Computational Economy Improvements in PRISM

Shaheen R. Tonse[1] and Nancy J. Brown
Environmental Energy Technologies Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA

Abstract

The PRISM piecewise solution mapping procedure, in which the solution of the chemical kinetic ODE system is parameterized with quadratic polynomials, is applied to CFD simulations of H_2+air combustion. Initial cost of polynomial construction is expensive, but it is recouped as the polynomial is reused. We present two methods that help us to parameterize only in places that will ultimately have high reuse. We also implement non-orthogonal Gosset factorial designs, that reduce polynomial construction costs by a factor of two over previously used orthogonal factorial designs.

Notation

N_s:	Number of chemical species
Δt:	Length of timestep
$\mathcal{C}$:	Chemical composition space
V_Tr:	Trajectory velocity
N_ER:	Number of Expected Reuses of a hypercube
N_AR:	Number of Actual Reuses of a hypercube
C_ODE:	Cost of an ODE time-integration
C_PC:	Cost of polynomial construction
C_PE:	Cost of a polynomial evaluation
N_D:	Reuse threshold for DPC
N_B:	Break-even usage of a hypercube
$N_<$:	Number of hypercubes with $N < N_\mathrm{B}$
$N_>$:	Number of hypercubes with $N > N_\mathrm{B}$
$\overline{N_<}$:	Mean usage of hypercubes with $N < N_\mathrm{B}$

Introduction

Reactive computational fluid dynamics (CFD) calculations with large modeling domains and chemical reaction sets are still time consuming on today's computers. A substantial fraction of the computing resources are used for integrating the chemical kinetic ordinary differential equation (ODE) system to calculate the changes in concentration of the chemical species and enthalpy. One class of methods used to reduce the costs of complex chemistry develops models to mimic the time evolution of the rate equations of chemical kinetics. In these approaches the time-integration calculation, which dictates the evolution of the chemical kinetics over time is viewed as a mapping from one chemical composition (and temperature) to another. This time integration is parameterized by an inexpensive approximate mapping with a simpler functional form. Approaches within this class include: Piecewise Reusable Implementation of Solution Mapping (PRISM) [1]; Fifth- to eighth-order polynomial parameterizations [2]; *In situ* adaptive tabulation (ISAT) [3,4]; and Laminar flamelet libraries [5]. As examples: in PRISM the parameterization is a set of quadratic polynomials defined within a hypercube; in ISAT it is a set of linear polynomials defined within an ellipsoidal region. In PRISM the parameterization is done piecewise on hypercubes of dimension N_s+1 which partition $\mathcal{C}$ and contain a distinct parameterization for each hypercube. Subsequently, if the time-evolution of a mixture is required, it is simple a matter of evaluating the polynomial.

In a previous study [1] we simulated laminar premixed and turbulent non-premixed H_2+air combustion and saw a factor of 10 speedup when comparing the cost of a single polynomial evaluation to a single ODE time-integration. Recent algorithmic improvements have since raised this factor to 15. This gain was offset slightly by the cost of constructing the polynomials for the hypercubes, which imposed an initial investment that yielded returns as the polynomials were reused multiple times. Mean reuse rates of several thousand per hypercube were observed, ample to

[1]Corresponding author: stonse@lbl.gov

recover costs, since cost-effectiveness was achieved at a reuse rate of 266 for the H_2 reaction set. The reuse distribution was a skewed distribution, with a large number of hypercubes having very low reuse (less than 10). The mean usage was increased by a smaller number of highly reused hypercubes. For the low reuse hypercubes it would be far more efficient to use the ODE solver directly rather than construct polynomials.

