}$$

A total of six reactive species, CO, CO_2, H_2, H_2O, O_2 and H are involved in the computations. The independent variables that are selected for the tabulation are the mixture fraction ξ, and the three reactive scalars: CO_2, H, and H_2O.

The density and composition tables are produced for the full mixture fraction space ($0 \leq \xi \leq 1$) while the table for composition changes due to chemical reaction is produced only for the reactive domain which is in the mixture fraction range $0.1 \leq \xi \leq 0.6$. This domain is established from laminar calculations for flames of H_2/CO_2 fuels far from, as well as close to, blow-off conditions [44].

Computation Procedure

The joint pdf transport equation is solved by Monte Carlo technique [15], in which the pdf is represented by a large number of stochastic particles. These particles evolve according to the joint pdf transport equation so as to simulate diffusion, convection, chemical reaction, and molecular mixing processes. Chemical reactions are represented exactly in the joint pdf transport equation, while mechanical as well as scalar dissipation require stochastic models. The reaction process is computationally the most expensive and consumes over 70% of the total CPU time required for the Monte Carlo simulation.

The initial conditions at the jet exit plane ($x = 0$) for mixture fraction, mean and rms profiles of velocity are specified for the fuel jet, pilot flame, and the co-flow regions. The mix-

ture fraction, ξ is initially given as unity in the fuel jet, stoichiometric value ($\xi_s = 0.37$) in the pilot, and zero in the air co-flow. For the Monte Carlo simulations, 20,000–50,000 particles are used. The jet velocities investigated are 50 m/s, 80 m/s, and 130 m/s which represent, respectively 19%, 30%, and 49% of the experimental blow-off velocity. The measured jet blow-off velocity for this fuel is ~ 265 m/s, and the flame visible length is 60–70 fuel jet diameters [44].

Neural Model Implementation

The implementation of the neural network model into the joint pdf/Monte Carlo simulation is carried out by modifying the reaction routine to access a neural network model routine for calculating the composition changes (instead of solving ODEs, or accessing a look-up table). For a potentially reactive composition (ξ, Γ_i, $i = 1 \ldots N_J$), the composition changes ($\Delta\Gamma_j$, $j = 1 \ldots N_J$) over a specific reaction time, are obtained by applying the neural network model according to the following algorithm:

$$\Delta\Gamma_j = \sum_{j=1}^{N_J} \sum_{m=1}^{K_1} x_{2,m} w_{j,m}^{(3)} + \phi_j^{(3)} \tag{20}$$

$$x_{2,m} = \tanh\left[\sum_{m=1}^{K_1} \sum_{l=1}^{K_2} x_{1,l} w_{m,l}^{(2)} + \phi_l^{(2)} \right] \tag{21}$$

$$x_{1,l} = \tanh\left[\sum_{i=1}^{N_J} \sum_{l=1}^{K_2} \Gamma_i w_{i,l}^{(1)} + \phi_l^{(1)} \right], \tag{22}$$

where w and ϕ are the weights matrices and bias vectors, respectively. K_1 and K_2 are the number of neurons in the first and second hidden layers, respectively. Superscripts 1, 2, and 3 in Eqs. 20–22 refer to the location before, within, and after the two hidden layers, respectively as shown in Fig. 1. For the cases where the chemistry is presented for multiple time intervals, each interval is handled by a separate neural model (i.e., different w and ϕ values).

While the composition changes are presented by different methods, the density and the properties are obtained using the same density and composition look-up tables. For a three-step reduced mechanism, the dimensions of these tables are small and do not affect the overall memory or the CPU time requirements.

Chemical Reaction Time Scales

In the DI scheme, the composition changes for each particle over a given reaction time Δt, is computed by solving a set of the ODEs as described by Eq. 1. However, the increment changes in the LUT and the ANN methods are represented for predetermined discrete time intervals. Hence, the evaluation of the composition changes over a given reaction time is performed by summing the changes over the tabulated (or modeled) time intervals until the required reaction time is covered. Representation of the chemistry over multiple time intervals, therefore, reduces the computational efforts considerably as the number of repeated computations is reduced. Selecting the appropriate time intervals, however, is not a straightforward procedure as it requires knowledge of the characteristic reaction time scales which are involved in the Monte Carlo simulation.

