LBNL-48858

ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

Computational Economy Improvements in PRISM

Tonse, Shaheen R., Moriarty, Nigel W., Franklach, Michael, Brown, Nancy J.

Environmental Energy Technologies Division

05/2003

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

Computational Economy Improvements in PRISM

Shaheen R. Tonse,[a,*] Nigel W. Moriarty[b] Michael Frenklach,[a,b]

and Nancy J. Brown,[a]

[a] Environmental Energy Technologies Division,
Lawrence Berkeley National Laboratory, Berkeley, CA 94270, USA.

[b] Department of Mechanical Engineering,
University of California at Berkeley, Berkeley, CA 94720–1740, USA.

Keywords: Chemical Kinetics, Response Surface

[*] Author for correspondence.

Telephone: Int + 1(510)486-4556

Fax: Int + 1(510)486-5928

Email: tonse@lbl.gov

May 7, 2003

Abstract

The PRISM piecewise solution mapping procedure is applied to reactive flow simulations of (9–species) H_2+air combustion. PRISM takes the solution of the chemical kinetic ODE system and parameterizes it with quadratic polynomials. To increase the accuracy, the parameterization is done piecewise, by dividing the multi-dimensional chemical composition space into hypercubes and constructing polynomials for each hypercube on demand. The polynomial coefficients are stored for subsequent repeated reuse. Initial cost of polynomial construction is expensive, but it recouped as the hypercube is reused, hence computational gain depends on the degree of hypercube reuse. We present two methods that help us to identify hypercubes that will ultimately have high reuse, this being accomplished before the expense of constructing polynomials has been incurred. One method utilizes the rate of movement of the chemical trajectory to estimate the number of steps the trajectory would make through the hypercube. The other method defers polynomial construction until a preset threshold of reuse has been met; an empirical method which, nevertheless, produces a substantial gain. The methods are tested on a 0-D chemical mixture and reactive flow 1 and 2-D simulations of selected laminar and turbulent H_2+air flames. The computational performance of PRISM is improved by a factor of about 2 for both methods.

1 Introduction

The study of combustion through finite-difference numerical simulations typically involves a marriage of computational fluid dynamics (CFD) and computational chemical kinetics. The role of the CFD includes determination of velocities, pressure, convection across cell boundaries, the equation of state, the modeling of turbulence, and the diffusion of chemical species across cell boundaries. The responsibility of the chemistry portion is to determine the changes in concentration of each chemical species and enthalpy in response to the chemical source terms. This is usually accomplished through the solution of a system of coupled ordinary differential equations (ODE's), one for each of the chemical species and one for the enthalpy. It often proves to be computationally expensive, e.g., combustion calculations with 3-D modeling domains and large chemical reaction sets are not practical on today's computers with the possible exception of supercomputers that have very restricted availability. A substantial fraction of the computing resources are used for integrating the chemical ODE's, e.g., a 2-dimensional calculation with the CFD portion performing all of the above-mentioned tasks, and with the chemistry of a H_2-air mixture, using a reaction set of 9 chemical species and 29 reactions, 85 to 90% of the CPU time is spent on the chemistry [1]. Yet this hydrogen reaction set is a comparatively simple one and if the larger reaction sets with 500-800 chemical species for diesel combustion are used, the fraction of CPU time spent on chemistry is even greater. A necessary solution to the enormous size of kinetic mechanisms for complex fuel oxidation and pollutant formation is to employ a comprehensive reaction mechanism that has been systematically reduced in size from its original, fully detailed form. Identifying and eliminating unimportant reactions and species is the major task of mechanism reduction, and must occur without sacrificing accuracy. The question of how the required level of chemical detail changes with respect to the species and combustion conditions must also be considered. Reduced mechanisms are usually built within a hierarchical spectrum where the complexity of the mechanism is governed by the level of detail required to answer specific questions regarding combustion performance or emission. Quantifying the errors associated with using reduced mechanisms is challenging. Calculations for low-dimensional combustion systems using both the comprehensive and reduced mechanism can be compared for a range of conditions to begin to establish ranges of validity for the reduced mechanism. For isolated cases, results of calculations for more complex combustion systems performed with both comprehensive and reduced mechanisms can be compared using advanced visualization. Some of the more advanced approaches offer the possibility of error control over limited domains. In order to reduce the severity of a brute-force approach to chemical kinetics problems varied approaches are used:

- Reduction of the reaction set, both in the number of chemical species and the number of reactions, in a systematic way after examining sensitivities and reaction fluxes; [2–6]
- Steady-state and partial equilibrium approximations; [7, 8] Quasi-steady state assumptions (QSSA) and partial equilibrium (PE) methods have been used widely to reduce mechanisms. Using the QSSA requires identifying those species that react on a very short time scale relative to the other species. The QSSA is then used to derive explicit algebraic expressions for concentrations of the QSS species that can be used to reduce the number of differential equations used to describe the evolution of the chemical species in time. Partial equilibrium approximations can also used to provide algebraic expressions for species concentrations. The difficulty with these approximations is identifying species that are in quasi-steady state and reactions that are in partial equilibrium as combustion conditions change. Reductions using QSSA and PE approximations are usually carried out manually on smaller chemical models referred to as skeleton models.

- The Intrinsic Low-dimensional Manifold Method [9–11] is based on a high-dimensional initial-state reaction trajectory in combustion quickly converging to a low-dimensional manifold of dimension n_c, where $n_c \ll n_s$, the total number of species. The approach takes advantage of the existence of many different reaction time scales for different species during the combustion process.

- Computational Singular Perturbation [12] is also a reduction technique based upon classifying reactions according to their time scales. During the various stages of combustion, fast reaction groups can be excluded when they are completed. At some time only a small set of important species remains.

- Using mathematical analysis methods to upgrade the perfomance of existing chemical kinetic numerical solvers, through implemention of DAEPACK [13]. The performance is especially impressive for large, sparse chemical mechanisms of several hundred species.

- Optimization Approach [14] A global reduction of the original system of chemical rate equations, replacing it with a reduced set which inherits the stability and the non-linear behavior of the original set of equations.

- Principal Component Analysis [15], an eigenvalue-eigenvector analysis of the sensitivity matrix of the normalized sensitivity coefficients and its transpose has also been used for mechanism reduction. The eigenvalues provide an absolute measure of the significance of some parts of the mechanism. The magnitude of the coefficients in the eigenvectors measure the importance of the reactions for a given eigenvalue. Taken together, the eigenvalues and eigenvectors measure the significance of reactions in the overall mechanism. Principal component analysis provides an objective criterion for selecting a minimum reaction set by including only those reactions which comprise the principal components. A bare-bones mechanism can be obtained that might be used as the starting point of other mechanism reduction scheme. These methods reduce the severity of the computational problem, but still require the solution of differential equations.

Another class of methods used to reduce the costs of complex chemistry develops models to mimic the time evolution of the ODE's. In these approaches the time-integration calculation, which dictates the evolution of the chemical kinetics over time interval Δt, is viewed as a mapping from one chemical composition (and temperature) to another:

$$\mathcal{F}(C_1^t, C_2^t, \ldots, T^t, \Delta t) \rightarrow (C_1^{t+\Delta t}, C_2^{t+\Delta t}, \ldots, T^{t+\Delta t}) \tag{1}$$

where C_i^t is the concentration of species i at time t. An inexpensive approximate mapping Φ with a simpler functional form is then constructed, replacing $\mathcal{F}$ with:

$$\Phi(C_1^t, C_2^t, \ldots, T^t, \Delta t) \rightarrow (C_1^{t+\Delta t}, C_2^{t+\Delta t}, \ldots, T^{t+\Delta t}) \tag{2}$$

Approaches within this class include: Solution Mapping [16,17]; Piecewise Reusable Implementation of Solution Mapping (PRISM) [18]; Fifth- to eighth-order polynomial parameterizations [19]; and *In situ* adaptive tabulation (ISAT) [20,21]. Laminar flamelet libraries [22] follow a similar philosophy but output a flame speed rather than time-advanced chemical concentrations. Typically the construction of a suitable model requires the solution of the chemical kinetics equations at several discrete chemical compositions followed by a parameterization procedure to evaluate the solution over a continuous range of input chemical compositions. Further utilization of the model does not require additional ODE integration, and is far less expensive. Because of the strongly nonlinear character of kinetics equations, Φ of Eqn. 2 is only valid for some localized neighborhood in chemical composition space; both the extent of this neighborhood and the order of the parameterization control the model accuracy. As examples: in PRISM Φ is a set of quadratic polynomials defined within a hypercube; in ISAT it is a set of linear polynomials defined within an ellipsoidal region. A key question in justifying these approaches is whether there is sufficient reuse of these localized neighborhoods to offset the initial construction cost.