Specific Objectives

In this paper we investigate two methods to exploit the skewness of the reuse distribution. The first method anticipates high hypercube reuse by utilizing the rates at which trajectories are moving, to calculate a trajectory "velocity" V_{Tr}. This quantity is then combined with an estimated trajectory length through the hypercube to determine the number of expected reuses, N_{ER}, and on this basis we decide whether or not polynomial construction is worth the expense. The second method takes advantage of the shape of the distribution to defer polynomial construction for a hypercube until a certain number of reuses has occurred. These methods are applied to four cases: a 1-D laminar premixed H_2+air flame, a 2-D premixed H_2+air turbulent jet, a 2-D flame with non-premixed H_2 and air turbulent jets, and a laminar premixed CH_4+air flame. We also apply a non-orthogonal Gosset factorial design to the CH_4+air flame

Method

Prism Overview We partition $\mathcal{C}$ into non-overlapping adjacent hypercubes with a distinct polynomial parameterization for each. Calculations for a hypercube are not performed, and storage is not allocated, until a reaction trajectory enters it for the first time. Once calculated, polynomials are stored in a data structure, to be retrieved whenever the time evolution of a composition within that hypercube is required. All calculations and hypercube positions use log(*concentration*) and reciprocal temperature because the underlying chemical rate equations typically conform better to a quadratic model under this transformation.

To parameterize the response of the time-integration, the ODE solver is called at selected points within the hypercube. Each point corresponds to a set of input concentrations, a temperature, and a time-step length. These input concentrations and temperature are propagated by the ODE solver over the specified time-step length, returning a set of final concentrations and temperature (responses) at each point. A quadratic regression is applied to each response, resulting in a set of N_{s}+1 polynomials for the hypercube. By evaluating Eqn. 1 using the initial concentrations, temperature, and time-step as input variables, we obtain as responses the concentrations and temperature at the end of the time-step. The concentration of species i, $C_i^{t+\Delta t}$, is given by:

$$\begin{aligned}\log C_i^{t+\Delta t} \quad = \quad & a_{i,0} + \sum_{j}^{N_{\mathrm{s}}} a_{i,j} \log C_j^t \\ & + a_{i,N_{\mathrm{s}}+1} \frac{1}{T^t} + a_{i,N_{\mathrm{s}}+2} \log \Delta t \\ & + \sum_{j}^{N_{\mathrm{s}}} \sum_{k \le j}^{N_{\mathrm{s}}} a_{i,jk} \log C_j^t \log C_k^t \\ & + a_{i,N_{\mathrm{s}}+1\ \ N_{\mathrm{s}}+1} \frac{1}{T^t} \cdot \frac{1}{T^t} \\ & + a_{i,N_{\mathrm{s}}+2\ \ N_{\mathrm{s}}+2} \log \Delta t \cdot \log \Delta t \\ & + \text{cross terms of } C_j, T, \Delta t \qquad (1)\end{aligned}$$

where a_i are the polynomial coefficients determined in this procedure. We ensure accuracy by requiring that each species within a hypercube has a relative error (of the regression residuals) in molar concentration of less than 5×10^{-3} and an absolute error of less than 10^{-4} times the sum of all the molar concentrations.

Computational costs for polynomial construction, polynomial evaluation and ODE integration are specific to the H_2+air chemical mechanism and the order of factorial design utilized, but not strongly dependent on the simulation's physical conditions, e.g., they do not vary much between a 0-D simulation and a 2-D turbulent jet. The factorial design dictates the number of ODE calls to be made during polynomial construction. The costs of ODE time-integration and polynomial evaluation both depend on the chemical mechanism. The relative computational costs, specific to the H_2 reaction set, for polynomial construction, polynomial evaluation and ODE integration are in the ratio $266 : \frac{1}{15} : 1$.

Trajectory Velocity (V_{Tr}) This method aims to identify hypercubes that have high reuse by predicting the number of time-steps (defined as Number of Expected Reuses: N_{ER}) taken when the chemical trajectory enters a new hypercube, and at that time deciding whether or not to construct the hypercube's polynomials.

We define trajectory velocity V_{Tr} as a vector in $\mathcal{C}$, with each component $(V_{\mathrm{Tr}})_i$ being the net rate of change of the ith species, calculated from the sum of the rates ($\dot{\omega}_{ij}$) of the reactions that contribute to its production: $(V_{\mathrm{Tr}})_i = \sum_{j=1}^{26} \dot{\omega}_{ij}$, where the total number of reactions in this particular mechanism is 26.