Tabulating (or developing neural models) for very small time intervals has the advantage of resolving the composition changes very accurately over fine reaction times. Consequently, an extensive number of arithmetical computations are required, leading to a significant increase in CPU time. Alternatively, selecting large time intervals reduces the computational cost but may lead to some errors. That is because the composition changes for the particles with reaction times smaller than the finest tabulated time scale are then evaluated by linear interpolation. The effect of the "linearization" on the accuracy depends on the nonlinearity in the chemical reactions over that specific reaction time. Therefore, gaining knowledge about the reaction time scales of the particles across the flame space, will assist in selecting the appropriate time intervals for tabulation and for training the neural network. A residence-time map is established from a representative pdf/Monte Carlo simulation for jet flames with different velocities ranging from

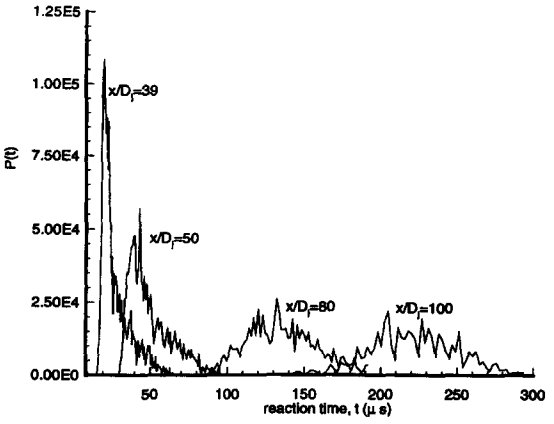

Fig. 9. Typical residence-time pdf of particles, for jet velocity of 80 m/s, at different streamwise locations.

50 m/s to 130 m/s. The residence times obtained at various axial locations down to $x/D_j = 100$, are stored and analyzed.

Figure 9 shows four pdf's for the characteristic reaction time obtained at four axial locations from the Monte Carlo/pdf simulation of a H_2/CO_2 flame with jet velocity $u_j = 80$ m/s. It is clear that the mean characteristic time increases from 15 μs at $x/D_j = 39$ to about 200 μs at $x/D_j = 100$. The variance also increases with increased distance from the nozzle exit. Based on these computed distributions, three time scales of 0.4, 12.5, and 200 μs are selected as the time intervals in which the chemistry is tabulated, or modeled by the neural network. These time scales are not the optimal selection but cover the entire reaction time space, and ensure reasonable computational cost and accurate representation of the chemistry.

COMPUTATIONAL AND PHYSICAL RESULTS

Calculations are performed for different jet velocities but with identical conditions at the pilot annulus and co-flow exit planes. All computations are carried out on a workstation DEC 3000/400-ALPHA platform. Unless stated otherwise all computations are interrupted when axial location $x/D_j = 100$ is reached.

First, a case study has been compiled as a reference base for comparison. Ten time intervals covering the range 0.4–200 μs are used for this case, and the composition increments are stored in a large table. The CPU time required for Monte Carlo simulation for the flames with jet velocities $u_j = 50$ and 80 m/s, are $CPU_{r1} = 37$, and $CPU_{r2} = 92$ min, respectively, and the RAM requirement is ~ 15 megabytes. These requirements are considered reasonable and, therefore, it would be a realistic objective to aim at achieving an equivalent performance from the neural network modeling approach.

The results and discussion are presented in two separate parts. The first part focuses on computational performance and comparison (in terms of CPU time and RAM allocation) between the DI scheme, the LUT method, and the ANN approach. The effects of the number of time intervals which are used for representing the chemistry on the computational performance and the accuracy of the physical results are also investigated. Detailed analysis of memory and computing time requirements is also documented. In the second part, the emphasis is on investigating the applicability and the accuracy of neural modeling to represent chemical reactions, and on comparing the neural model/Monte Carlo predictions with experimental data.

Using the direct integration method with Monte Carlo simulation is computationally very expensive and is well suited for a parallel processing application. A typical example is used here with a four-step chemical kinetic mechanism and a low jet velocity flame with $u_j = 50$ m/s. The computational time requirement for direct integration is 350 min per jet diameter. This is more than an order of magnitude higher than the CPU time requirement using look-up tables. This case is used here as an illustration of the immense CPU time requirement using the direct integration method, and no further calculations are presented using this method.

Computational Performance Analysis

Table 2 lists the computational requirements for flames with jet velocities $u_j = 50$ m/s and 80 m/s, where a single time interval of 0.4 μs has been used for the chemistry. The first

TABLE 2

CPU Time and Memory (RAM) Required for Monte-Carlo Simulation (N_p = 20,000) Using Different Methods to Represent the Chemistry. Jet Velocities u_j = 50 and 80 m/s. Tabulated Reaction Time of 0.4 μs (CPU ratio is based on the case study)