In PRISM we describe the thermo-chemical state of a fluid with N_s species and reaction temperature at any instant of time as a point $\mathbf{r}(t) \equiv (C_1^t, C_2^t, \ldots, C_{N_\mathrm{s}}^t, T^t)$, in $\mathcal{C}$, where $\mathcal{C}$ is chemical composition space of N_s+1 dimensions. A solution–mapping technique is used, in which the result of a time-integration of the chemical rate equations is parametrized by a set of algebraic quadratic polynomial response surfaces in $\mathcal{C}$, i.e. the replacement of $\mathcal{F}$ by Φ. The solution–mapping is done piecewise on hypercubes which partition $\mathcal{C}$ and contain a distinct Φ parameterization for each hypercube. In principle enthalpy could be used instead of temperature, and could be advantageous in most systems as enthalpy varies only slightly in many systems. After much thought we chose temperature as it is more naturally an independent variable, and it is explicitly present in the Arrhenius rate equations whose time-integrals are being approximated by the polynomials. Additionally, the CFD codes we use provide temperature rather than enthalpy as an input variable. The conversions between temperature and enthalpy would add an additional efficiency penalty to the polynomial evaluation.

As an example, a pure chemistry case with zero spatial dimensions has a single chemical trajectory governed only by the chemical ODE system. Evolving a concentration through time in this manner entails successive evaluations, with the response from one time-step used as input for the next. The location of the initial point $\mathbf{r}(t_0) \equiv [C_1^{t_0}, C_2^{t_0}, \ldots, C_{N_\mathrm{s}}^{t_0}, T^{t_0}]$ determines the first hypercube for which polynomials are to be constructed. Perhaps for a few time-steps the polynomials of this hypercube will successively map

$\Phi(\mathbf{r}(t_0), \Delta t_0) \rightarrow \mathbf{r}(t_1)$, where $t_1 = t_0 + \Delta t_0$,

$\Phi(\mathbf{r}(t_1), \Delta t_1) \rightarrow \mathbf{r}(t_2)$,

$\Phi(\mathbf{r}(t_2), \Delta t_2) \rightarrow \mathbf{r}(t_3) \ldots$

all the while remaining in the same hypercube, but at some time $t = t_n$, the solution $\mathbf{r}(t_{n+1})$ will fall outside the hypercube. Figure 1 depicts this with an example that uses two chemical species, where the hypercubes are squares. When the solution falls outside the hypercube we must determine within which new hypercube the point $\mathbf{r}(t_{n+1})$ lies, construct new polynomials for it, and so on.

[Figure 1 about here.]

In a previous study [18] 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 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.

In this paper we investigate two methods to exploit the skewness of the reuse distribution. The first method makes an *a priori* estimation of the number of expected reuses to identify hypercubes that will ultimately not be used much. A hypercube can have a high degree of reuse as a result of being in a part of $\mathcal{C}$ where trajectories move relatively slowly, and thus take many steps as they move through the hypercube. The method anticipates high reuse by utilizing the rates at which trajectories are moving, to calculate a trajectory "velocity" V_{Tr}, either using solely chemical rate information, or by including the contribution to trajectory displacement from CFD effects. The resulting quantity V_{Tr} is then combined with an estimated trajectory length through the hypercube to determine the number of expected reuses, N_{ER}. On the basis of N_{ER} 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. Until then the ODE solver is called. Although it appears that this method has the disadvantage that the reuse distribution is not known until the problem has been run once, in practice this is not serious since simulations are frequently re-run with small changes in input parameters and the shape of the reuse distribution does not change significantly between runs. All the reuse distributions we have encountered during the course of simulations are amenable to this method.

In Section 2 we briefly review the PRISM method and describe the two new methods by which hypercube reuse is improved. In Section 3 these methods are applied to four cases: a point reaction case with zero spatial dimensions and 3 CFD simulations: a 1-D laminar premixed H_2+air flame, a 2-D premixed H_2+air turbulent jet, and 2-D non-premixed H_2 and air turbulent jets. Notation is defined in Appendix A.

2 Method

2.1 Prism Overview

Solution mapping [16, 17] makes numerical chemistry calculations more efficient by parameterizing the time–integration of the chemical rate equations with a set of algebraic polynomial response surfaces in chemical composition space $\mathcal{C}$. We partition $\mathcal{C}$ into non-overlapping adjacent hypercubes with a distinct polynomial parameterization for each. The hypercube edges and corners are restricted to regular intervals along the axes. This allows us to use a simple indexing for each hypercube, that permits fast and efficient searching when locating a hypercube. 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 $N_\mathrm{s}+1$ polynomials for the hypercube. These $N_\mathrm{s}+1$ polynomials taken together comprise the inexpensive mapping (see Eqn. 2) and determine the time evolution of *any* input point located in the hypercube. By evaluating them 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 polynomial coefficients and hypercube coordinates are stored in a data structure for subsequent reuse. Detailed descriptions of the steps of the procedure follow.

Polynomial Construction

The N_s+2 dimensional space is divided into hypercubes of a predetermined size, with one axis assigned to each species, one to temperature, and one to the time-step Δt. This last dimension has been added as a convenience so that Δt can be treated as a continuously varying input variable. The concentrations and the time-step are transformed into their logarithms and temperature into its reciprocal, a choice that reflects the Arrhenius form of the reaction rate equation. This transformation produces a better quadratic parametrization; for instance, use of reciprocal temperature results in error reduction of approximately 30%.

Determination of the polynomial expression begins by locating the hypercube in which the given input point sits. Once this has been done, to parametrize the response of the ODE system, the ODE solver needs to be called repeatedly at selected points about the hypercube. Each point corresponds to a set of concentrations, a temperature and a time-step length. The concentrations and temperature are propagated by the ODE solver for the length of time specified, returning a set of concentrations and temperature.

In order to produce a quadratic polynomial of $N_\mathrm{v} = N_\mathrm{s} + 2$ variables with $1 + N_\mathrm{v} + \frac{1}{2}N_\mathrm{v} \cdot (N_\mathrm{v} + 1)$ constant, linear and quadratic coefficients, we use the methods of surface response theory to ensure that number and placement of the points preserves the coefficient resolution, i.e., the ability of the regression algorithm to accurately calculate coefficient values of the order required. The optimal placement of these points in our problem is determined by the use of orthogonal composite designs based on the 2_V^{11-4} fractional factorial design [23]. The points are largely on corners of the hypercube, with an additional point at the hypercube center and one more outside each face, known as a star point (see Fig. 2 for a diagrammatic 3 D representation). The superscript on the design nomenclature indicates the number of points on the corners and the Roman numeral subscript is a measure of the mixing of the polynomial coefficients. The design 2_V^{11-4} has 128 points on the corners, with the center point and 22 star points increasing this to 151. Orthogonal designs have an important numerical advantage because determination of the polynomial expressions from the computed data points uses a diagonal covariance matrix, thus avoiding the general solution of the normal equations.

[Figure 2 about here.]

We found it useful to shift the corner points slightly toward the hypercube center because placing a point exactly at a corner wastes resources by unnecessarily parameterizing space outside the physical hypercube. By trial and error we found that reducing the distance of these points to the hypercube center by 25% maximized the accuracy.