Multiplying V_{Tr} by the time-step, $V_{\mathrm{Tr}} \cdot \Delta t$ gives a first order approximation to the trajectory displacement vector in $\mathcal{C}$.

In our case studies, the influence of CFD on trajectory movement through $\mathcal{C}$ is significant, so we augment $\mid V_{\mathrm{Tr}} \mid \cdot \Delta t$ with the combined displacements of the CFD and the chemistry contributions, done by measuring the displacement of a trajectory from one time-step to the next. Since the hypercube is often accessed by more than one CFD grid cell, it is necessary to associate the CFD cell with the trajectory, by using the CFD grid indices. When a trajectory enters a hypercube the chemical mixture and cell index are stored. On the next time-step if the trajectory from the same CFD cell remains in the same hypercube, then the displacement in $\mathcal{C}$ is calculated. Using the same linear extrapolation idea as for the purely chemical case above we then calculate N_{ER}. This modification shows improved accuracy over the purely chemical version,

In addition to the trajectory length of the time-step $\mid V_{\mathrm{Tr}} \mid \cdot \Delta t$, we require the expected trajectory length through the hypercube. To estimate transit length, we linearly extrapolate the trajectory entry point coordinates in the direction vector of V_{Tr} until it exits the hypercube and define

$$N_{\mathrm{ER}} \equiv \frac{\textit{trajectory length}}{\mid V_{\mathrm{Tr}} \mid \cdot \Delta t} \tag{2}$$

Whenever the trajectory enters a new hypercube we calculate N_{ER}. If N_{ER} is above break-even cost usage, (≈ 250) we proceed to construct polynomials for the hypercube and store them in a data structure. If N_{ER} does not meet the usage requirement we do not construct polynomials as we would not recover the polynomial construction costs. The hypercube is flagged as "ODE only"and subsequent visits from a chemical trajectory to this hypercube result in an ODE time integration.

There remain hypercubes where actual reuse N_{AR} far exceeds N_{ER}. The majority of hypercubes with this behavior are visited by trajectories from large numbers of CFD cells (> 20). For these cases the first trajectory into the hypercube, (which is the one used to calculate N_{ER}) is likely not the one that uses the hypercube the most. A later trajectory hits the hypercube at a location where it has a much smaller trajectory velocity and *this* is the trajectory that carries the most weight in determining hypercube usage. This later trajectory would give a better estimate of N_{ER}. To take advantage of this we modified our procedure so that if a hypercube fails the N_{ER} cut, the ODE solver is called, but N_{ER} calculations continue to be performed for later trajectories entering the same hypercube. If one of them passes the cut, then polynomials are constructed.

Deferred Polynomial Construction (DPC) Our objective here is to study the consequences on computational expense of deferring the construction of polynomials for a hypercube until it has first been reused a certain number of times, N_{D}. In the interim the ODE solver is called to advance the trajectory through the hypercube. The criteria to determine N_{D} are cost-based; waiting until N_{D} reuses have occurred eliminates polynomial construction for hypercubes with reuse less than N_{D}, but simultaneously increases the cost associated with the remaining hypercubes by subjecting them to unnecessary ODE integrations.

Consider a hypercube reused N times. If N is less than N_{D} the total cost is that of calling the ODE solver N times: $C_{\mathrm{ODE}} \cdot N$. If N is greater than N_{D} the cost has contributions from calls to the ODE solver (C_{ODE}), then polynomial construction (C_{PC}), and subsequent polynomial evaluation (C_{PE}): $C_{\mathrm{ODE}} \cdot N_{\mathrm{D}} + C_{\mathrm{PC}} + C_{\mathrm{PE}} \cdot (N - N_{\mathrm{D}})$. For a set of hypercubes whose reuse distribution is denoted $f(N)$ the total cost is obtained by convolving the cost with $f(N)$:

$$\begin{aligned} \textit{Total cost} \;=\; & \int_0^{N_{\mathrm{D}}} C_{\mathrm{ODE}} \cdot f(N) N dN \\ & + \int_{N_{\mathrm{D}}}^{N_{\max}} [C_{\mathrm{ODE}} \cdot N_{\mathrm{D}} + C_{\mathrm{PC}} \\ & + C_{\mathrm{PE}} \cdot (N - N_{\mathrm{D}})] \cdot f(N) dN \end{aligned} \tag{3}$$