Method	RAM (megabytes)	CPU Time (min)	CPU Ratio	Jet Velocity (m/s)
Direct integration	8.87	1226	33	50
Look-up table	12.85	630	17	50
Neural model	10.00	1230	33	50
Direct integration	8.87	1353	14.7	80
Look-up table	12.85	770	8.4	80
Neural model	10.00	1582	17.2	80

impression does not identify any obvious advantages, either in CPU time or in RAM requirement, in using neural computing over the tabulation approach, or against the direct integration method. This observation is correct for the three-step scheme and for short chemical time scales. The large CPU time needed for the neural model is attributed to using a short time scale to represent the chemistry. With decreasing time scales, both the ANN and LUT methods approach the DI scheme in terms of computational time requirement. The memory requirement for the direct integration and the neural network methods are comparable, and (both) require less memory than the LUT approach since virtually no storage space is reserved for the chemistry term. The advantage in memory reduction using the ANN approach over the LUT method is not fully explored in this case, but it becomes apparent for larger chemical kinetic mechanisms. Detailed analysis on this point is carried out in the following section.

The computations are then repeated under the same operational conditions (jet velocity, number of particles, etc...), but with the chemistry being represented over a larger time increment of 1.6 μs. The computational requirements, given in Table 3, show an improvement in the performance of both methods in comparison to the short time increment shown in Table 2. Both methods recorded comparable reduction in CPU time by a factor of $\sim$ 3.5. This is an encouraging result, however, the CPU time (for both methods) is still about one order of magnitude larger than the reference case. It is also worth mentioning that the simulations (not shown) obtained with both time intervals, are very similar and indicate that the "linearization approximation" for particles with reaction time smaller than 1.6 μs, has no evident effects on the accuracy of the results.

In order to improve the performance of the ANN and the LUT methods, the Monte Carlo simulation is repeated with the chemistry being represented for three time intervals (0.4, 12.5,

TABLE 3

CPU Time and Memory (RAM) Required for Monte-Carlo Simulation (N_p = 20,000) Using Different Methods for Representing the Chemical Source Term. Jet Velocities u_j = 50 and 80 m/s. Tabulated Reaction Time of 1.6 μs (CPU ratio is based on the case study)

Method	RAM (megabytes)	CPU Time (min)	CPU Ratio	Jet Velocity (m/s)
Look-up table	12.85	251	6.7	50
Neural model	10.00	383	10.3	50
Look-up table	12.85	226	2.45	80
Neural model	10.00	293	3.18	80

TABLE 4

Comparison of CPU Time and RAM Requirements Using Different Methods for Handling the Chemistry. Jet Velocities: u_j = 50, 80, 130 m/s. Modeled residence time intervals of 0.4, 12.5, and 200 μs, and N_p = 20,000 Particles (CPU ratio is based on the case study)

Method	RAM (megabytes)	CPU Time (min)	CPU Ratio	Jet Velocity (m/s)
Look-up table	13.31	107	2.9	50
Neural model	10.02	119	3.2	50
Look-up table	13.31	176	1.9	80
Neural model	10.02	170	1.8	80
Look-up table	13.31	242	—	130
Neural model	10.02	204	—	130

and 200 μs). The computational demands which are given in Table 4, and shown also in Fig. 10, illustrate the continuous improvement in the performance of both methods as the number of time intervals increase. The figure also shows that the gap in CPU time between the methods is narrowing, and that the computation time is approaching that of our reference case which uses ten time increments. The reduction in CPU time is larger for the higher jet velocity flames (noticeable with both methods). It is worth noting that for the high jet velocities ($u_j \geq 80$ m/s), the CPU time required by the ANN approach is smaller than that required by the LUT method. The reason for such behavior is not fully known, hence generalization of this observation is not conclusive, and requires further examination.

Characteristics of H_2 / CO_2 Flame

In this section, the results of Monte Carlo simulations are interpreted from a physical viewpoint. The emphasis is on predicting the flame characteristics under different operational conditions. The adequacy of the developed neural network model for representing the chemistry of H_2/CO_2 flame is first discussed.

Neural Model Predictions

Figures 11 and 12 show the output of ANN models trained for three-step and five-step chemistry, respectively, and for a reaction time of 12.5 μs and 0.8 μs. The output over the training set as well as over a set of original inputs is shown for Γ_{CO_2} in Fig. 11, and for Γ_O in Fig. 12. Each sample in these figures represents a different value of ξ and composition within the reactive space (the reactive scalars in the case of the three-step chemical mechanism are: Γ_{CO_2}, Γ_{H_2O}, and Γ_H, and for the