To obtain the concentrations, $C_i^{t+\Delta t}$, and temperature, $T^{t+\Delta t}$, at the end of a time-step, the construction process results in a polynomial expression composed of quadratic terms of each species concentration, C_i^t, the temperature, T^t, and time-step, Δt. Thus the concentration of species i, $C_i^{t+\Delta t}$, is given by:

$$\begin{aligned}\log C_i^{t+\Delta t} &= a_{i,0} + \sum_j^{N_\mathrm{s}} a_{i,j} \log C_j^t + a_{i,N_\mathrm{s}+1} \frac{1}{T^t} + a_{i,N_\mathrm{s}+2} \log \Delta t \\ &\quad + \sum_j^{N_\mathrm{s}} \sum_{k \le j}^{N_\mathrm{s}} a_{i,jk} \log C_j^t \log C_k^t + a_{i,N_\mathrm{s}+1\ \ N_\mathrm{s}+1} \frac{1}{T^t} \cdot \frac{1}{T^t} + a_{i,N_\mathrm{s}+2\ \ N_\mathrm{s}+2} \log \Delta t \cdot \log \Delta t \\ &\quad + \text{cross terms between the } C_j, T, \Delta t \qquad (3)\end{aligned}$$

where a_i are the polynomial coefficients determined in this procedure. The number of the coefficients in this quadratic polynomial of $N_\mathrm{v} = N_\mathrm{s}{+}2 = 11$ independent variables is $1{+}N_\mathrm{v}{+}\frac{1}{2}N_\mathrm{v}{\cdot}(N_\mathrm{v}{+}1) = 78$, which accounts for constant, linear and quadratic relationships between variables. It is now possible to evaluate the $N_\mathrm{s}{+}1$ polynomials and obtain the time evolution of *any* input point within the hypercube.

Hypercube Storage and Retrieval

We now organize hypercube information for placement into a data structure for future re-use. First the hypercube is indexed via the coordinates at its center (concentrations, temperature and Δt) and because we have restricted hypercubes to lie only at regular positions, the indices have simple integer values. A key is constructed by concatenating the indices in each dimension in a way that is reproducible and unique. The information placed in the data structure consists of this key, the hypercube size and position, and the polynomial coefficients for $N_{\rm p}$=$N_{\rm s}$+1 polynomials in $N_{\rm v}$=$N_{\rm s}$+2 variables. The number of coefficients is:

$$N_{\rm p} \times \left[1 + (N_{\rm v}) + \frac{(N_{\rm v})(N_{\rm v}+1)}{2}\right] \tag{4}$$

Multiplying $N_{\rm p}$ by the floating point representation size (8 bytes) gives a memory requirement of about 8 kilobytes per hypercube for $N_{\rm s}$=9, of which the coefficients account for the major portion.

The data structure complex consists of a memory-resident binary search tree, a memory-resident doubly linked list and a direct access disk file, all associated through cross-referencing pointers. The binary tree has the key of a hypercube at each node. During a search for a hypercube, when a key is matched, the polynomial coefficients are retrieved from either the memory-resident list or the disk file. Additionally, the hypercubes can be stored on disk between simulations and reused in subsequent runs that have the same reaction set and differ only slightly in physical conditions, giving a substantial saving on repeat calculations.

In summary, given a point $\mathbf{r}^t(C_1^1, C_2^t, \ldots, C_{N_{\rm s}}^t, T^t)$ and a time-step Δt, the PRISM procedure is comprised of the following steps:

1. Determine within which hypercube it lies and calculate the key.

2. Traverse binary search tree and search for the key.

3. If key is found retrieve the coefficients from either memory or disk. If key is not found, construct polynomials.

4. Evaluate the polynomials to obtain $\mathbf{r}^{t+\Delta t} = \phi(\mathbf{r}^t, \Delta t)$.

Accuracy

For PRISM to be useful it needs two qualities: accuracy and economy. The second is quite obvious: a significant calculational speedup is all that is required, and we discuss this later. The question of accuracy is more difficult to quantify. What is an acceptable error per time-step? Do these errors propagate in simulation time giving increasingly inaccurate results? If so, can these errors be reduced to provide sufficiently accurate results? The issue of errors is quite challenging for time-dependent because we cannot recognize an unacceptable solution unless it appears physically implausible.

A measure of accuracy available at the moment of construction would be very helpful to gauge the validity of a hypercube's polynomial expressions. Since we determine the exact solution at many points within the hypercube during the polynomial construction phase, we simply evaluate the polynomials at the same points to obtain the relative error and use the root mean square of the residuals as a measure of accuracy of the hypercube.

We apply a conservative criterion for local accuracy that is chosen to preserve a high level of global accuracy. Specifically, we require that each species within a hypercube has a relative error 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. For our case studies we find that $< 5\%$ of the hypercubes are not sufficiently accurate when these criteria are applied. When a polynomial fit is rejected by the error criteria, the hypercube is tagged as being "inaccurate", and subsequently the ODE solver is used instead of evaluating the polynomial. Therefore, an inability to meet the accuracy requirements has consequences only for the economy of the method, specifically only the cost of constructing polynomials is wasted, but the accuracy of the simulations is not sacrificed. A small fraction of "inaccurate" hypercubes actually suggests optimal hypercube size has been achieved since it indicates that the hypercubes have not been sized too small. This approach to error control maintained global concentration errors of 0.1–0.5% in the present computations. The factors that influence the accuracy of the polynomial expression are:

Hypercube size: Accuracy improves as hypercube size decreases and parametrization becomes more accurate. The hypercube should be as large as possible without accuracy being sacrificed because this reduces the total number of hypercubes, e.g., filling a given region of space with hypercubes of half the size would require 2^{N_s+2} as many hypercubes! With respect to expense of generation as well as storage requirements, larger sides are advantageous. Table 1 indicates the trend of decreasing error with decreasing edge size, as illustrated on lines 1,2, & 3, where the size is varied from 0.25 to a full order of magnitude. Independently we have also varied the temperature interval, (see lines 1,4 & 5) and observe the same behavior. We note that the hypercube sides can be relatively large, 0.25–0.5 of an order of magnitude in concentration and time-step, and 10–20 K for temperature. This is of course dependent on the chemical mechanism, temperature and pressure. We have found that this resolution holds for all our case studies where temperature varies from 300K–2000K, pressure is atmospheric, and in the non-premixed case where there is a range of local stoichiometries.

[Table 1 about here.]

Number of points: The accuracy also depends on the number of points used in the polynomial fitting procedure. Decisions that affect the number and placement of these points are described earlier in sub-section 2.1. Lines 1,6 & 7 of Table 1 show the degradation of accuracy as the number of points is decreased. For our 11 dimensional application we changed the number of points used and considered 151, 87 and 55 points. Increasing from 87 to 151 points slightly improves accuracy, and decreasing to 55 points reduces accuracy significantly.

Enthalpy and elemental mass conservation: We are able to take advantage of the conservation relations that constrain the problem of intent. The mapping of a point over a time-step should conserve both total enthalpy and number of atoms of each element involved. While the ODE solver that supplies the points conserves these quantities very well, the resulting fitted polynomials do not always do so and the resulting solution is a small distance from the exact final solution. Part of the inherent accuracy loss in using a parametrized polynomial may be restored by enforcing conservation. In N_{s}+1 space the exact reaction trajectory of the system will always follow a hypersurface on which enthalpy and mass are conserved. By applying mass and enthalpy conservation our final point is shifted back onto that hypersurface, not necessarily onto the exact final point but hopefully closer to it. (We have an under-determined system with N_{s}+1 variables but only $\mathrm{N}_{\mathrm{elem}}$+1 conservation equations where $\mathrm{N}_{\mathrm{elem}}$ is the number of elements in the system.) We impose conservation after the polynomial has been evaluated by adjusting the final values of the species concentrations using a simple approach consisting of the following steps: (1) Set up $\mathrm{N}_{\mathrm{elem}}$+1 conservation equations; (2) allow $\mathrm{N}_{\mathrm{elem}}$+1 species concentrations to vary, requiring that the remaining concentrations be constant; (3) solve the resulting $\mathrm{N}_{\mathrm{elem}}$+1 simultaneous linear equations. The choice of species to be designated as variable is particularly important, and our criterion is to keep to a minimum the relative change in species concentration that occurs after the linear system solution. For each conserved element, the algorithm selects as a variable, the most abundant species which contains that element. Since we have an additional equation for enthalpy conservation, of the remaining species, that with largest concentration is also chosen as a variable regardless of its elemental makeup.

The 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 $266 : \frac{1}{15} : 1$. PRISM is intended for use in an operator-split CFD simulation where the response from a PRISM operation is passed through several CFD operations, such as diffusion and convection, and their output is used as input for the PRISM in the next time-step.