Here $N_{\max}$ is the upper limit of $f(N)$ and we take the liberty of replacing the summation by an integral in light of large values of N. Note that setting $N_{\mathrm{D}} = 0$ reduces Eqn.3 to the special case where polynomial construction is not deferred.

To gain some insight into the problem we chose several simple functions for $f(N)$, and analytically evaluated and differentiated Eqn.3 with respect to N_{D} to find the value of N_{D} which resulted in minimum cost. This gave us insight into the shapes of reuse distributions that would benefit from a DPC cut. It also told us that the value of N_{D} is close to N_{B}. We also noted that the minima were shallow, which implied that searching for the most optimal N_{D} would not net us much more gain than simply setting N_{D} to a reasonably close value, such as N_{B}. Subsequently we derived a simple formula which shows the computational gain that results from moving from N_{D}=0 (no DPC) to N_{D}=N_{B}, for any simple integrable $f(N)$:

$$\text{Computational Gain} = C_{\mathrm{PC}}[N_{<}(1 - \frac{\overline{N_{<}}}{N_{\mathrm{B}}}) - N_{>}] \tag{4}$$

Using this, one can determine whether it is worthwhile setting the deferred polynomial construction threshold to N_B given *any* hypercube reuse distribution $f(N)$.

Results and Discussion

The V_{Tr} and DPC methods are tested on 4 different time-evolving reactive flow simulations. The Coyote CFD code [6] is used, within which the chemistry calculation uses the DVODE differential equation solver [7] and the CHEMKIN thermodynamic library [8]. Three of the simulations use a 9-species H_2+air reaction set, obtained by removing carbon from the GRI-Mech 2.11 mechanism [9]. The fourth, CH_4+air simulation uses DRM19, a reduced version of GRI-Mech [9]. For each case we show how the methods result in fewer polynomial being constructed, and the resulting computational gain will be shown on plots of CPU time vs. timestep number. Table summarizes the performance with information on number of hypercubes constructed, their mean reuse, CPU time used in chemistry, and total CPU time, for the H_2 flame examples.

The 1-Dimensional Laminar Flame example includes the influence of convection and diffusion between CFD grid cells: a propagating premixed 1-D laminar flame. The physical configuration is a 1 cm long (200 CFD grid cells) tube, closed at one end and open at the other. A small portion of the tube near the open end is filled with hot burned gas, while the remainder is filled with unburned stoichiometric H_2-air at room temperature. A flame forms at the interface between the burned and unburned gas mixtures and propagates toward the closed end. During the simulation N_{ER} is calculated when a hypercube is first used, and at the end the reuse statistics for all hypercubes are gathered, giving us their respective N_{AR}. Figure 1 shows the effect of including or excluding the effect of fluid mechanics on trajectory movement in N_{ER} calculation. The correlation is improved if N_{ER} is calculated taking into account the effect of fluid mechanics on trajectory movement. The economy of the method is illustrated by Fig. 2a which shows accumulated CPU time in seconds vs. timestep number for different cases. The expense is highest when using an ODE solver instead of PRISM to advance the chemistry. The CPU time per timestep is reflected by the slope of this curve, which increases slightly as the simulation progresses, due to the decreasing number of CFD cells containing cool, unburnt gas as the flame front progresses down the tube. For PRISM with no N_{ER} or DPC requirement, (labeled "no cut") there is a sharp initial rise caused by the expense of polynomial construction for approximately 3000 hypercubes early in the run; subsequent CPU time is used mainly for polynomial evaluation. This is because for a premixed 1-D laminar flame with approximate translational symmetry nearly all of active $\mathcal{C}$ is accessed early. At later times as the flame propagates through the mixture it has translational symmetry and so covers about the same portion of $\mathcal{C}$ as earlier. The simulation was run for 90000 time-steps with almost all hypercube construction in the first 5000 timesteps. The slope at later times is less than that of the ODE curve, as polynomial evaluation is less expensive than ODE time-integration. The case requiring hypercubes to have $N_{ER} > 250$ before allowing polynomial construction results in polynomials being constructed for far fewer hypercubes so that the initial rise is smaller. The total CPU time for this is only about half of that for the "no cut" case. Good performance is also seen from the DPC method. N_B depends mainly on the chemical reaction set, and is 266. Figure 2a shows the accumulated CPU time. As with the $N_{ER} > 250$ case, far fewer polynomials are constructed, resulting in a