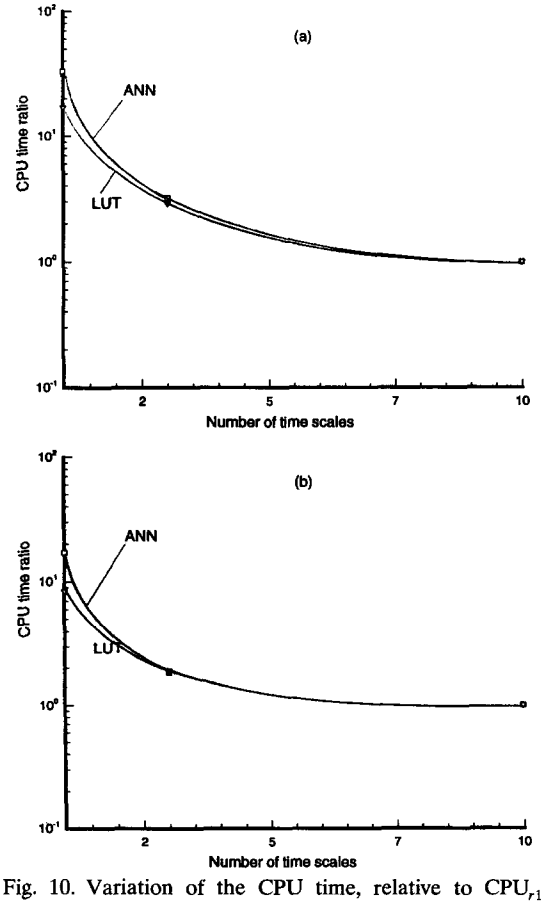

Fig. 10. Variation of the CPU time, relative to CPU_{r1} (= 37 min) for LUT and ANN methods, and different time scales that are used for representing the chemistry of H_2/CO_2 flame with jet velocities: (a) u_j = 50 m/s. (b) u_j = 80 m/s.

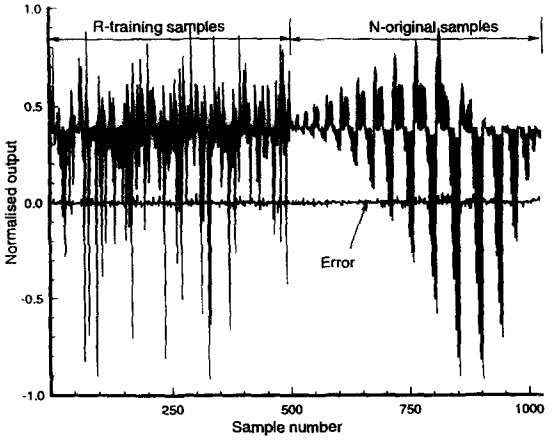

Fig. 11. A typical neural network prediction of (normalized) composition changes of Carbon dioxide, $\Delta\Gamma_{CO_2}$ over reaction time 12.5 μs, and the associated errors (for clarity, the outputs are shifted in the vertical direction by a fixed interval).

five-step case: Γ_{CO_2}, Γ_O, Γ_{OH}, Γ_H, and Γ_{O_2}). The error function $E_p = 0.05$ for Γ_{CO_2} in the three-step case, and $E_p = 0.4$ for Γ_O in the five-step case. In practical terms, an adequate ANN approximation is considered if the error function is smaller than unity ($E_p < 1$).

The figures also show very good predictions over the entire original sample space, indicating good generalization capability of the models. That is, the ability to represent data samples is good over the complete sample space and is not just confined to the training subset domain. The figures highlight a few nodes which exhibit slightly larger errors than the average. These errors can be attributed to the possible existence of unrealistic compositions in the training set. The network is given no indication as to the relevance of these compositions and treats each node as a valid sample. If the number of these invalid samples becomes large, then oscillations and slow convergence are likely, and the network will probably reach a local minimum. The ability of the neural network to represent the chemistry with reasonable accuracy, regardless of some irregularities in the training set, is excellent proof of the intrinsic capability of this approach to approximate highly nonlinear functions.

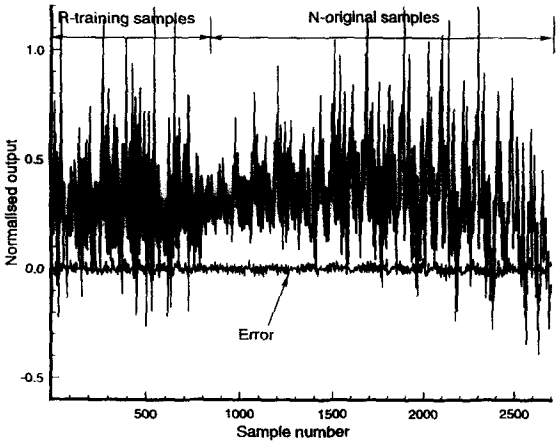

Fig. 12. A typical neural network prediction of (normalised) composition changes of Oxygen radicals, $\Delta\Gamma_O$ over reaction time 0.8 μs, and the associated errors (for clarity, the outputs are shifted in the vertical direction by a fixed interval).