2.2 Trajectory Velocity (V_{Tr}) and Expected Reuse (N_{ER})

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. In short, this is accomplished (described below) by dividing the expected length of trajectory through the hypercube by the expected length for each time-step. The N_{ER} calculation costs significantly less than polynomial construction.

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 $\mathcal{C}$. A second approach, more accurate and expensive, but still small compared to the cost of polynomial construction, is to integrate the ODE over a single time-step Δt. In between these two extremes are other possibilities, such as linearly extrapolating V_{Tr} to the point where it exits the hypercube and taking the average of the initial and final values, or using the value of V_{Tr} at the center of the trajectory, and so on. These various possibilities for V_{Tr} were evaluated and we selected the simplest, first order method as there was insufficient improvement to justify the more complicated methods. The cost associated with this method is derived from the chemical rate contributions to species production, and is much less than the cost of polynomial construction, or even that of an ODE call. The three costs are in the ratio $\frac{1}{7}$: 266 : 1 respectively.

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

$$N_{\mathrm{ER}} \equiv \frac{trajectory\ length}{\mid V_{\mathrm{Tr}} \mid \cdot \Delta t} \tag{5}$$

Whenever the trajectory enters a new hypercube we calculate N_{ER}. If N_{ER} is above break-even 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.

Up to this point V_{Tr} has been derived using trajectory movement solely from chemical rates. In some of our case studies, the influence of CFD on trajectory movement through $\mathcal{C}$ is significant. For these we replace $\mid V_{\mathrm{Tr}} \mid \cdot \Delta t$ 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. We acknowledge that it is undesirable for the chemistry module to require any knowledge of the CFD solver, beyond the chemical state at the beginning of the time-step and the time-step length itself. Often, for any particular hypercube we are simultaneously measuring reuse from the trajectories of up to 20 CFD cells at a given time and we must determine if any of these has sufficient expected reuse to warrant hypercube construction. As a result we need to differentiate between them. The chemistry module's knowledge of the external CFD solver is used solely for book-keeping purposes and does not affect any internal chemistry calculations. 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, but 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.

The length of the timestep is also treated as a hypercube dimension (see Eqn. 3). Possible changes in it are not treated by the V_{Tr} method and could conceivably result in a hypercube with a slow V_{Tr} being used less than foreseen. A large change in Δt caused by a sudden fluctuation in fluid conditions could result in many constructed hypercubes ceasing to be used. The value of Δt is handed down by the parent CFD code.

2.3 Deferred Polynomial Construction (DPC) Method

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 $f(N)$:

$$\textit{Total cost} = \int_0^{N_{\mathrm{D}}} C_{\mathrm{ODE}} \cdot f(N) N dN + \int_{N_{\mathrm{D}}}^{N_{\max}} [C_{\mathrm{ODE}} \cdot N_{\mathrm{D}} \ + C_{\mathrm{PC}} \ + C_{\mathrm{PE}} \cdot (N - N_{\mathrm{D}})] \cdot f(N) dN \quad (6)$$

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.6 to the special case where polynomial construction is not deferred.

To gain some insight into the problem we chose several several simple functions for $f(N)$, and analytically evaluated and differentiated Eqn.6 with respect to N_{D} to find the value of N_{D} which resulted in minimum cost. Function #1 was a constant straight line function; functions #2–5 all decreased with increasing N, either exponentially, linearly, or as the reciprocal of a power. Table 2 shows whether such cost minima existed, and if so, their location.

[Table 2 about here.]

For the flat function (#1) and the exponential (#2), minima were never found. This was not a surprise for function #1, as we intuitively expect to find minima only in cases where $f(N)$ is large for small N and drops sharply. However, the lack of a minimum for the exponential tells us that we also need substantial counts at large N: exponential forms decline too quickly. For the linearly declining (#3) and the two reciprocal power functions (#4,5), minima exist. For #3, $N_{\mathrm{D}} \geq N_{\mathrm{B}}$ (the break-even usage) while for #4,5 it lies between 1 and N_{B}. Additionally, for #3 $N_{\max}$ must be greater than $3N_{\mathrm{B}}$, explicitly indicating that the tail must be significant. The minima in all these cases occur as a result of opposing contributions from:

- hypercubes with $N < N_{\mathrm{D}}$ (which never undergo polynomial construction)
- hypercubes with $N_{\mathrm{D}} < N < N_{\mathrm{B}}$ (for which polynomial construction is a waste, but is done anyway)
- hypercubes with $N > N_{\mathrm{B}}$ (for which polynomial construction is beneficial)

The relative populations in these groups depend on the shape of the function and the parameters N_{D} and $N_{\max}$. This exercise with analytical functions gives us valuable insight into the shapes of reuse distributions that would benefit from a DPC cut. It also tells us that the value of N_{D} is close to N_{B}. We also note that the minima are shallow, which implies that searching for the most optimal N_{D} will not net us much more gain than simply setting N_{D} to a reasonably close value, such as N_{B}. Using the definitions and derivations in Appendix B we develop a simple formula which shows the computational gain that results from moving from $N_{\rm D}$=0 (no DPC) to $N_{\rm D}$=$N_{\rm B}$, for any simple integrable $f(N)$:

$$\text{Computational Gain} = C_{\rm PC}[N_<(1 - \frac{\overline{N_<}}{N_{\rm B}}) - N_>] \tag{7}$$

Using this, a PRISM user can determine whether it is worthwhile setting the deferred polynomial construction threshold to $N_{\rm B}$ given *any* hypercube reuse distribution $f(N)$. The quantities in Eqn. 7 are easily determined from $f(N)$. There may have been a more optimal value of $N_{\rm D}$ somewhere between 0 and $N_{\rm B}$ but it would take far more effort to extract it. A look at Eqn. 7 tells us that unless $f(N)$ has $\overline{N_<} \rightarrow N_{\rm B}$ or unless $N_<$ and $N_>$ are comparable, the gain should be substantial.

[Figure 3 about here.]

In Figure 3 we show the reuse distributions from two of the simulations that we have run. (During the simulation, hypercube reuse statistics are gathered, and at the end a hypercube reuse frequency distribution is plotted.) Both distributions have the properties necessary to make the DPC method practical: a large number of hypercubes with very low reuse and a tail (in excess of an exponential drop) at very high reuse.

3 Results and Discussion

The $V_{\rm Tr}$ and DPC methods are tested on 3 different time-evolving reactive flow simulations. The Coyote CFD code [24] is used, within which the chemistry calculation uses the DVODE differential equation solver [25] and the CHEMKIN thermodynamic library [26]. An additional 0-dimensional chemistry-only case is run for the $V_{\rm Tr}$ method. All simulations use the 9-species H_2+air reaction set described in Table 3, obtained by removing carbon from the GRI-Mech 2.11 mechanism [27]. The factorial design for constructing hypercubes ($2^{11-4}_{\rm V}$) uses 151 points, with hypercube side lengths of

$0.1{\cdot}log_{10}(concentration)$ for the 3 major species H_2, O_2 and H_2O, $0.25{\cdot}log_{10}(concentration)$ for the other species, 20 K for temperature and $0.2{\cdot}log_{10}(timestep)$. These values are chosen for reasons of accuracy.

For each case we show how the 2 methods result in fewer polynomial being constructed, and the resulting computational gain will be shown on plots of CPU time vs. timestep number. Only the portion of CPU time utilized in chemistry calculations is reported.

[Table 3 about here.]

Zero-Dimensional

The first example considers premixed combustion purely in chemical composition space $\mathcal{C}$, with zero physical dimensions so that the reaction trajectory is determined solely by chemical kinetics. The initial mixture is stoichiometric H_2-air at 1200 K, high enough for burning to commence. The timestep is fixed at 10^{-7} s. Once the simulation is started the fuel is consumed and the system comes to rest adiabatically at an equilibrium temperature of 2819 K after 250 μs.

[Figure 4 about here.]

Figure 4a shows the correlation of N_{AR} with N_{ER}, with many points falling on the dashed line (N_{AR} = N_{ER}). The actual hypercube reuse, N_{AR} has integer values, while N_{ER} can have fractional values. Since for a 0-D calculation there is only one trajectory, N_{ER} was recalculated for each timestep through each hypercube and the value plotted was the largest N_{ER} encountered by the trajectory during the traversal of each hypercube.