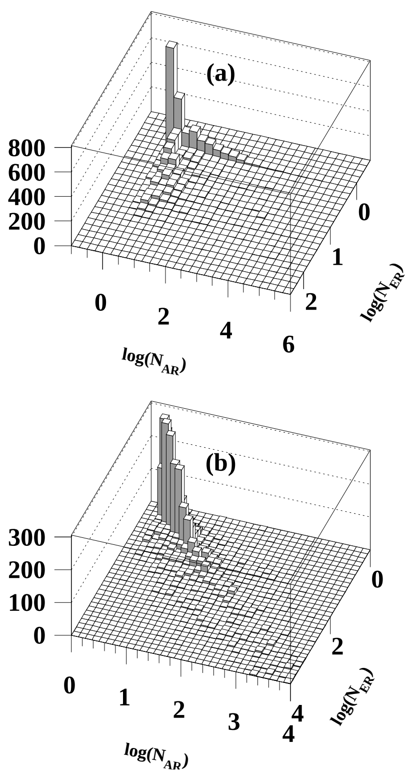

Figure 1: (a) A 2-D frequency distribution of N_{AR} vs. N_{ER} of every hypercube from the laminar flame case. N_{ER} is calculated solely using chemical information in the same way as for the 0-D case.
(b) A 2-D frequency distribution of N_{AR} vs. N_{ER} of every hypercube from the laminar flame case. N_{ER} is calculated using both chemical and CFD information.

	ODE	PRISM	V_{Tr}	DPC
Laminar				
# hcubes	–	5484	349	382
$< reuse >$	–	1238	19004	17601
CPU(chem)	2300	850	400	350
CPU(tot)	3100	1700	1250	1200
Premixed turbulent				
# hcubes	–	23120	3211	3438
$< reuse >$	–	1771	12411	11701
CPU(chem)	21800	5700	3700	3700
CPU(tot)	27800	11600	9500	9500
Non-premixed turbulent				
# hcubes	–	115383	5617	9751
$< reuse >$	–	356	6738	4206
CPU(chem)	50200	41700	20500	18400
CPU(tot)	56500	47700	26400	24600

Table 1: Summary performance comparison for the laminar, premixed turbulent, and non-premixed turbulent cases, each of which was run in 4 modes: (i) ODE-only, (ii) PRISM (without V_{Tr} or DPC), (iii) V_{Tr} (trajectory velocity), (iv) DPC (deferred polynomial construction). Shown are the number of hypercubes for which polynomial construction occurred, the mean reuse of those hypercubes, the accumulated CPU time (chemistry only), and the accumulated CPU time (CFD + chemistry).

smaller initial rise. At later times the slopes of the N_{ER}, DPC and "no cut" cases are nearly the same, indicating that we are correctly rejecting the hypercubes that should be rejected and retaining those that should not.