Monte Carlo Simulation

Monte-Carlo simulation is carried out for the H_2/CO_2 flame with jet velocity $u_j = 130$ m/s down to $x/D_j = 100$, and using 50,000 stochastic particles. This number of particles is sufficient to ensure satisfactory statistical representation of the flow field [23]. The chemistry is represented over three time scales, 0.4, 12.5, and 200 μs. The CPU time for the simulation using the ANN approach is 557 min, compared with 617 min required by the LUT method. These results reinforce our earlier observation showing the ANN method is becoming computationally cheaper than the look-up table method.

Neural Model Versus Look-up Table

In this section, the concentration of major species and the flame temperature for the flame with $u_j = 80$ m/s, which are obtained using the ANN approach, are compared with those acquired using the LUT method. Unfortunately, no experimental data are available for this velocity. The mean peak temperature and mass fractions of H_2O and CO at different streamwise locations $x/D_j = 2$, 6, 10, 20, 30, 39, and 50, are shown in Tables 5 and 6. These averages are obtained from the instantaneous values (scatter plots). The radial locations in which the concentrations and the flame temperature reach their peak values are similarly predicted by both methods. In general, both

TABLE 5

Comparison of Mean Peak Values of Temperature and CO and H_2O Mass Fractions Between the Look-up Table (LUT) and Artificial Neural Network (ANN) Model for the Flame with Jet Velocity u_j = 50 m/s

x/D_j	2	6	10	20	30	39	50
T (K)							
LUT	1612	1603	1616	1633	1649	1674	1699
ANN	1610	1617	1600	1608	1601	1617	1622
Y_{H_2O}(%)							
LUT	16.9	15.9	15.6	16.7	16.3	17.0	16.9
ANN	16.2	16.0	15.2	16.1	16.6	15.6	16.4
Y_{CO}(%)							
LUT	2.8	3.1	2.9	3.5	3.6	3.6	3.6
ANN	3.5	3.2	2.6	3.3	3.0	3.1	3.8

methods produce comparable results and show consistency in the distribution of the concentrations and the flame temperatures.

Comparison with Experimental Data

Direct comparison between the calculated instantaneous values of the flame temperature and the mass fractions of major species, and the experimental data for the flames with jet velocity u_j = 130 m/s at x/D_j = 6 and 39, are shown in Figs. 13 and 14, respectively. The overall trend in the scatter is similar to the experimental ones with the latter showing more scatter which is partly due to experimental error. The mean peak values of the flame temperature ($\overline{T}$), and the mass fractions $\overline{Y}_{CO}$ and $\overline{Y}_{H_2O}$, are in good agreement with experiment and show that the peak flame temperature decreases to a low value ~ 1400 K at x/D_j = 6. The results at x/D_j = 6 show also that a large number of particles which have temperatures as low as 1000 K, exist within the reactive space ($0.1 < \xi < 0.6$). Such low-temperature fluid particles do not necessarily imply here that local extinction has occurred as these may result from pure mixing between particles from the lean and rich side of stoichiometry. However, the probability of such occurrences, measured in terms of reactedness, is less than 0.5% (~ 2% based on the experimental data).

The predictions of the temperature and the mass fractions of CO and H_2O, are in very good agreement with the experimental results at x/D_j = 6. At x/D_j = 39 the flame is fully reacted with almost all particles near stoichiometric ξ_s having temperatures greater than 1400 K, but the prediction is less satisfactory. Although the scatter plots of the temperature

TABLE 6

Comparison of Mean Peak Values of Temperature and CO andH_2O Mass Fractions Between the Look-up Table (LUT) and Artificial Neural Network (ANN) Model for the Flame with Jet Velocity u_j = 80 m/s

x/D_j	2	6	10	20	30	39	50
T (K)							
LUT	1658	1570	1567	1592	1626	1633	1680
ANN	1603	1470	1502	1587	1612	1617	1619
Y_{H_2O}(%)							
LUT	16.7	14.9	15.0	15.6	16.8	16.8	16.8
ANN	16.1	13.6	14.5	15.3	16.3	16.3	16.3
Y_{CO}(%)							
LUT	3.2	2.8	2.7	3.2	3.7	3.5	3.6
ANN	3.2	2.0	2.5	2.8	4.0	3.5	3.4