1-Dimensional Laminar Flame

The second example is a more complex system that 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 end to atmospheric pressure. 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 4 shows the effect of including or excluding the effect of fluid mechanics on trajectory movement from th N_{ER} calculation. Examining a plot of N_{AR} against N_{ER}, (Figure 4b) where N_{ER} is calculated solely using chemical information, reveals that N_{AR} and N_{ER} exhibit less correlation than seen in the 0-dimensional case (Fig. 4a). The correlation is improved if N_{ER} is calculated taking into account the effect of fluid mechanics on trajectory movement (Fig. 4c). The overall effectiveness of the method is illustrated by Fig. 5a 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 it did 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 $N_{\mathrm{ER}} > 250$ 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 which does not construct polynomials until a preuse threshold (set equal to the break-even usage N_{B}) has been met. N_{B} depends mainly on the chemical reaction set, and is 266. Figure 5a shows the accumulated CPU time. As with the $N_{\mathrm{ER}} > 250$ case, far fewer polynomials are constructed, resulting in a 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. Table 4 summarizes the performance with information on number of hypercubes constructed, their mean reuse, CPU time used in chemistry, and total CPU time.

[Figure 5 about here.]

[Table 4 about here.]

2-D Axisymmetric Premixed Turbulent Jet

As a third example we have simulated a premixed H_2+air 2-D turbulent jet, starting from a quiescent non-combusting state and proceeding until a turbulent flame has developed. For this and the following non-premixed case study, an LES turbulence model [28] is applied in Coyote in which sub-grid features are modeled with flux-gradient approximations using an eddy viscosity. The sub-grid length-scale filter is about 3 times the CFD cell size. 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 5b shows the accumulated CPU time curves for the same 4 cases as were shown in the laminar flame. The DPC method gives the best result. 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. Again, Table 4 summarises the cases with information on number of hypercubes constructed, their mean reuse, and CPU times.

2-D Axisymmetric Non-premixed Turbulent Jet

The final 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 5c 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. Table 4 gives summary performance information. The non-premixed case differs from the previous cases in that it accesses a substantially larger portion of $\mathcal{C}$. Addtitionally, during the simulation, we noticed that a factor

2 reduction in Δt occurred for a short period at the beginning, as the CFD code was attempting to avoid numerical instability. Both these effects resulted in a larger number of hypercubes being created. The “no cut” suffers substantially as a result of this. The N_{ER} calculation detected that most of these low Δt hypercubes had insufficient reuse and so did not construct polynomials for them. However, in general this case did not give as good improvement over the ODE as the other 2 cases, probably because of the larger portion of $\mathcal{C}$ accessed.

Summary and Conclusion

The PRISM method improves the computational performance of a reactive flow calculation through a solution–mapping technique that replaces the time–integration of the chemical rate equations with a set of algebraic polynomial response surfaces. This paper describes two approaches for improving the efficiency of PRISM. It is important to clarify the costs incurred running PRISM, those associated with polynomial construction and those associated with polynomial evaluation. The two methods reduce the cost of polynomial construction, accomplishing this by constructing polynomials only for hypercubes that exhibit a high degree of reuse during the course of the simulation. They each improve PRISM’s efficiency by approximately a factor of two but cannot be used simultaneously to achieve a factor of four because they each identify approximately the same subset of highly reused hypercubes.

The first approach, the Trajectory Velocity (V_{Tr}) method, employs the rate of movement of a chemical trajectory combined with an estimated path length through a hypercube to estimate the expected hypercube reuse. For the chemistry-only case, the reuse is derived solely from the chemical rate equations. Three cases with fluid dynamics (FD) were investigated: a propagating laminar flame, a turbulent premixed flame, and a turbulent diffusion flame. In the three cases that include FD, that portion of the trajectory displacement caused by the FD must also be included in the reuse evaluation because it improves the correlation between expected and actual reuse. Most hypercubes are visited by multiple trajectories since the FD cases involve a grid with multiple cells, each with its own chemical trajectory. The overall reuse of the hypercube is heavily influenced by the contribution from the trajectories that utilize it most. This is accounted for by recalculating expected reuse every time a new trajectory enters the hypercube until sufficient reuse is indicated, at which time polynomials are constructed. In the three flames considered, the efficiency was improved by a factor of 1.5 to 2.5.

The second approach to improving PRISM's efficiency is referred to as the Deferred Polynomial Construction (DPC) method. Efficiency is improved by deferring polynomial construction until a hypercube has been reused a specific number of times (N_{D}). We demonstrate that it is not necessary to have detailed knowledge of the shape of the hypercube reuse function to improve efficiency. Substantial efficiency gains are found as long as the reuse distribution is skewed toward low values with a long tail in excess of that of an exponential distribution. It is unnecessary to determine the value of N_{D} that maximizes the efficiency because reasonable values can be obtained by setting N_{D} equal to the breakeven usage for hypercube construction. All the distributions encountered in our case studies were well suited to DPC. In the three cases considered, the efficiency was improved by a factor of 1.5 to 2.5.

Preliminary results with CH_4 flames have shown that the V_{Tr} and DPC methods are vital toward increasing hypercube reuse over and beyond the level of viablilty. As PRISM development toward CH_4 combustion progresses, one of the methods will definitely be chosen for inclusion in the implementation.

Acknowledgements

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

Appendix A Notation

- **General**

N_{s}:	Number of chemical species
Δt:	Length of timestep
$\mathcal{C}$:	Chemical composition space of $N_{\mathrm{s}}+1$ dimensions

- **Trajectory Velocity-related**

V_{Tr}:	Trajectory velocity, the rate at which a chemical trajectory is moving through $\mathcal{C}$.
N_{ER}:	Number of Expected Reuses of a hypercube
N_{AR}:	Number of Actual Reuses of a hypercube

- **Deferrred Polynomial Construction-related**

C_{ODE}:	Cost of an ODE time-integration
C_{PC}:	Cost of polynomial construction for a hypercube
C_{PE}:	Cost of a polynomial evaluation
N_{B}:	Break-even usage of a hypercube, equal to $\frac{C_{\mathrm{PC}}}{C_{\mathrm{ODE}}-C_{\mathrm{PE}}}$

N_{D}**:** Reuse threshold for deferred polynomial construction of a hypercube.

Appendix B

For the DPC method we develop a simple formula which shows the computational gain that would result from moving from $N_{\mathrm{D}} = 0$ to $N_{\mathrm{D}} = N_{\mathrm{B}}$, for any reuse distribution that can be described by a simple integrable function $f(N)$, and show the conditions necessary for a substantial gain to be obtained. The aim is to use this approach rather than search for the optimal value of N_{D}, as it is a simpler approach. Using the definitions:

- $N_< \equiv \int_0^{N_{\mathrm{B}}} f(N)dN$ i.e. number of hypercubes with $N < N_{\mathrm{B}}$
- $N_> \equiv \int_{NB}^{N_{\max}} f(N)dN$ i.e. number of hypercubes with $N > N_{\mathrm{B}}$
- $\overline{N_<} \equiv \frac{1}{N_<}\int_0^{N_{\mathrm{B}}} f(N)NdN$, mean usage of hypercubes with $N < N_{\mathrm{B}}$
- $\overline{N_>} \equiv \frac{1}{N_>}\int_{N_{\mathrm{B}}}^{N_{\max}} f(N)NdN$, mean usage of hypercubes with $N > N_{\mathrm{B}}$

and the relation: $N_{\mathrm{B}} = C_{\mathrm{PC}}/(C_{\mathrm{ODE}} - C_{\mathrm{PE}})$, we calculate an expression for the gain, starting from a form of Eqn. 6. The total cost for all hypercubes with usage less than N_{B} is:

$$\begin{aligned}
Cost(N < N_{\mathrm{B}}) &= \int_0^{N_{\mathrm{D}}} C_{\mathrm{ODE}} N f(N)dN + \int_{N_{\mathrm{D}}}^{N_{\mathrm{B}}} f(N)[C_{\mathrm{ODE}}N_{\mathrm{D}} + C_{\mathrm{PC}} + C_{\mathrm{PE}}(N - N_{\mathrm{D}})]dN \\
&= \int_0^{N_{\mathrm{B}}} C_{\mathrm{ODE}} N f(N)dN + \int_{N_{\mathrm{D}}}^{N_{\mathrm{B}}} f(N)[C_{\mathrm{ODE}}N_{\mathrm{D}} + C_{\mathrm{PC}} + C_{\mathrm{PE}}(N - N_{\mathrm{D}}) - C_{\mathrm{ODE}}N]dN \\
&= C_{\mathrm{ODE}} N_< \overline{N_<} + \int_{N_{\mathrm{D}}}^{N_{\mathrm{B}}} f(N) C_{\mathrm{PC}}(1 + \frac{N_{\mathrm{D}} - N}{N_{\mathrm{B}}})dN \qquad (8) \\
&= N_<[C_{\mathrm{ODE}}\overline{N_<} + C_{\mathrm{PC}}(1 - \frac{\overline{N_<}}{N_{\mathrm{B}}})] \qquad \ldots \text{if} N_{\mathrm{D}} = 0 \qquad (9) \\
&= N_< C_{\mathrm{ODE}} \overline{N_<} \qquad \ldots \text{if} N_{\mathrm{D}} = N_{\mathrm{B}} \qquad (10)
\end{aligned}$$

So, the gain in moving N_{D} from 0 to N_{B} is their difference:

$$N_< C_{\mathrm{PC}}(1 - \frac{\overline{N_<}}{N_{\mathrm{B}}}) \qquad (11)$$

Similarly for hypercube with usage greater than N_{B}:

$$\begin{aligned} Cost(N > N_{\mathrm{B}}) &= \int_{N_{\mathrm{B}}}^{N_{\max}} f(N)[C_{\mathrm{ODE}}N_{\mathrm{D}} \ + C_{\mathrm{PC}} \ + C_{\mathrm{PE}}(N - N_{\mathrm{D}})]dN \\ &= N_{>}[C_{\mathrm{PC}}(1 + \frac{N_{\mathrm{D}}}{N_{\mathrm{B}}}) + C_{\mathrm{PE}}\overline{N_{>}}] \end{aligned} \tag{12}$$

and the gain in moving N_{D} from 0 to N_{B} is:

$$-N_{>}C_{\mathrm{PC}} \tag{13}$$

The negative sign signifies a loss as these hypercubes had to undergo unnecessary ODE solves for a while before polynomial construction. Combining Eqns. 11, 13 the total gain in moving N_{D} from 0 to N_{B} is:

$$C_{\mathrm{PC}}[N_{<}(1 - \frac{\overline{N_{<}}}{N_{\mathrm{B}}}) - N_{>}] \tag{14}$$

(also Eqn. 7)

References

[1] Tonse, S. Observations of the Coyote CFD reactive flow code, 1997.

[2] Frenklach, M.; Kailasanath, K.; Oran, E. S. Progress in Astronautics and Aeronautics 1986, 105, 365–376.

[3] Frenklach, M. Modeling of Large Reaction Systems. In Complex Chemical Reaction Systems, Mathematical Modelling and Simulation, Vol. 47; Warnatz, J.; Jäger, W., Eds.; Springer-Verlag: Berlin, 1987.

[4] Wang, H.; Frenklach, M. Combust Flame 1991, 87, 365–370.

[5] Frenklach, M. Reduction of chemical reaction models. In Numerical Approaches to Combustion Modeling; Oran, E. S.; Boris, J. P., Eds.; American Institute of Aeronautics and Astronautics: Washington, D.C., 1991.

[6] Hewson, J. C.; Bollig, M. Reduced mechanisms for NO_x emissions from hydrocarbon diffusion flames. In Twenty Sixth (International) Symposium on Combustion; The Combustion Institute: Pittsburgh, PA, 1996.

[7] Peters, N.; Williams, F. A. The structure of methane flames. In Complex Chemical Reaction Systems, Mathematical Modelling and Simulation, Vol. 47; Warnatz, J.; Jäger, W., Eds.; Springer-Verlag: Berlin, 1987.

[8] Ramshaw, J. D. Phys Fluid 1980, 23, 675.

[9] Maas, U.; Pope, S. B. Combust Flame 1992, 88, 239–264.

[10] Blasenbrey, T.; Maas, U. Proc Comb Inst 2000, 28, 1623.

[11] Bongers, H.; van Oijen, J. A.; de Goey, L. P. H., submitted for publication in Proc Comb Inst.

[12] Lam, S. H.; Goussis, D. A. Int J Chem Kinet 1994, 26, 461–486.

[13] Schwer, D. A.; Tolsma, J. E.; Green, W. H.; Barton, P. I. Combust Flame 2002, 128, 270–291.

[14] Petzold, L. R.; Zhu, W. A I Ch E Jou 1999, 45, 869.

[15] Brown, N. J.; Li, G.; Koszykowski, M. L. Int J Chem Kinet 1997, 29, 393–414.

[16] Frenklach, M. Reduced Mechanisms. In Combustion Chemistry; Springer-Verlag: Berlin, 1984; Chapter 7.

[17] Marsden, A. R.; Frenklach, M.; Reible, D. D. J Air Pollut Control Assoc 1987, 37, 370–376.

[18] Tonse, S. R.; Moriarty, N. W.; Brown, N. J.; Frenklach, M. Israel J Chem 1999, 39, 97–106.

[19] Turanyi, T. Comp Chem 1994, 18, 45–54.

[20] Pope, S. B. Combust Theory Modelling 1997, 1, 41–63.

[21] Yang, B.; Pope, S. B. Combust Flame 1998, 112, 85–112.

[22] Bray, K. N. C.; Peters, N. Laminar Flamelets in Turbulent Flames. In Turbulent Reacting Flows; Libby, P. A.; Williams, F. A., Eds.; Academic Press: San Diego, CA, USA. 92101-4311, 1994.

[23] Box, G. E. P.; Draper, N. R. Empirical Model Building and Response Surfaces John Wiley and Sons: New York, 1987.

[24] Cloutman, L. D. "Coyote: A computer program for 2D reactive flow simulation", Technical Report UCRL-ID-103611, Lawrence Livermore National Laboratory, 1990.

[25] Brown, P. N.; Byrne, G. D.; Hindmarsh, A. C. SIAM J Sci Stat Comput 1989, 10, 1038-1051 Also, LLNL Report UCRL-98412, June 1988.

[26] Kee, R. J.; Rupley, F. M.; Meeks, E.; 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, Sandia National Laboratory, 1996.

[27] Frenklach, M.; Wang, H.; Goldenberg, M.; Smith, G. P.; Golden, D. M.; Bowman, C. T.; Hanson, R. K.; Gardiner, W. C.; Lissianski, V. "GRI-Mech—An Optimized Detailed Chemical Reaction Mechanism for Methane Combustion", Technical Report GRI-95/0058, Gas Research Institute, 1995 http://www.me.berkeley.edu/gri_mech/.

[28] Cloutman, L. D. “The LUVD11 Large Eddy Simulation Model”, Technical Report UCRL-ID-107128, Lawrence Livermore National Laboratory, 1991.