The 2-D Axisymmetric Premixed Turbulent Jet example is a simulation of a premixed H_2+air 2-D turbulent jet, starting from a quiescent non-combusting state and proceeding until a turbulent flame has developed. The physical configuration is a cylindrically symmetric chamber of radius 8 cm and height 20 cm, with the inlet at the center of the base and open at the top to atmospheric pressure. The inlet conditions are stoichiometric H_2+air at 21 m/s and 300 K from a jet of radius 0.35 cm. The chamber is initially filled with air at 300 K with the exception of a "hot-spot" of air at 1600 K placed near the fuel jet, to initiate combustion. The simulation was run for about twice the time needed to reach a steady state. Figure 2b shows the accumulated CPU time curves for the same 4 cases as were shown in the laminar flame. At timestep 30000 (about the time steady state was reached) the ratio of total CPU times between the DPC and "no cut" curves was 0.6. At timestep 60000 the ratio was 0.7.

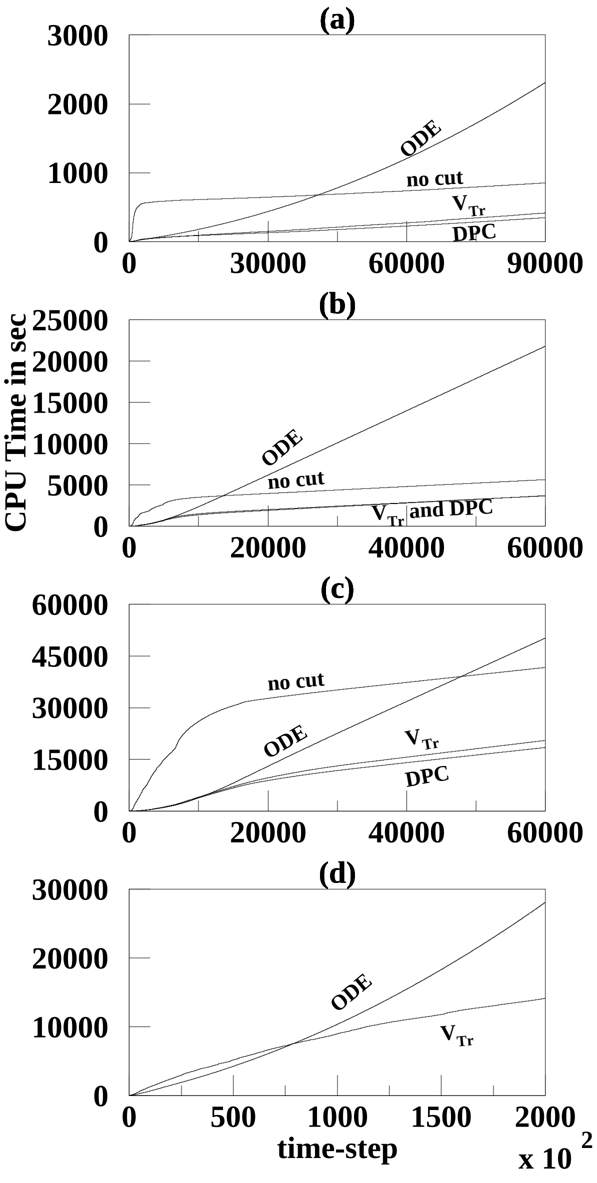

Figure 2: The cumulative CPU time used for the chemistry portion of the simulation vs. CFD timestep number, for an ODE case, a normal PRISM case ("no cut,)", PRISM with V_{Tr} ($N_{\mathrm{ER}} > 250$), and PRISM with DPC ($N_{\mathrm{D}} = 266$) for: (a) 1-D H_2 laminar flame (b) premixed turbulent jet (the V_{Tr} and DPC curves overlap) (c) non-premixed turbulent jet. (d) 1-D laminar CH_4 flame with a Gosset design and $N_{\mathrm{ER}} > 304$. Conducted on a 1.8 GHz Athlon Linux workstation. (RAM is not an issue.) Only the portion of CPU time utilized in chemistry calculations is reported.

The 2-D Axisymmetric Non-premixed Turbulent Jet example is a non-premixed 2-dimensional turbulent jet with coaxial H_2 and air inflows, starting from a quiescent non-combusting state and proceeding until a turbulent flame has developed. The physical configuration is similar to that above, except that there are two concentric inlets at the center of the base, with H_2 at 21 m/s and 300 K in the inner jet of radius 0.35 cm, and air at 1 m/s and 300 K in the outer jet, which has radial extent from 0.5 to 8 cm. We run for about twice the time needed to reach a steady state. Figure 2c shows the CPU usage for the "no cut," $N_{\mathrm{ER}} > 250$ and DPC cases. The $N_{\mathrm{ER}} > 250$ and DPC perform about equally well, with the former slightly better. The non-premixed case differs from the previous cases in that it accesses a substantially larger portion of $\mathcal{C}$.

The 1-Dimensional CH_4+air Laminar Flame example is identical physically to the hydrogen case. but uses a stoichiometric CH_4+air mixture. To extend PRISM to CH_4+Air combustion, we had developed a 22-dimension factorial design, 2_V^{22-13}. This design was used in preliminary PRISM calculations and it gave accurate results; however, the number of ODE calls required for polynomial construction was 554 therefore correspondingly hypercube reuse was required to recoup construction costs. We have found that a factorial design produced by the Gosset program (http://www.research.att.com/ njas/gosset/) provides better accuracy than an orthogonal fractional design, and requires only about 300 ODE calls. Gosset designs are not orthogonal, necessitating a full matrix inversion at the polynomial construction stage, but this cost is easily offset by the reduced number of ODE calls. The polynomial construction cost is reduced almost a factor of 2 over that of the orthogonal design. Fig. 2d compares CPU times for an ODE run and for a case which uses the Gosset design and V_{Tr}. The DPC method CPU performance was identical to that of V_{Tr}, and so is not shown.

Conclusion

The Trajectory Velocity (V_{Tr}) method, which employs the rate of movement of a chemical trajectory combined with an estimated path length through a hypercube to estimate the expected hypercube reuse, was applied to 4 cases: a propagating laminar flame, a turbulent premixed flame, and a turbulent diffusion flame. In the three flames considered, the efficiency was improved by a factor of 1.5 to 2.5.

In the Deferred Polynomial Construction (DPC) method efficiency is improved by deferring polynomial construction until a hypercube has been reused a specific number of times. In the three cases considered, the efficiency was improved by a factor of 1.5 to 2.5.

Preliminary results for the application of PRISM to CH_4 flames is encouraging.

Acknowledgements

The work was supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Chemical Sciences Division of the U.S. Department of Energy, all under contract No. DE-AC03-76SF00098.

LBNL internal report LBNL-52401

References

[1] Tonse, S. R., Moriarty, N. W., Brown, N. J., and Frenklach, M., *Israel J. Chem.*, 39:97–106 (1999).

[2] Turanyi, T., *Comp. Chem.*, 18:45–54 (1994).

[3] Pope, S. B., *Combust. Theory Modelling*, 1:41–63 (1997).

[4] Yang, B. and Pope, S. B., *Combust. Flame*, 112:85–112 (1998).

[5] Bray, K. N. C. and Peters, N., (Libby, P. A. and Williams, F. A., eds.), *Turbulent Reacting Flows*, Academic Press, San Diego, CA, USA. 92101-4311, 1994, pp. 63–113.

[6] Cloutman, L. D., "Coyote: A computer program for 2D reactive flow simulation," Technical Report UCRL-ID-103611.

[7] Brown, P. N., Byrne, G. D., and Hindmarsh, A. C., *SIAM J. Sci. Stat. Comput.*, 10:1038–1051 (1989), Also, LLNL Report UCRL-98412, June 1988.

[8] Kee, R. J., Rupley, F. M., Meeks, E., and Miller, J. A., "Chemkin-III: A Fortran chemical kinetics package for the analysis of gas-phase chemical and plasma kinetics," Technical Report SAND96-8216, UC-405.

[9] Frenklach, M., Wang, H., Goldenberg, M., Smith, G. P., Golden, D. M., Bowman, C. T., Hanson, R. K., Gardiner, W. C., and Lissianski, V., "GRI-Mech—an optimized detailed chemical reaction mechanism for methane combustion," Technical Report GRI-95/0058, http://www.me.berkeley.edu/gri_mech/.