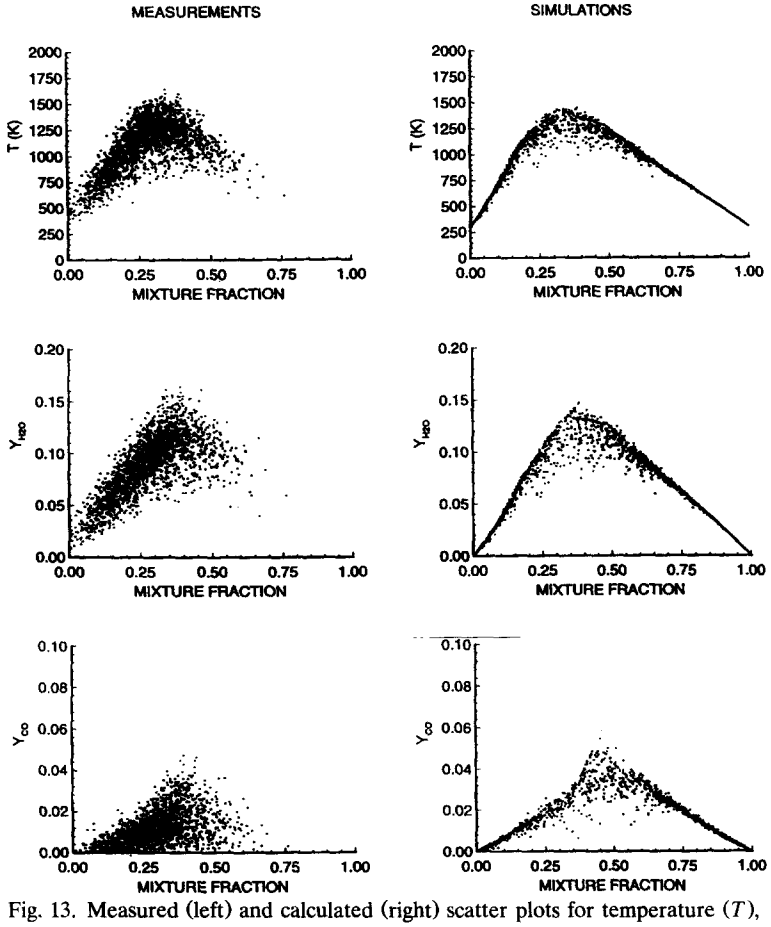

Fig. 13. Measured (left) and calculated (right) scatter plots for temperature (T), and the mass fractions of CO and H_2O plotted against mixture fraction ξ, for the H_2/CO_2 flame with u_j = 130 m/s, at x/D_j = 6.

and the mass fractions are similar to the measured distributions, and their peak values occur at similar mixture fractions, the actual values show significant differences. The maximum flame temperature at $x/D_j = 39$ is $\sim$ 300 K lower than the measured temperature, and the mass fractions of H_2O and CO are underpredicted by $\sim$ 23% and $\sim$ 39%, respectively. Similar results have also been obtained when look-up tables are used to represent the chemistry. Taing et al. [23] have used look-up table method in their calculations and reported similar under-prediction of the mass fractions of H_2O and CO. The problem of substantially underpredicting CO levels has also been reported for different types of fuels, and different numerical schemes [45–47]. Hence, it is reasonable to assume that the discrepancies between the predictions and the experimental results are not a consequence of adopting the neural network modeling approach.

CONCLUSIONS

The feasibility of using the artificial neural network approach to represent complex, highly nonlinear chemical reactions is demonstrated successfully. This approach is implemented in the pdf/Monte Carlo simulation of turbulent jet diffusion flames of H_2/CO_2 fuel mixture. The simulations show good predictions of the flow field and the flame characteristics in comparison to other approaches, and also to the experimental data.

Neural networks with multilayer perceptron architecture, are found to be largely insensitive to the number of neurons used in a two-hidden-layers configuration with a balanced

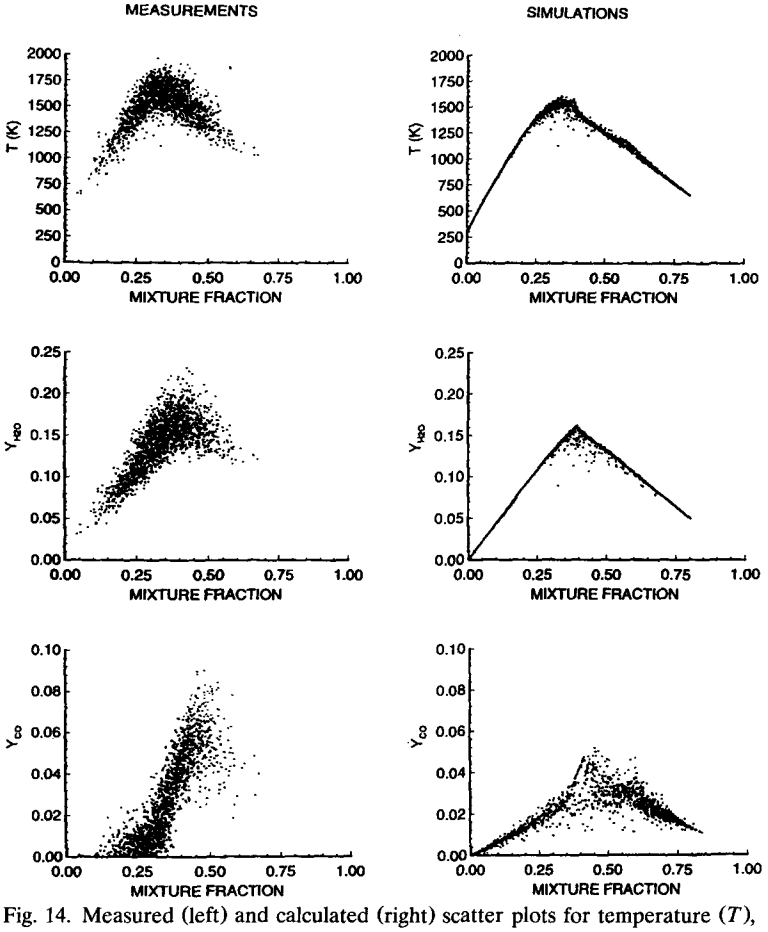

Fig. 14. Measured (left) and calculated (right) scatter plots for temperature (T), and the mass fractions of CO and H_2O plotted against mixture fraction ξ, for the H_2/CO_2 flame with $u_j = 130$ m/s, at $x/D_j = 39$.

number of neurons. The optimization of the network's architecture and performance rely heavily on the selection of the training samples. An algorithm using dynamic randomization of the training set has been developed to reduce oscillations and to prevent local minimum traps during the learning stage.

The superiority of the neural network approach becomes clearer for chemical kinetic mechanisms containing more than four reactions. Both the CPU time and memory storage requirements become more favorable than those of the look-up table and the direct integration approaches.

This work has been supported by the Australian Research Council. Farid C. Christo acknowledges also the generous financial support of the Department of Mechanical and Mechatronic Engineering, Sydney University, New South Wales, Australia.

REFERENCES

1. Kuo, K., *Principles of Combustion*, Wiley, New York, 1986.
2. Chieger, N., *Energy, Combustion and Environment*, McGraw-Hill, New York, 1981.
3. Fletcher, C. A. J., *Computational Techniques for Fluid Dynamics*, 2nd ed., Springer-Verlag, 1990, vol. I.
4. Feiereisen, W. J., Shirani, E., Frezier, J. H., and Reynolds, W. C., in *Turbulent Shear Flows* (L. J. S. Bradbury, F. Durst, B. E. Launder, F. W. Schmidt, and J. H. Whitelaw, Eds.), Springer-Verlag, 1982, Vol. 3, pp. 309–319.
5. Williams, F. A., in *Turbulent Mixing in Non-Reactive and Reactive Flows*, Plenum, New York, 1975, pp. 189–208.

6. Liew, S. K., Bray, N. C., and Moss, J. B., *Combust. Sci. Tech.*, 27:69–73 (1981).
7. Peters, N., *Combust. Sci. Technol.* 30:1–17 (1983).
8. Rogg, B., Behrendt, F., and Warnatz, J., *Twenty-First Symposium (International) on Combustion*, The Combustion Institute, Pittsburgh, 1986, pp. 1533–1541.
9. Bilger, R. W., *Phys. Fluids* 5:436–444 (1993).
10. Klimenko, A. Yu., *Fluid Dynamics* 25:327–335 (1990), translated from *Mekh. Zhidkoshi Gaza* 3:3–10 (1990).
11. Smith, N. S. A., Bilger, R. W., and Chen, J.-Y., *Twenty-Fourth Symposium (International) on Combustion*, The Combustion Institute, Pittsburgh, 1992, pp. 263–269.
12. Klimenko, A. Yu., Private communications (1995).
13. Pope, S. B., in *Turbulent Shear Flows* (L. J. S. Bradbury, F. Durst, B. E. Launder, F. W. Schmidt, and J. H. Whitelaw, Eds.), Springer-Verlag, 1980, Vol. 2, pp. 7–16.
14. Pope, S. B., *Combust. Sci. Technol.* 25:159–174 (1981).
15. Nguyen, T. V., and Pope, S. B., *Combust. Sci. Technol.* 42:13–45 (1984).
16. Libby, P. A., and Williams, F. A., in *Turbulent Reacting Flows* (P. A. Libby and F. A. Williams, Eds.), Springer-Verlag, 1980, pp. 1–43.
17. Sreenivasan, K. R., Tavoularid, S., and Corrsin, S., in *Turbulent Shear Flows* (L. J. S. Bradbury, F. Durst, B. E. Launder, F. W. Schmidt, and J. H. Whitelaw, Eds.), Springer-Verlag, 1982, Vol. 3, pp. 96–112.
18. Bray, K. N. C., in *Turbulent Reacting Flows* (P. A. Libby and F. A. Williams, Eds.), Springer-Verlag, 1980, pp. 115–180.
19. Peters, N., in *Reduced Kinetic Mechanisms and Asymptotic Approximations for Methane–Air Flames* (D. Mitchell Smooke, Ed.), Springer-Verlag, 384, 1990, p. 48.
20. Bilger, R. W., Esler, M. B., and Stårner, S. H., in *Reduced Kinetic Mechanisms and Asymptotic Approximations for Methane–Air Flames* (D. Mitchell Smooke, Ed.), Springer-Verlag, 384, 1990, p. 86.
21. Maas, U., and Pope, S. B., *Combust. Flame* 88:239–264 (1992).
22. Maas, U., and Pope, S. B., *Twenty-Fourth Symposium (International) on Combustion*, The Combustion Institute, Pittsburgh, 1992, pp. 103–112.
23. Taing, S., Masri, A. R., and Pope, S. B., *Combust. Flame* 95:133–150 (1993).
24. Turànyi, T., *Twenty-Fifth Symposium (International) on Combustion*, The Combustion Institute, Pittsburgh, 1994, pp. 948–955.
25. Turànyi, T., *Computers Chem.* 18(1):45–54 (1994).
26. Beale, R., and Jackson, T., *Neural Computing: An Introduction*, IOP Publishing, 1990.
27. Norris, A. T., PhD thesis, Cornell University, 1993.
28. Lippmann, R. P., IEEE ASSP Magazine, 1987.
29. Cybenko, G., *Mathematics of Control Signal and Systems*, 1989, pp. 303–314.
30. Funahashi, K., *Mathematics of Control Signal and Systems*, 1989, pp. 183–192.
31. Rumelhart, D. E., McClelland, J. L., *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, Vol. 1. Foundations, Cambridge, MIT Press, 1986.
32. Wu, Jian-Kang, *Neural Networks and Simulation Methods*, Marcel Dekker, New York, 1994.
33. de Villiers, J., and Barnard, E., *IEEE Trans. Neural Networks* 4 (1) (Jan. 1992).
34. Pope, S. B., *Prog. Ener. Combust. Sci.* 11:119–192 (1985).
35. Pope, S. B., *Combust. Sci. Technol.* 28:131–135 (1982).
36. Chen, J.-Y., and Kollmann, W., in *Turbulent Shear Flows* (L. J. S. Bradbury, F. Durst, B. E. Launder, F. W. Schmidt, and J. H. Whitelaw, Eds.), Springer-Verlag, 1991, Vol. 7, pp. 278–292.
37. Norris, A. T., and Pope, S. B., *Combust. Flame* 83:27–42 (1991).
38. Pope, S. B., *Theort. Comput. Fluid Dynamics* 2:255–270 (1991).
39. Masri, A. R., and Pope, S. B., *Combust. Flame* 81:13–29 (1990).
40. Pope, S. B., *AIAA J.* 22:896–904 (1994).
41. Pope, S. B., in *Turbulent Shear Flows* (L. J. S. Bradbury, F. Durst, B. E. Launder, F. W. Schmidt, and J. H. Whitelaw, Eds.), Springer-Verlag, 1982, Vol. 3, pp. 113–123.
42. Stårner, S. H., and Bilger, R. W., *Combust. Flame* 61:29–38 (1985).
43. Rogg, B., and Williams, F. A., *Twenty-Second Symposium (International) on Combustion*, The Combustion Institute, Pittsburgh, 1988, pp. 1441–1451.
44. Masri, A. R., Dibble, R. W., and Barlow, R. S., *Combust. Flame* 91:285–309 (1992).
45. Jones, W. P., and Kollmann, W., in *Turbulent Shear Flows* (L. J. S. Bradbury, F. Durst, B. E. Launder, F. W. Schmidt, and J. H. Whitelaw, Eds.), Springer-Verlag, Vol. 5, 1987, pp. 297–309.
46. Sion, M., and Chen, J.-Y., *Combust. Sci. Technol.* 88:89–114 (1992).
47. Correa, M. Sanjay, Gulati, G., and Pope, S. B., *Twenty-Fifth Symposium (International) on Combustion*, The Combustion Institute, Pittsburgh, 1994, pp. 1167–1173.

Received 26 June 1995; revised 7 November 1995