List of Tables

Table 1:

	number of	bin width		accuracy (x 10^3)		
	data points	$\log_{10}$(conc)	T (K)	ΔT	ΔH_2O	ΔOH
1	87	0.5	20	1.02	6.34	9.46
2	87	0.25	20	0.15	1.09	1.86
3	87	1.0	20	2.89	40.64	41.35
4	87	0.5	10	1.00	6.44	11.06
5	87	0.5	40	1.13	7.03	9.49
6	151	0.5	20	0.79	4.70	9.12
7	55	0.5	20	3.00	23.71	38.74
8	87	0.5	20	1.29	6.35	9.46
9	151	0.25	20	0.091	0.658	0.678

Table 2:

#	$f(N)$	Cost minimum
1	$f(N) = k$	no minimum found
2	$f(N) = a_0 e^{-N/N_0}$	no minimum found
3	$f(N) = -mN + k$	N_{D} = N_{max} - 2N_{B}
4	$f(N) = \frac{k}{N}$	N_{D} = $\frac{N_{\mathrm{B}}}{log(\frac{N_{\mathrm{max}}}{N_{\mathrm{D}}})}$
5	$f(N) = \frac{k}{N^3}$	N_{D} = $\frac{N_{\mathrm{D}}^3}{N_{\mathrm{max}}^2} - 2N_{\mathrm{B}}$

Table 3:

reaction			forward rate coefficient[a] A	n	E
2O+M	$\rightleftharpoons$	O_2+M	1.2×10^{17}	-1.00	
O+H+M	$\rightleftharpoons$	OH+M	5.0×10^{17}	-1.00	
O+H_2	$\rightleftharpoons$	H+OH	5.0×10^{4}	2.67	6290.
O+HO_2	$\rightleftharpoons$	OH+O_2	2.0×10^{13}		
O+H_2O_2	$\rightleftharpoons$	OH+HO_2	9.63×10^{6}	2.00	4000.
H+O_2+M	$\rightleftharpoons$	HO_2+M	2.8×10^{18}	-.86	
H+2O_2	$\rightleftharpoons$	HO_2+O_2	3.0×10^{20}	-1.72	
H+O_2+H_2O	$\rightleftharpoons$	HO_2+H_2O	9.38×10^{18}	-.76	
H+O_2	$\rightleftharpoons$	O+OH	8.3×10^{13}		14413.
2H+M	$\rightleftharpoons$	H_2+M	1.0×10^{18}	-1.00	
2H+H_2	$\rightleftharpoons$	2H_2	9.0×10^{16}	-.60	
2H+H_2O	$\rightleftharpoons$	H_2+H_2O	6.0×10^{19}	-1.25	
H+OH+M	$\rightleftharpoons$	H_2O+M	2.2×10^{22}	-2.00	
H+HO_2	$\rightleftharpoons$	O+H_2O	3.97×10^{12}		671.
H+HO_2	$\rightleftharpoons$	O_2+H_2	2.8×10^{13}		1068.
H+HO_2	$\rightleftharpoons$	2OH	1.34×10^{14}		635.
H+H_2O_2	$\rightleftharpoons$	HO_2+H_2	1.21×10^{7}	2.00	5200.
H+H_2O_2	$\rightleftharpoons$	OH+H_2O	1.0×10^{13}		3600.
OH+H_2	$\rightleftharpoons$	H+H_2O	2.16×10^{8}	1.51	3430.
2OH(+M)	$\rightleftharpoons$	H_2O_2(+M)	7.4×10^{13}	-.37	
2OH	$\rightleftharpoons$	O+H_2O	3.57×10^{4}	2.40	-2110.
OH+HO_2	$\rightleftharpoons$	O_2+H_2O	2.9×10^{13}		-500.
{ OH + H_2O_2	$\rightleftharpoons$	HO_2 + H_2O	1.75×10^{12}		320.
{ OH + H_2O_2	$\rightleftharpoons$	HO_2 + H_2O	5.8×10^{14}		9560.
{ 2HO_2	$\rightleftharpoons$	O_2 + H_2O_2	1.3×10^{11}		$-1630.$
{ 2HO_2	$\rightleftharpoons$	O_2 + H_2O_2	4.2×10^{14}		12000.

[a] The forward rate coefficients $k = AT^n e^{-E/\mathrm{R}T}$; R is the universal gas constant, T is the temperature in K, the units of E are cal/mol.

Table 4:

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

Figure Captions

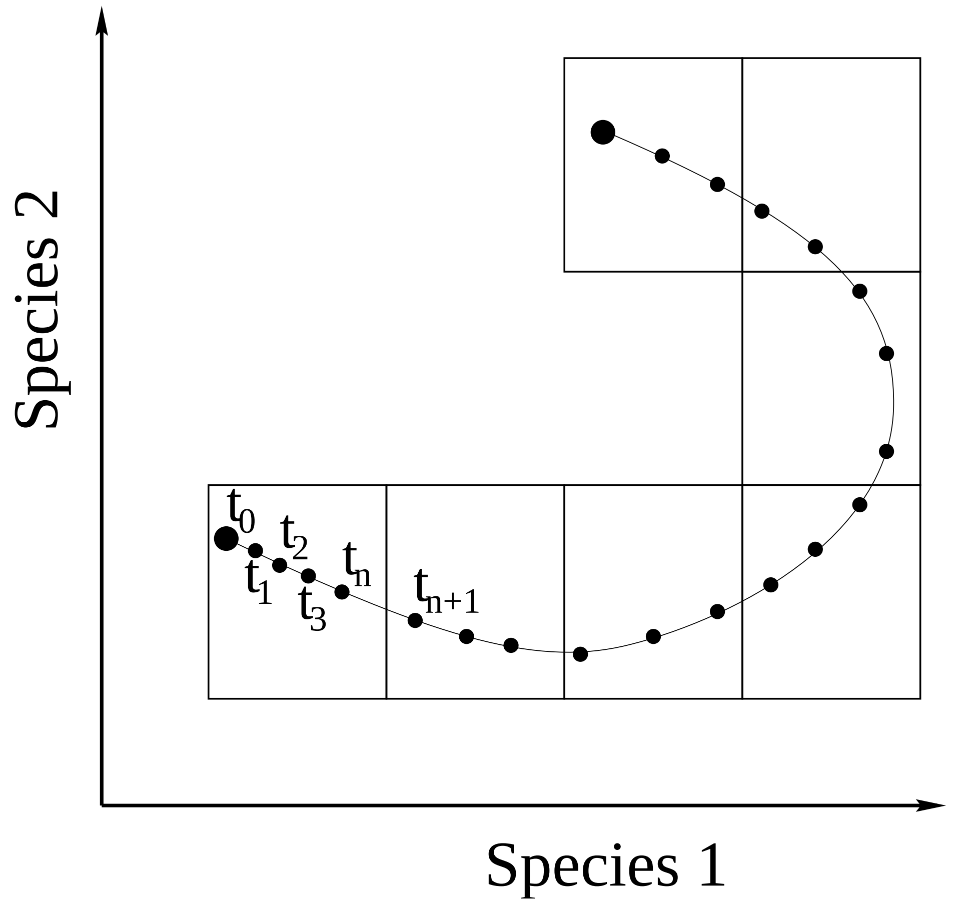


Figure 1:

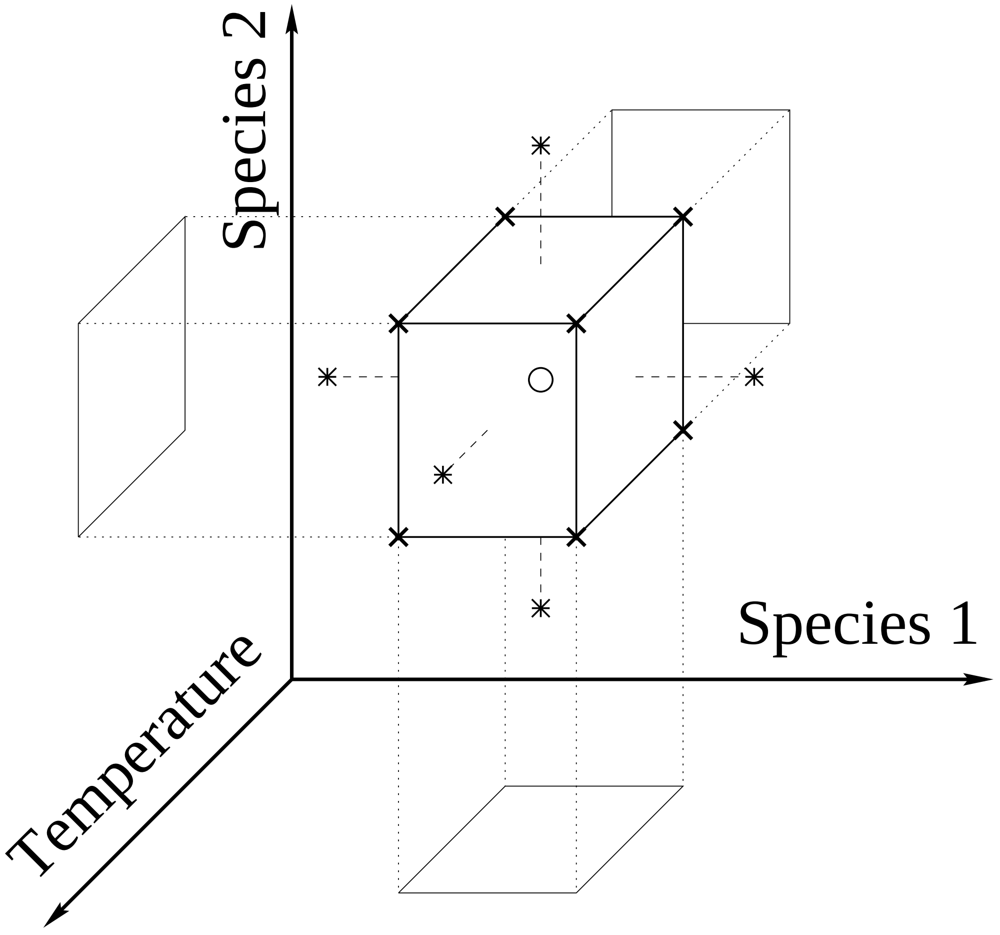
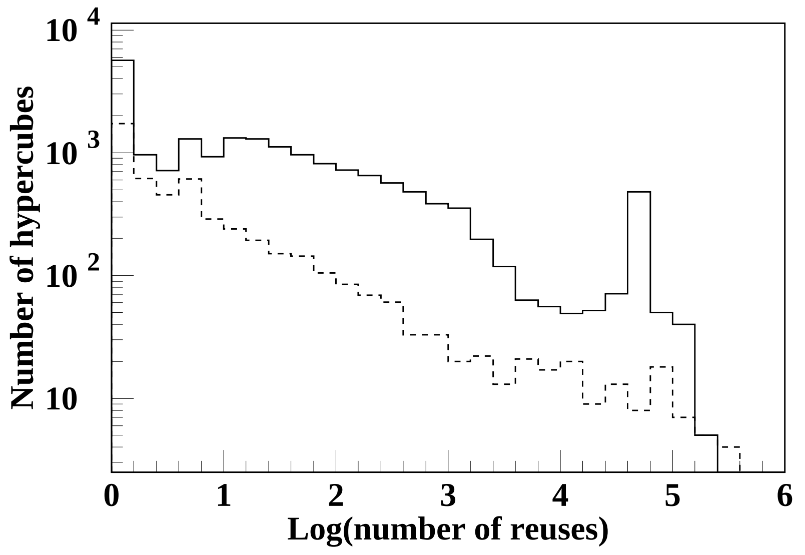


Figure 2:



10^4
10^3
10^2
10
Number of hypercubes
0
1
2
3
4
5
6
Log(number of reuses)

Figure 3:

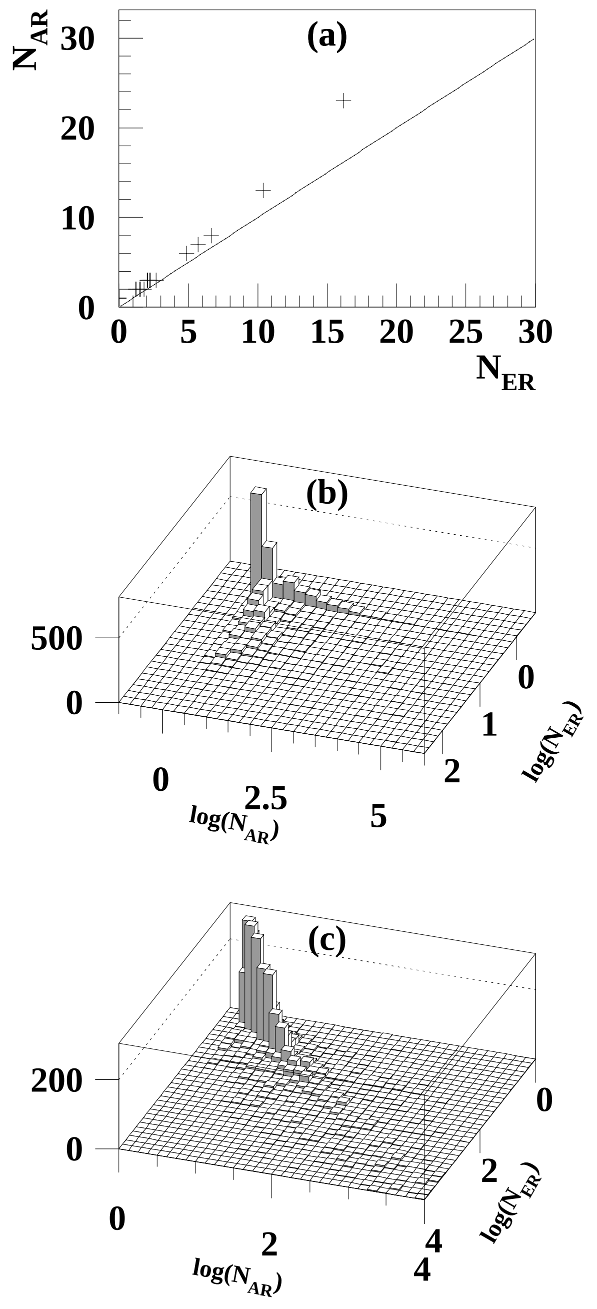
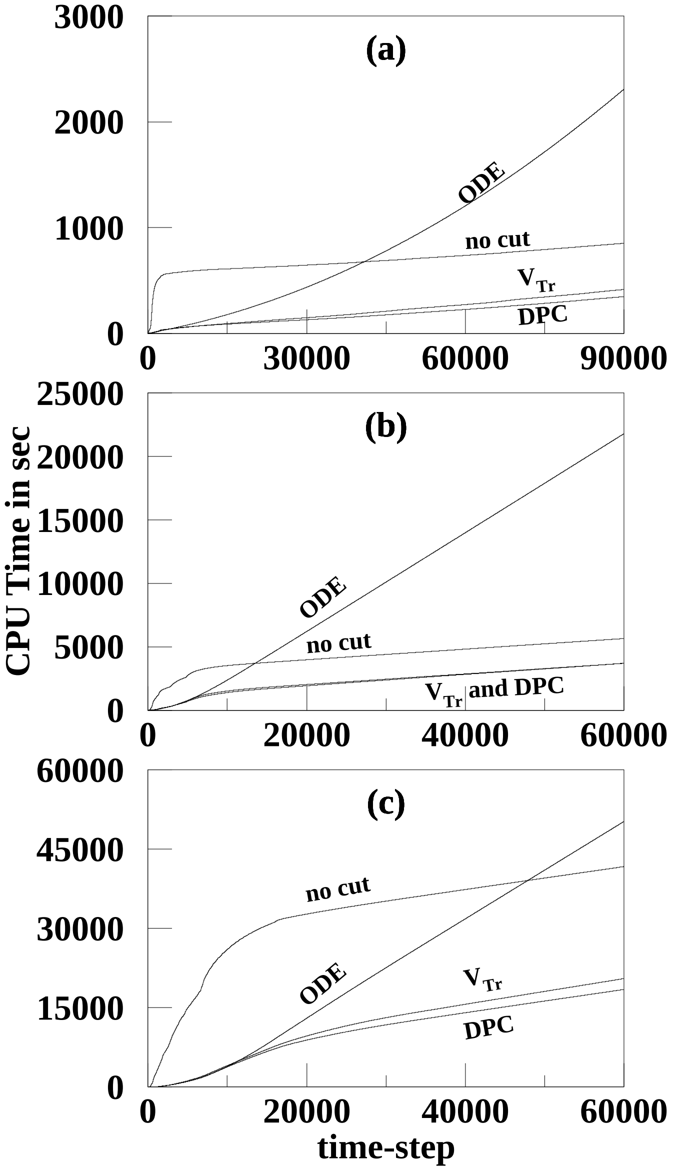

(a)
N_AR
N_ER
0
5
10
15
20
25
30
(b)
500
0
log(N_AR)
2.5
5
log(N_ER)
1
2
(c)
200
log(N_AR)
log(N_ER)
4

Figure 4:



(a)
ODE
no cut
V_{Tr}
DPC
(b)
ODE
no cut
V_{Tr} and DPC
(c)
no cut
ODE
V_{Tr}
DPC
CPU Time in sec
time-step

Figure 5: