Available online at www.sciencedirect.com
ScienceDirect
Combustion and Flame 147 (2006) 150–162
Combustion and Flame
www.elsevier.com/locate/combustflame

# Modeling unsteady reacting flow with operator splitting and ISAT

Michael A. Singer [a,*], Stephen B. Pope [b], Habib N. Najm [c]

[a] *Center for Applied Mathematics, Cornell University, Ithaca, NY 14853, USA*
[b] *Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA*
[c] *Combustion Research Facility, Sandia National Laboratories, Livermore, CA 94551, USA*

Received 14 November 2005; received in revised form 6 June 2006; accepted 17 June 2006

Available online 1 September 2006

**Abstract**

We examine the utility of in situ adaptive tabulation (ISAT) for the simulation of two-dimensional unsteady laminar reacting flow. The numerical scheme used to solve the low-Mach-number reacting flow equations is an operator-split projection scheme which incorporates ISAT by a Strang subsplitting procedure. The scheme is parallelized using a combination of OpenMP and MPI. ISAT is used for the pure reaction substeps, while convection and diffusion are treated explicitly by a stabilized Runge–Kutta method. We apply the scheme to a two-dimensional problem involving a laminar premixed methane–air flame interacting with a counterrotating vortex pair using detailed GRIMech3.0 chemical kinetics. Computational performance is examined; we observe an overall speed-up factor due to ISAT of approximately 2.5–3.
© 2006 The Combustion Institute. Published by Elsevier Inc. All rights reserved.

*Keywords:* Flame/vortex interaction; ISAT; Premixed flame; Runge–Kutta; Strang splitting

## 1. Introduction

Numerical simulations of reacting flow with detailed chemical kinetics are expensive computationally. Much of the computational cost is associated with the evaluation of detailed transport models and solving the equations that govern the combustion chemistry. At the same time, reacting flow simulations with detailed chemical kinetic modeling are becoming increasingly valuable components in the design and development of engines, combustors, and reactors. Further, simulation results are widely used as input in regulatory and business decisions [1]. As a result, the development and application of computationally efficient numerical approaches for reacting flows has broad application.

In [2], Singer et al. introduced the coupling of in situ adaptive tabulation (ISAT) [3] with the operator-split projection scheme of Najm and Knio [4]. This coupling is done via a Strang subsplitting procedure that further separates reaction from convection and diffusion. Singer et al. applied the scheme, in the context of a low-Mach-number reacting flow code (`dflame`), to a one-dimensional laminar premixed methane–air flame using GRIMech3.0 [5]. Second-order temporal convergence was demonstrated, as well as the impact of the ISAT error tolerance, $\varepsilon_{\text{tol}}$, on

* Corresponding author. Present address: Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
*E-mail address:* msinger@mit.edu (M.A. Singer).

0010-2180/$ – see front matter © 2006 The Combustion Institute. Published by Elsevier Inc. All rights reserved.
doi:10.1016/j.combustflame.2006.06.007

the accuracy of the solution. The use of ISAT to compute the reaction substeps resulted in a reaction substep speed-up of approximately 13; the overall CPU time speed-up was approximately 7.5.

The work described here further develops and extends the numerical scheme introduced in [2] and represents the first application of ISAT to the computation of multidimensional unsteady laminar flames. In particular, `dflame` and ISAT are used in parallel and applied to two-dimensional unsteady vortical flow involving a counterrotating vortex pair interacting with a premixed methane–air flame. This configuration has been studied extensively [6–8]; here we focus on computational performance issues rather than the revelation of new physics. In Section 2 we provide a brief review of the numerical algorithm described in [2,4]. We then discuss the parallel strategies used in `dflame` and ISAT and comment on the coupling of the two codes. Section 3 begins with a specification of the flame/vortex interaction problem. This introduction is followed by a qualitative demonstration of results and a discussion of the performance of the scheme and its use of ISAT. Section 4 provides additional discussion and comments on various aspects of the performance of the scheme. Finally, conclusions and final observations are made in Section 5.

## 2. Numerical algorithm

### 2.1. Overview

The numerical algorithm used here is the operator-split projection scheme of Najm and Knio [4] with the performance enhancements of Singer et al. [2]. The scheme is implemented in the Fortran code `dflame`, which uses a temporally second-order Strang splitting procedure to separate reaction from diffusion; convection is treated explicitly and accounted for in both the reaction and diffusion substeps. The reaction substeps are computed using ISAT and second-order Runge–Kutta (RK2) using a Strang subsplitting procedure detailed in Singer et al. [2]. The diffusion substeps are treated explicitly using a second-order Runge–Kutta–Chebyshev (RKC) [9] scheme detailed in Najm and Knio [4].

Second-order temporal convergence of the scheme has been demonstrated in the context of a one-dimensional laminar premixed flame [2]. In addition, the impact of the ISAT error tolerance, $\varepsilon_{\text{tol}}$, on the accuracy of the solution has been studied [2].

### 2.2. Serial ISAT

The ISAT algorithm is an information storage/retrieval procedure that is implemented as a binary tree; the binary tree is traversed based on cutting planes [3]. Given the state of a system (defined, for example, by species mass fractions, enthalpy, and pressure) at time $t_0$, $\boldsymbol{\phi}(t_0)$, and a time increment, $\Delta t$, ISAT determines $\boldsymbol{\phi}(t_0 + \Delta t)$, the state of the system at time $t_0 + \Delta t$. This new state is determined by approximating the solution to the equations governing a homogeneous, adiabatic, isobaric system of chemical species [3]. To determine $\boldsymbol{\phi}(t_0 + \Delta t)$, which is accurate to within a user-specified error tolerance $\varepsilon_{\text{tol}}$, ISAT uses one of four approaches:

- `primary retrieve`: the query composition, $\boldsymbol{\phi}(t_0)$, lies within the ellipsoid of accuracy (EOA) [3] of a termination node in the ISAT table. A linear approximation to the solution, $\boldsymbol{\phi}(t_0 + \Delta t)$, is returned;
- `secondary retrieve`: a `primary retrieve` was unsuccessful. The EOA of a nearby termination node contains the query composition and a solution is returned based on this nearby node;
- `grow`: a `secondary retrieve` was unsuccessful. ISAT is able to grow the EOA of a primary or nearby node to contain the query point. A solution is then returned;
- `add`: a `grow` was unsuccessful. The governing equations are integrated and a new entry is added to the ISAT table.

### 2.3. Parallelism

To minimize the wall clock time consumed by a reacting flow simulation, it is beneficial to use parallelism to accelerate the calculation. To this end, the algorithm is parallelized using both OpenMP and MPI. The parallel strategy that minimizes wall clock time, however, is dependent on a variety of factors including the problem at hand (e.g., steady, unsteady), the chemical mechanism, and the numerical scheme. As a result, there are many ways to parallelize ISAT, with each approach having strengths and weaknesses depending on the regime in which it is applied.

For the work considered here, we use the ISAT message passing library `x2f_mpi` [11]. `x2f_mpi` is a library built on top of ISAT that implements different message-passing strategies by using the message-passing interface (MPI). The `x2f_mpi` library was constructed for use with PDF methods for simulating turbulent reacting flow. As a result, the message-passing strategies implemented in `x2f_mpi` may not be optimal for laminar flame computations. In the present context, `x2f_mpi` randomly distributes grid node compositions among all the MPI nodes at the beginning of each reaction substep. Each MPI node then calls ISAT where composition queries are satis-

fied by `retrieving`, `growing`, or `adding` to the local ISAT table (each MPI node has its own ISAT table). The updated grid node compositions are then passed back to the appropriate MPI node. Note that, since each MPI node uses only its local ISAT table (which is independent of the ISAT tables residing on other MPI nodes), this random distribution approach avoids the $\mathcal{O}(n_s^2)$ (where $n_s$ is the number of chemical species) computational expense of searching ISAT tables residing on other MPI nodes for a sufficiently accurate solution (searching an ISAT table is an $\mathcal{O}(n_s^2)$ operation because it involves matrix–vector multiplications). As a consequence, however, table replication across the MPI nodes is likely to occur.

### 2.4. Coupling parallel `dflame` and `x2f_mpi`

Using parallel `dflame` and the parallel `x2f_mpi` and ISAT libraries involves combining software based on two fundamentally different parallel paradigms: shared memory parallelism implemented using OpenMP (`dflame`) and distributed memory parallelism implemented with MPI. Since the primary objective of the present work is to focus on `dflame` performance enhancements provided by `x2f_mpi` and ISAT, a hybrid programming approach is used to combine the codes. This approach emphasizes minimizing code rewriting and further algorithm development while leveraging the performance gains of running parallel code. Alternatively, `dflame` could have been rewritten using MPI or ISAT rewritten using OpenMP. Either of these approaches involves considerable resources and is outside the scope of the present work.

The hybrid strategy runs the nonreacting portion of `dflame` on a single master MPI node, which also serves as the master OpenMP processor. Therefore, following MPI and ISAT initializations, the `dflame` code runs unaltered on the master MPI node, which spawns OpenMP processes until the reaction substep. That is, the following tasks are executed: `dflame` is initialized, input files are read, time advancement is started, and the first nonreacting RKC substep and Strang subsplitting substep are executed in parallel using OpenMP (the OpenMP processes are spawned from the master MPI node/OpenMP processor). Meanwhile, the remaining nonmaster MPI nodes rest at idle. To execute the reaction substep, `x2f_mpi` is called by all MPI nodes. However, only the master node has nontrivial compositions: the remaining MPI nodes have no compositions to process because `dflame` runs only on the master MPI node (in addition to the OpenMP processor). All compositions on the master node are then randomly distributed among all MPI nodes for processing. Once processed using the local ISAT table, all compositions are returned to the master MPI node and the reaction substep is complete. Following the reaction substep, the `dflame` code resumes running on the single master MPI node (which, again, is also the master OpenMP processor) using OpenMP: the second nonreacting substep is performed, the momentum equations are solved, the corrector phase of the algorithm is executed, and the time step is completed. This process is repeated until the user-specified number of time steps has been executed and the simulation is complete.

The numbers of OpenMP and MPI processors do not have to be equal and the optimal combination to minimize wall clock time may change as the simulation progresses (e.g., after the ISAT table is constructed and populated). The present work does not address the issue of dynamic processor allocation and instead relies on fixed and equal numbers of OpenMP processors and MPI nodes.

## 3. Results

To evaluate the computational performance of the scheme in the context of an unsteady flow, we consider a two-dimensional laminar flame strained and distorted by a counterrotating vortex pair. The premixed methane–air flame burns at atmospheric pressure into a rich ($\Phi = 1.2$) 25% $N_2$-diluted mixture at room temperature. The vorticity field corresponding to the initial vortex is a second-order Gaussian as in [12]. Therefore, the initial condition is a superposition of the velocity field induced by a periodic row of vortex pairs and the temperature, density, and mass fraction distributions corresponding to a premixed flame (see Fig. 1). This flame-flow configuration has been studied extensively in [6–8]: here we focus on performance enhancements due to ISAT.

The simulations are performed on a $0.4 \times 1.6$ cm rectangular domain discretized using $256 \times 1024$ equally spaced mesh points. To decrease CPU time, only one vortex is simulated and periodic and symmetric boundary conditions are applied in the horizontal direction. In the vertical direction, outflow boundary conditions are imposed. As time elapses, the premixed flame propagates downward while the counterrotating vortex pair moves upward.

Chemical kinetics are modeled using GRIMech3.0 [5] with 53 species and 325 reactions. As in [2], the dynamic viscosity, thermal conductivity, and diffusion coefficients are precomputed using Chemkin [13, 14] and tabulated as functions of temperature. Then, transport quantities used in the calculations are computed from table entries using linear interpolation. This approach reduces the amount of CPU time spent computing transport properties (in comparison to, for example, mixture averaged transport) while maintain-

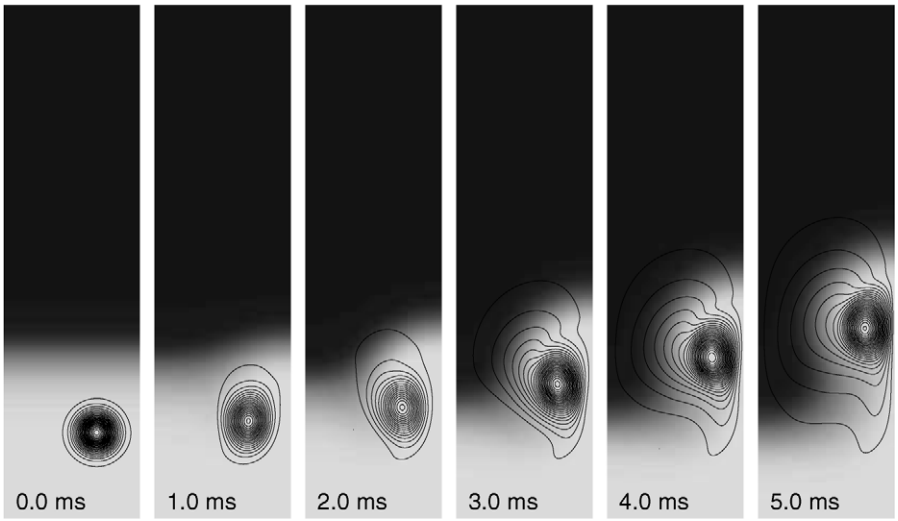


Fig. 1. Two-dimensional flame–flow interaction showing contours of vorticity superimposed on a gray-scale representation of temperature.

ing the spatial and temporal convergence rates of the scheme [12]. As a result, the largest fraction of the overall CPU time is spent computing the reaction substep. It should be noted that use of transport models that are more expensive computationally (e.g., mixture-averaged transport) will reduce the fraction of the overall CPU time spent in the reaction substep. Consequently, less computational speed-up is expected.

To start the calculations, a freely propagating premixed methane–air flame with $\Phi = 1.2$ and the above indicated chemical mechanism is first computed in one spatial dimension using Chemkin. The steady-state solution is then interpolated onto a uniform one-dimensional grid and used as the initial condition for a one-dimensional `dflame` calculation. One-dimensional `dflame` is then run to relax the flame onto the `dflame` mesh (which is different from the Chemkin mesh) and to propagate the flame within the computational domain to the desired location. The flame is then replicated in the horizontal direction. The resulting configuration corresponds to a downward propagating flame in two spatial dimensions. We then specify the strength and location of the vortex and combine the resulting velocity field with the existing temperature, density, and species mass fraction data from the premixed flame. The result is the two-dimensional configuration in the left-most plot in Fig. 1. This configuration is used to initialize the calculations.

As in [4], we set $S = 8$ and $L = 4$, corresponding to the number of RKC stages in the convection–diffusion substep and momentum equation update, respectively. We use a fixed time step of $\Delta t \approx 700$ ns. Due to the computational expense of running the two-dimensional simulations, a parametric study of the impact of the ISAT error tolerance on the accuracy of the solution is not possible. Instead, we apply the results from the detailed error analysis that were performed in the one-dimensional case [2] and set $\varepsilon_{\text{tol}} = 1 \times 10^{-7}$. As demonstrated in [2], this value of $\varepsilon_{\text{tol}}$ limits ISAT errors (in reference to DVODE solutions obtained from the original scheme) to less than 1% for the one-dimensional premixed flame test case. The ISAT ODE absolute and relative error tolerances are $\varepsilon_{\text{abs}} = \varepsilon_{\text{rel}} = 10^{-10}$, as in [2]. The parameters used to specify the strength, location, and size of the counter-rotating vortex are identical to those used in [4].

All calculations are performed on an SGI Origin 2000 with R10000 processors running at 195 MHz. In all cases involving ISAT, the number of OpenMP processors specified is 15 and the number of MPI nodes is 15. For comparison purposes, `dflame` is run using DVODE [15] to compute the reaction substep as done in the original algorithm [4]. In these cases, 15 OpenMP processors are specified along with 1 MPI node.

### 3.1. Qualitative results

Evolution of the two-dimensional flame-flow configuration is demonstrated by examining vorticity contours of the counterrotating vortex and temperature contours of the flame. Fig. 1 shows the time evolution of the flame/vortex interaction. The solid lines indicate vorticity contours. The gray-scale shading indicates the temperature field: dark/light shading corresponds to high/low temperature. As can be seen, the flame propagates downward into the reactants while

the vortex propagates upward by its self-induced velocity. The results observed here are consistent with those obtained in [4].

### 3.2. Original dflame performance

To evaluate the performance gain of the ISAT scheme in relation to the original dflame algorithm, it is necessary to run dflame without ISAT for the flame/vortex problem and obtain representative CPU timings. When this is done, the chemistry substeps are computed using DVODE [15] and Strang subsplitting is not required. To compute the solutions to the systems of stiff, nonlinear equations in the reaction substeps, DVODE must compute Jacobians. In the present dflame implementation, these Jacobians are computed in one of two ways: by finite differences (FD) or via a hard-coded subroutine that computes the Jacobian analytically. The latter method is dependent on the chemical mechanism. Both methods preserve the scheme's second-order temporal accuracy, but the computational expenses are different. Here we focus on DVODE calculations made using FD Jacobians. Although more expensive computationally than computing Jacobians analytically using the hard-coded subroutine (as demonstrated below), use of FD Jacobians represents a more appropriate performance comparison for the ISAT scheme: ISAT must also compute Jacobians and we do not have such an analytical Jacobian subroutine to use. Further discussion of the performance of the DVODE scheme using analytical Jacobians is reserved for Section 4.4. All DVODE calculations are performed with absolute and relative error tolerances $\varepsilon_{\text{abs}} = \varepsilon_{\text{rel}} = 10^{-6}$. These are the same tolerances used in [4].

The amount of CPU time that is required to execute one dflame time step is dependent, in part, on the fraction of the computational domain occupied by the flame. That is, as time evolves and a larger fraction of the computational domain is occupied by the flame and combustion products, the CPU time required by DVODE to integrate the reaction substep increases. In fact, the implicit chemistry integration step implemented in DVODE takes the most time for mesh cells in the primary flame reaction zone, an intermediate time for those in the products region, and the least time for those in the reactants. This phenomenon was observed in the context of a one-dimensional premixed methane–air flame in [2] due to the increasing size of the products region as the flame propagates into the reactants, but is more acute in two-dimensions because of the increased area covered by the contorted primary flame reaction zone. Therefore, computations performed near $t = 0.0$ ms (the leftmost plot in Fig. 1) consume less CPU time per time step than computations performed near $t = 5.0$ ms (the rightmost plot in Fig. 1).

One method of quantifying the performance enhancement due to ISAT is by measuring the speed-up factor,

$$\text{speed-up factor} \equiv \frac{\text{wall clock time per time step using DVODE scheme}}{\text{wall clock time per time step using ISAT scheme}}, \tag{1}$$

as done in [10]. To measure the speed-up due to ISAT, therefore, it is necessary to run dflame using the DVODE algorithm for the same number of time steps as executed for the ISAT version of dflame. This process is expensive computationally and is not feasible with the present computational resources. Instead, we quantify the speed-up in two ways. First, the original dflame code is run for approximately 1150 time steps ($t \approx 0.82$ ms) and then stopped. During this time, there is minimal interaction between the flame front and the counterrotating vortex. The dflame wall clock times per step obtained from this run (approximately 2100 μs per grid point on average) are then used to estimate the speed-up due to ISAT. Note, however, that this approach to measuring speed-up results in speed-up factors lower than those obtained had the original dflame code been run for the same length of simulation time as ISAT-dflame. Therefore, we also quantify the speed-up due to ISAT at longer simulation time by using CPU timing results from a previous flame/vortex calculation by Najm [16]. Here it was found that the overall time step consumes approximately 1.6 times more CPU time at $t \approx 5$ ms than at $t \approx 0.82$ ms. Consequently, we report also the estimated speed-up with this correction factor.

### 3.3. ISAT-dflame performance

Here we explore the performance enhancement due to ISAT. We examine the wall clock time per step for different portions of the algorithm, the speed-up of the reaction substep due to ISAT, the overall code speed-up due to ISAT, and ISAT performance characteristics (e.g., table size, percentage retrieves, CPU time distribution within ISAT).

When using ISAT and x2f_mpi on multiple MPI nodes, ISAT performance data are generated for each node. For the present tests, which use 15 MPI nodes, it is neither feasible nor useful to present all data from all nodes. Therefore, we present representative data from one MPI node with the understanding that different nodes likely have slightly different performance characteristics. The extent to which the performance among the MPI nodes running x2f_mpi differs is

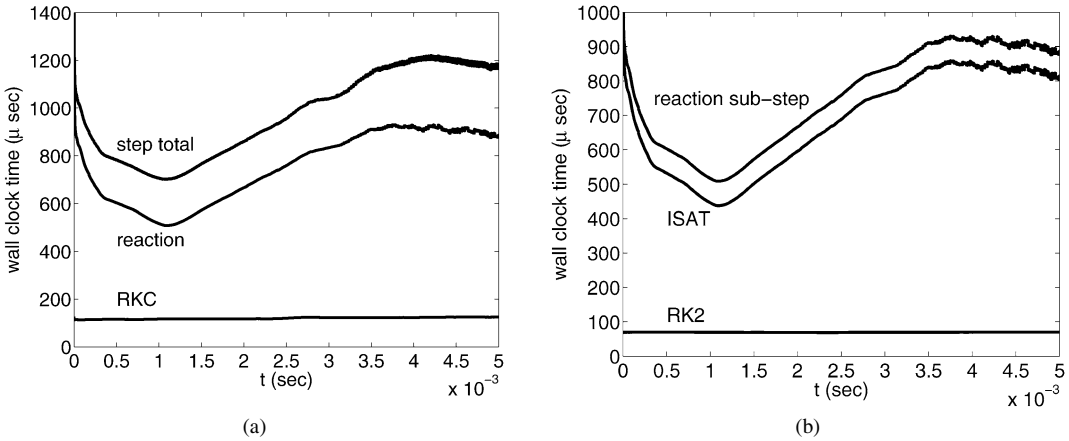


Fig. 2. Wall clock time per grid point per time step of the ISAT scheme: (a) total time step; (b) reaction substep.

determined by the ISAT load balancing. This issue is discussed further in Section 4.3.

The computational performance of the ISAT-`dflame` scheme is evaluated by starting from the same initial conditions as used above (seen pictorially in the leftmost plot in Fig. 1). The simulation is run until $t = 5$ ms, at which time the counterrotating vortex has significantly strained and distorted the premixed flame front.

Fig. 2a plots the wall clock time per grid node per time step for the total time step, the reaction substep, and the RKC diffusion substeps. From the figure we observe that the step-total starts around 1100 μs and decreases rapidly to approximately 725 μs at $t \approx 0.2$ ms. During this time ISAT is constructing and building tables (as discussed below) and there is minimal flame/vortex interaction, as demonstrated in Fig. 1. The time per step then continues to decrease for $0.2 < t \lesssim 1$ ms, but at a slower rate. The flame and vortex are in close proximity during this time. For $1.1 \lesssim t \lesssim 3.5$ ms, the reaction and step-total wall clock times increase steadily. During this time, there is appreciable interaction between the flame front and the vortex as demonstrated in Fig. 1. This interaction is accompanied by access to regions of composition space not previously accessed in the calculations. Consequently, as discussed below, ISAT must perform computationally expensive table `adds` and `grows` leading to an increase in the total wall clock time per step. In particular, from $t \approx 1.1$ to $t \approx 3.5$ ms, the number of ISAT `adds` and `grows` increases by approximately 800% and 325%, respectively. In addition, the time required to perform each `add` and `grow` increases by approximately 70% and 50%, respectively. These increases in CPU times are due to the increased area occupied by the flame. Hence, the large increase in the number of ISAT `adds` and `grows` combined with the increased CPU time required per ISAT `add` and `grow` are the dominant reasons behind the rise in wall clock time from $t \approx 1.1$ to $t \approx 3.5$ ms. For $3.5 \lesssim t \lesssim 4.25$ ms, the wall clock time spent in the reaction substep remains nearly constant. For $t \gtrsim 4.25$ ms, there is a steady decrease in the reaction and wall clock times per step. As discussed below, this result is due to an increase in the fraction of ISAT queries satisfied by `primary` and `secondary retrieves`.

Fig. 2a also shows that the largest fraction of the wall clock time per step is consumed in the reaction substep: the RKC diffusion substeps consume the majority of the remaining time. Further, the trends in the total step CPU time discussed above follow closely those of the reaction substep. Changes in the overall CPU time per step are due to changes in the CPU time required to compute the reaction substep. The rest of the scheme is explicit and therefore its performance is insensitive to the motion of the flame in the computational domain and the fraction of the domain occupied by the flame. Throughout the calculations performed here, the RKC time per step remains nearly constant.

Fig. 2b shows the wall clock time distribution of the reaction substep: the reaction substep curve is identical to that shown in Fig. 2a and represents the wall clock time per grid node per time step spent in the reaction substep. Also shown is the time spent in ISAT and the time spent performing the RK2 subsplitting steps. From the figure we observe that the majority of the reaction substep time is spent in ISAT: the remaining time is spent in the RK2 substeps. The trends in the wall clock time variations of the reaction substep closely follow those of the ISAT wall clock time, while the RK2 time remains constant throughout, as expected from an explicit scheme. Variations in the reaction substep wall clock time are due to variations in the time required by ISAT to compute all necessary species compositions. That is, the wall

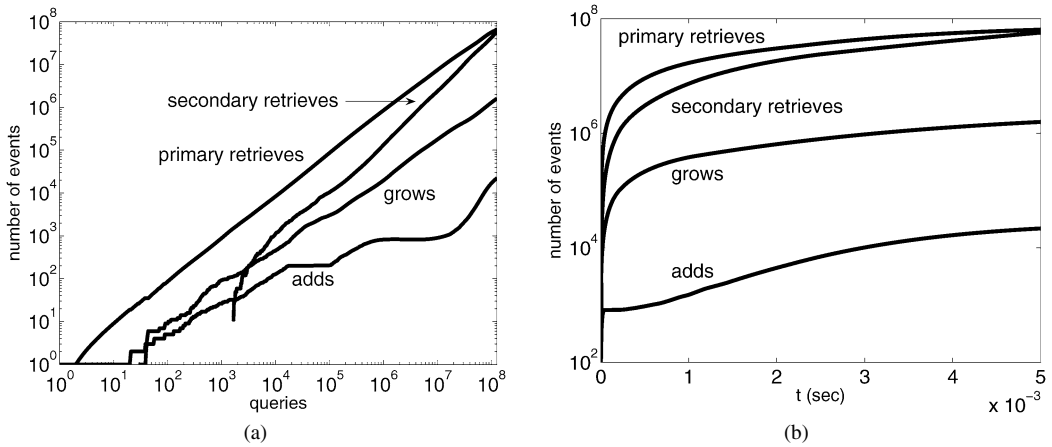


Fig. 3. The number of ISAT `primary retrieves`, `secondary retrieves`, `grows`, and `adds` as functions of (a) the number of ISAT queries and (b) simulation time.

clock time is impacted greatly by the number of compositions satisfied by ISAT `primary retrieves`, `secondary retrieves`, `grows`, and `adds`.

Figs. 3a and 3b show the number of ISAT events versus ISAT queries ($q$) and simulation time, respectively. We note that the majority (e.g., queries $< 10^7$) of Fig. 3a shows data for ISAT events that occur during the initial start-up time of the simulation (e.g., $t < 0.4$ ms). During this time, ISAT queries are satisfied by a combination of `retrieves`, `grows`, and `adds`: `primary retrieves` clearly satisfy most ISAT queries. We also note that there are no table `adds` for $1.7 \times 10^4 \lesssim q \lesssim 1 \times 10^5$. Hence, the ISAT table is populated during the first time step and these table entries are reused (without further table `additions`) for the first six time steps. Another range of no table `adds` is observed for $8 \times 10^5 \lesssim q \lesssim 8 \times 10^6$ corresponding to $0.03 \lesssim t \lesssim 0.3$ ms. During this time we also note an increase in the fraction of `secondary retrieves` as indicated by an increased slope of the `secondary retrieves` line. For $q \gtrsim 8 \times 10^6$ there is an increase in the number of table `additions` suggesting an increase in CPU time. From Fig. 3a we note that the ISAT table continues to increase in entries for $q \gtrsim 2 \times 10^7$ and contains over 20,000 entries by the time $t \approx 5$ ms. The size of each ISAT table (one per MPI node) is approximately 1.1 GB.

In Fig. 3b, the number of ISAT events is plotted as a function of the simulation time. For $t \gtrsim 0.75$ ms there is a rise in the number of ISAT table `adds`. As the simulation time elapses, the increase in the number of table `grows` and `adds` begins to impact the performance of the scheme as demonstrated in Fig. 2. In addition, there is an increase in the number of `secondary retrieves`, which are more computationally expensive than `primary retrieves`. This also contributes to an increase in the wall clock time per step.

To examine the performance of ISAT on a normalized scale, Fig. 4a plots the fraction of ISAT queries that are satisfied by each ISAT operation as a function of ISAT queries. We observe from the figure that, for the entire simulation, most ISAT queries are satisfied by computationally inexpensive `primary retrieves`. For $q \gtrsim 3 \times 10^5$, however, the fraction of `secondary retrieves` increases while the fraction of `primary retrieves` decreases. Meanwhile, the fraction of ISAT `grows` decreases steadily. For $2 \times 10^2 \lesssim q \lesssim 1.5 \times 10^7$, the fraction of ISAT `adds` decreases steadily. Consistent with the findings described above, this behavior suggests that existing table entries are being reused and further growth of the table is minimal. For $q \gtrsim 1.5 \times 10^7$, there is an increase in the fraction of ISAT `adds`, indicating table growth. Nonetheless, at $q \approx 5 \times 10^7$, approximately 95% of ISAT queries are satisfied by either `primary` or `secondary retrieves`. When $q \approx 1.1 \times 10^8$, approximately 42% of queries are satisfied by `secondary retrieves` and 44% are satisfied by `primary retrieves`: less than 1.4% are satisfied by `grows`. For $q \gtrsim 1 \times 10^8$, the fraction of ISAT `grows` and `adds` remains nearly constant: the fraction of ISAT `primary` and `secondary retrieves` decreases and increases, respectively.

Complementary to Fig. 4a, Fig. 4b shows the fraction of CPU time spent performing different ISAT operations as a function of the number of ISAT queries. Clearly, throughout the simulation, the bulk of the ISAT CPU time is spent evaluating $f(x)$ which is the reaction mapping (see [3] for a detailed discussion of the reaction mapping). For $q \gtrsim 80$, ISAT `grows` consume the most CPU time among `pri-`

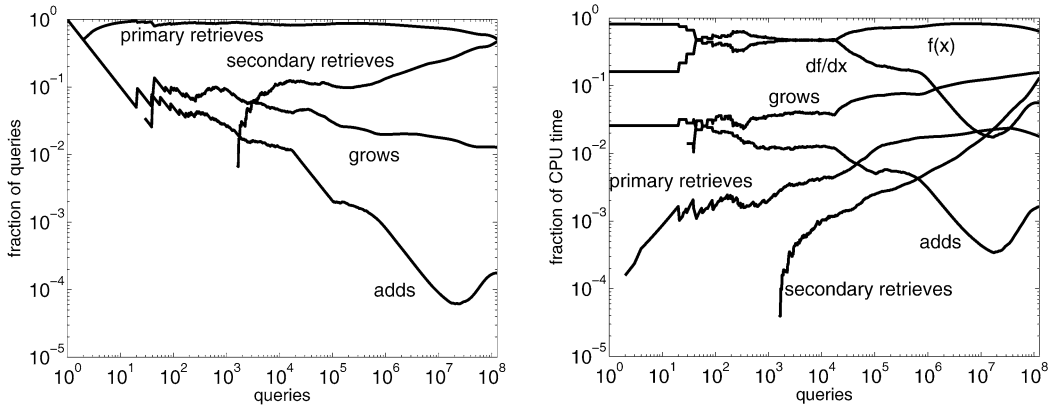


Fig. 4. ISAT performance: (a) fraction of ISAT queries satisfied by each ISAT operation versus number of queries; (b) fraction of CPU time spent performing different operations versus number of queries.

mary retrieves, secondary retrieves, grows, and adds. For $q \lesssim 1.5 \times 10^7$, primary retrieves consume a greater fraction of the CPU time than secondary retrieves. When $q \approx 1.5 \times 10^7$, the CPU time of primary and secondary retrieves are approximately equal. For $q \gtrsim 1.5 \times 10^7$, secondary retrieves consume a greater fraction of the CPU time than primary retrieves. This explains, in part, the overall rise in the CPU time that is spent in ISAT for $t \gtrsim 1$ ms and observed in Fig. 2b. The fraction of CPU time spent performing ISAT table adds decreases or remains constant for $q \lesssim 1.5 \times 10^7$. For $q \gtrsim 1.5 \times 10^7$, there is a rise in the fraction of CPU time spent adding: this corresponds to an increase in the number of table adds (as demonstrated in Fig. 4a). The fraction of CPU time spent evaluating the reaction mapping decreases steadily: at $q \approx 1 \times 10^7$ and $q \approx 1.1 \times 10^8$, $f(x)$ consumes approximately 80% and 65% of the ISAT CPU time, respectively. During this same period of time, the fraction of CPU time spent performing secondary retrieves increases from approximately 1.75% to 12%. Among the ISAT operations, ISAT grows consume the largest fraction of CPU time: at $q \approx 1.1 \times 10^8$, the fraction of time spent growing and performing secondary retrieves is comparable. We also note from Fig. 4b that the fraction of CPU time spent computing $\mathrm{d}f/\mathrm{d}x$, the gradient of the reaction mapping, is qualitatively similar to the time spent performing ISAT table adds: $\mathrm{d}f/\mathrm{d}x$ is computed only for table adds.

To characterize the computational savings due to ISAT, Fig. 5 shows the speed-up (defined by Eq. (1)) as a function of simulation time. We note that the speed-up of the reaction substep is greater than the speed-up of the total time step. As discussed above,

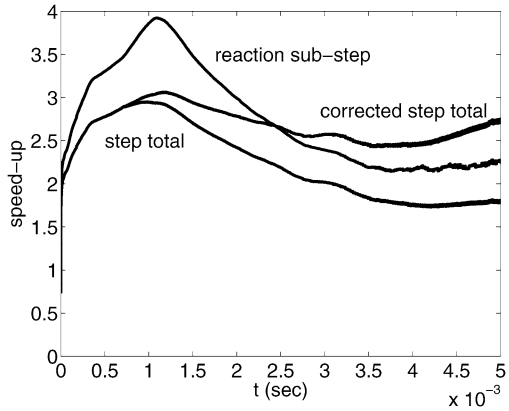


Fig. 5. Speed-up of the total time step and the reaction substep for the ISAT-dflame scheme. The corrected step total speed-up (see Section 3.2) is also shown.

this is because the performance enhancement provided by ISAT only affects the reaction substep: CPU time per step spent outside the reaction substep is not reduced by ISAT. Toward the end of the simulation (i.e., $t \gtrsim 4$ ms), the overall time step speed-up factor is approximately 1.75. Fig. 5 also shows the corrected step total speed-up factor (see Section 3.2) as a function of the simulation time. Here again we note the initial rise in speed-up factor for $t \lesssim 1$ ms. Following a peak speed-up of approximately 3, the corrected speed-up factor declines but less so than the step total speed-up that is not corrected. This difference is due to the fact that the corrected speed-up accounts for the average dflame reaction substep slow-down during the interaction of the flame and the vortex. Consequently, a realistic estimation of the speed-up factor due to ISAT is approximately 2.5–3.

## 4. Discussion

### 4.1. Evaluating trade-offs: accuracy, efficiency, and memory

The use of ISAT introduces error into reacting flow calculations: there is error due to the use of ISAT table entries via linear approximation. This error can, however, be controlled by adjusting the user-specified ISAT error tolerance, $\varepsilon_{\text{tol}}$, as demonstrated in [2]. In addition to changing the level of solution accuracy, $\varepsilon_{\text{tol}}$ also changes the ISAT table size required to obtain the specified level of solution accuracy. That is, as $\varepsilon_{\text{tol}}$ decreases, the size of the ISAT table that is required increases. Consequently, more computer memory is required to store the ISAT table. The quantitative relationship between $\varepsilon_{\text{tol}}$ and ISAT table size depends on a number of factors including: number of species, number of reactions, and region of composition space accessed.

Adjustments to $\varepsilon_{\text{tol}}$ also affect the performance of ISAT. As $\varepsilon_{\text{tol}}$ increases, ISAT performance improves due to the relaxed error criterion, which allows for more computationally inexpensive ISAT table `retrieves`. Further, relatively large values of $\varepsilon_{\text{tol}}$ lead to ISAT tables which are relatively small and require few time steps to build and populate. Consequently, the performance benefit of a tabulation approach is realized with relatively few time steps. The extent to which the performance is enhanced by the use of large $\varepsilon_{\text{tol}}$ is problem dependent.

As described, there are three primary trade-offs to consider when using ISAT: accuracy, efficiency, and memory. The present work demonstrates that, while introducing an acceptable level of error and requiring a modest amount of computer memory, using ISAT can reduce significantly the wall clock time required by multidimensional unsteady reacting flow simulations with detailed chemistry. Further performance enhancement can be achieved by, for example, relaxing ISAT accuracy requirements.

### 4.2. ISAT message-passing strategies

ISAT performance is examined in the context of one message-passing strategy: grid node compositions are randomly distributed among all of the MPI nodes. The results above, which indicate that most of the reaction substep CPU time is spent in ISAT, suggest that improved ISAT message-passing strategies could impact the overall performance of the scheme. It should be remembered, however, that `x2f_mpi` was developed in the context of PDF methods, which are implemented using a distributed memory approach to parallelism. In this regime, each processor has its own set of particle compositions which must be processed by ISAT: only those compositions which cannot be processed locally are message passed. This regime is quite different than the present where, essentially, all particle compositions reside on a single processor and are message passed for ISAT processing (i.e., all compositions except those processed by the master MPI node are message-passed).

To enhance ISAT performance gains by reducing message passing, the `dflame` code could be parallelized using MPI rather than OpenMP. In this case, each MPI node has a particular region of the computational domain (i.e., a fixed set of grid points) for which it is responsible. During the reaction substep, each node would then use the local ISAT table to process only those grid points which reside on the local MPI node. Grid points which contain chemical compositions for which a sufficiently accurate ISAT table entry does not exist are then message-passed using the `x2f_mpi` library as done in PDF calculations. Those compositions processed locally are, unlike the present use of `x2f_mpi`, not message passed to another MPI node. Hence, the amount of message passing required to perform one time step is reduced significantly, which could lead to a significant reduction in wall clock time.

### 4.3. Load balancing

As time evolves and the flame is strained and distorted by the counterrotating vortex, a larger fraction of the computational domain becomes occupied by the flame front. When this occurs, it is necessary to distribute evenly among all MPI nodes the computational load of computing the reaction substeps; otherwise, load imbalance may occur which increases the wall clock time. To help achieve load balancing, we randomly distribute compositions among all MPI nodes. In this section we examine the resulting load balance that is achieved by this message-passing strategy.

To examine quantitatively the balance of load, we introduce the expression

$$P_i = 100 \times \frac{T_i}{\max(T_i)}, \quad i = 0, \ldots, 14, \tag{2}$$

where $T_i$ is the cumulative CPU time that node $i$ spends in ISAT. Hence, good load balancing is indicated by $P_i \approx 100$ for all MPI nodes. Conversely, load imbalance is indicated by different $P_i$ for each MPI node.

Fig. 6 shows $P_i$ for each MPI node and indicates that all MPI nodes spend within 97.5% of the same amount of time in ISAT. This result indicates excellent ISAT load balancing and contributes to the computational performance enhancement provided by ISAT.

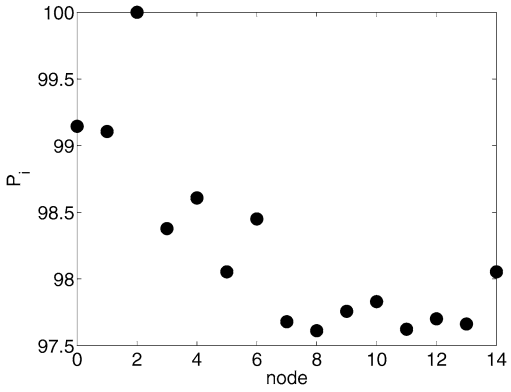


Fig. 6. $P_i$ (defined Eq. (2)) for each MPI node.

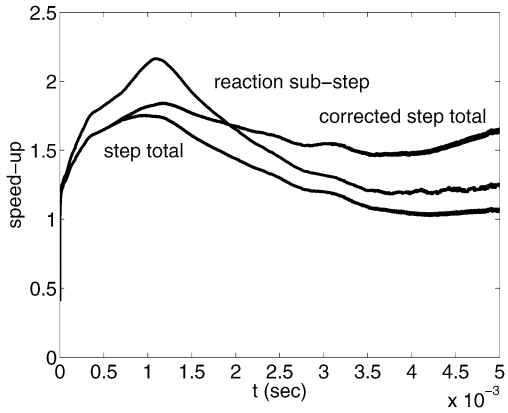


Fig. 7. Speed-up of the total time step and the reaction substep using analytical Jacobian calculation for DVODE. The corrected step total speed-up (see Section 3.2) is also shown.

### 4.4. Analytical Jacobians

As described above, a subroutine is available to compute analytically the Jacobians required by the DVODE integrator. Use of this routine eliminates the need to compute Jacobians using finite differences and reduces the wall clock time of the DVODE reaction substep by approximately 45%. Consequently, the speed-up of the reaction substep due to ISAT is reduced.

Fig. 7 plots the speed-up of the ISAT scheme as a function of simulation time. Here, the ISAT speed-up is in comparison with the original DVODE scheme using the hard-coded subroutine to compute analytically the Jacobians required by DVODE. The corrected speed-up factor is also shown, which indicates a speed-up due to ISAT of approximately 1.5–1.75.

### 4.5. Maximum ISAT speed-up

Periodically throughout the calculations, the ISAT tables are written to disk. In the event of a `dflame` restart, the resulting ISAT files may then be read in to avoid having to reconstruct the ISAT tables from scratch. To investigate further the computational cost of creating and filling the ISAT tables, we rerun the calculations performed in Section 3.3 for $0 < t \leqslant 5$ ms. Rather than starting with empty ISAT tables, however, we use the ISAT tables from the previous calculations. Consequently, nearly all ISAT table queries are satisfied by computationally inexpensive ISAT table `retrieves` and we are able to estimate the maximum speed-up that is achievable by ISAT (maximum speed-up is achieved when all ISAT queries are satisfied by `primary retrieves`).

Shown in Fig. 8a is the wall clock time per grid point per time step for the total time step, the reaction substep, and the RKC diffusion substep. From the figure we observe that for $t \lesssim 2.25$ ms the reaction time is less than the RKC time. For $2.25 \lesssim t \lesssim 3.75$ ms, the reaction time increases nearly linearly as does the total step time. For $3.75 \lesssim t \leqslant 5$ ms, the reaction and total step times are nearly constant. The reason for these results is discussed below.

The wall clock time distribution for the reaction substep is shown in Fig. 8b as a function of simulation time. As evident, for $t \lesssim 1.75$ ms the time spent in ISAT is less than the time spent performing the RK2 substeps. In this regime, use of a numerical scheme that is more efficient than RK2 to compute the substeps would appreciably decrease the overall CPU time. For $1.75 \lesssim t \lesssim 3.75$ ms, the time spent in ISAT increases steadily. This is due to an increase in the fraction of ISAT queries that are satisfied by `secondary retrieves` (no additional `adds` or `grows` are performed) and a corresponding decrease in the fraction of ISAT `primary retrieves`. ISAT `secondary retrieves` are performed due to tabulation error (and the propagation thereof) and the parallel implementation. That is, because compositions are distributed randomly among the processors, satisfaction of all queries to the ISAT table via `primary retrieves` does not occur. For $3.75 \lesssim t \leqslant 5$ ms, the time spent in ISAT remains nearly constant. During this time the fraction of ISAT queries satisfied by `primary` and `secondary retrieves` remains nearly constant. Performing additional runs using the same ISAT tables will further develop the tables and likely lead to additional reduction in CPU time.

Fig. 9 shows the reaction substep and overall time step speed-up due to ISAT; the corrected step total speed-up factor is also shown. From the figure we note that the use of existing ISAT tables increases significantly the reaction substep speed-up. In particular, the reaction speed-up factor is approximately 10–15 over the original DVODE scheme. Similarly, the un-

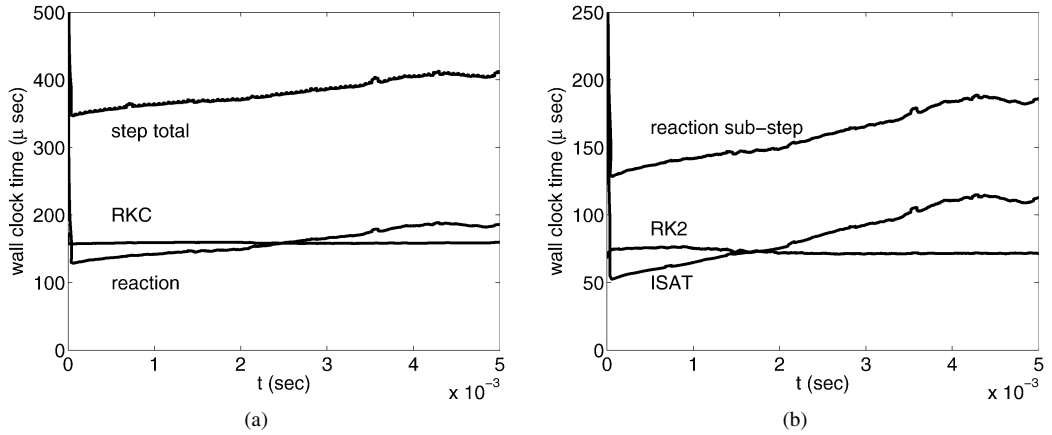


Fig. 8. Wall clock time per grid point per time step of the ISAT scheme. ISAT tables from previous calculations are read in. (a) Total time step; (b) reaction substep.

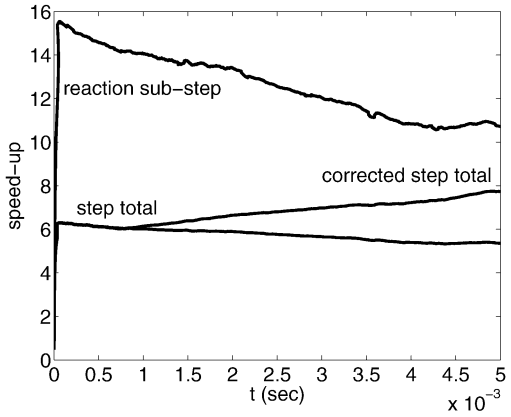


Fig. 9. Speed-up of the total time step and the reaction substep using prebuilt ISAT tables. The corrected step total speed-up (see Section 3.2) is also shown.

corrected total time step speed-up factor observed here is appreciably greater than in previous calculations which required construction of the ISAT table. The uncorrected speed-up factor of the total time step for the present calculations is approximately 5–6. The corrected speed-up factor results indicate a qualitatively different result than the uncorrected results. When we account for the increased CPU time required by `dflame` when the flame and the vortex interact, the ISAT speed-up factor increases with simulation time. This result is due to the fact that, for $t \gtrsim 1$ ms, the DVODE slow-down is greater than the ISAT slow-down. Hence there is a net increase in the speed-up due to ISAT. In this case, the speed-up due to ISAT is approximately 6–8.

The reuse of ISAT tables discussed here is designed to assess the maximum potential speed-up due to ISAT for the flame/vortex problem being considered. The results above further demonstrate that significant computational resources are required to construct and build the ISAT table. As a result, table reuse is a valuable feature that ISAT allows and can facilitate the study of complex unsteady reacting flows. If, for example, one were to study the impact of initial or boundary conditions on the present flow configuration, then ISAT tables could be reused potentially saving a significant amount of CPU time. Further, calculations performed with different transport models (e.g., mixture averaged diffusion), which do not affect the reaction substep could also benefit from ISAT table reuse. But, modifications to the problem may change the portion of composition space which is accessed during the computation. Consequently, further expansion and modification of the ISAT table (via `grows` and `adds`) will likely be required, thereby reducing the speed-up. Nonetheless, the potential for speed-up is great due to the reuse of existing ISAT tables, especially for flows that exhibit a large initial transient followed by more stationary behavior.

### *4.6. Comparison of one- and two-dimensional results*

In [2], a one-dimensional premixed methane–air flame was computed using a serial version of the ISAT-`dflame` algorithm. The version of ISAT which was used in that work only performed `primary retrieves`: no `secondary retrieves` were used. In both the one- and two-dimensional problems, the majority of ISAT queries are satisfied by `retrieves`, followed by `grows`, followed by `adds`. In the case of the one-dimensional flame, the number of ISAT `adds` and `grows` levels off after the table is constructed and populated. This result is due to the relatively small portion of composition space that is accessed in the one-dimensional computation. In a reference frame that moves with the flame, the com-

putation is a steady-state problem; therefore, reuse of entries in the ISAT table is large once the entries of the ISAT table cover the low-dimensional manifold in composition space. In contrast, the two-dimensional flame/vortex problem is unsteady. Consequently, the ISAT table continues to expand (via `adds` and `grows`) as the computation proceeds.

The speed-ups due to ISAT that are observed for the one- and two-dimensional computations are significantly different. For the one-dimensional case, the overall speed-up factor increases steadily and then levels off at approximately 7–8. For the two-dimensional problem, there is greater variation in the speed-up due to the unsteadiness of the problem. The overall speed-up that is observed for the flame/vortex problem is, however, appreciably lower than that observed in the one-dimensional flame computations. The reasons for these differences are primarily two-fold. First, because the flame/vortex problem is unsteady, ISAT must tabulate a larger portion of the composition space than for the one-dimensional flame problem. Consequently, there are more table `adds` and `grows` (as discussed above) that are expensive computationally. Second, because of the parallel implementation of the ISAT algorithm (as discussed above), each processor has its own ISAT table. Consequently, different processors may compute and tabulate the same regions of composition space thereby increasing the simulation time that is required before table `retrieves` dominate the number of ISAT events. In contrast, the one-dimensional computation was a serial calculation so that no issues with table overlap were present. In addition, there are communication costs associated with the parallel implementation of ISAT that were not present in the one-dimensional implementation of the algorithm.

### 4.7. Implications for turbulent flames

The speed-up results observed here for the two-dimensional computations demonstrate the potential to accelerate the computation of turbulent reacting flows. In particular, when performing three-dimensional direct numerical simulation, there are frequently large regions of the computational domain with little flame activity. In these regions, queries to the ISAT table are likely to be satisfied by table `retrieves` thereby reducing CPU time significantly. In addition, there are more grid points in a three-dimensional domain than in a two-dimensional domain that stand to benefit from using ISAT.

In the simulation of a turbulent reacting flow, the region of composition space that is accessed is likely to be larger than the portion accessed in the present work. Consequently, the ISAT tables will be larger, require more memory to store, and will take longer to construct. During the start-up time when the majority of the queries to the ISAT table are satisfied by `grows` and `adds`, little or no speed-up is anticipated; in fact, the use of ISAT may slow the computation (as seen in the present work). Nonetheless, if the simulation proceeds to the point where table `retrieves` dominate, the potential speed-up is appreciable and may be greater than that observed in the present work.

## 5. Conclusions

We have demonstrated the use of ISAT and the Strang-based subsplitting scheme of Singer et al. [2] in the context of a two-dimensional unsteady reacting flow with detailed chemistry. To accelerate computations and thereby decrease CPU time, parallel versions of `dflame` and ISAT were outlined and used: a hybrid programming approach was used to couple the two codes. The scheme was applied to the case of a laminar flame front interacting with a counterrotating vortex pair: this configuration has been studied extensively both experimentally and numerically [6–8] and the results obtained here are consistent with those found in [8]. ISAT performance was examined and compared to that of the original DVODE-based scheme. Here it was found that the ISAT scheme provides an overall code speed-up of approximately 2.5–3.

Further discussion also addressed alternative ISAT message-passing strategies. Here it was noted that the number of ISAT compositions requiring message passing could be reduced by parallelizing the `dflame` algorithm using MPI. This reduction in message passing could lead to further performance gains. In addition, ISAT performance can also be enhanced by using a message-passing scheme tailored for laminar flames (rather than the random distribution approach used here which was designed for PDF methods). Such an approach can reduce both communication and ISAT table-building times. Load balancing among the MPI nodes was also examined for the ISAT scheme. For the present computations, we found that each MPI node spends within 97.5% of the same amount of time in ISAT; this result suggests excellent ISAT load balancing with the present message-passing strategy. The speed-up due to ISAT in comparison to the original `dflame` algorithm with a subroutine to compute analytically the Jacobians required by the DVODE integrator was also examined. Here we found that ISAT provided an overall time speed-up of approximately 1.5–1.75. Finally, ISAT calculations were performed using existing ISAT tables (constructed from previous runs). In this case, the overall time-step speed-up factor due to ISAT was approximately 6–8.

## Acknowledgments

M.A.S. acknowledges support from a NASA Graduate Student Researchers Program Fellowship and the Langley Research Center. S.B.P. acknowledges support from U.S. Department of Energy Grant DE-FG02-90ER14128. H.N.N. has been supported by the U.S. Department of Energy (DOE) and by the DOE Office of Basic Energy Sciences (BES), Division of Chemical Sciences, Geosciences and Biosciences. Computations were performed at Sandia National Laboratories. Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94-AL85000.

## References

[1] D.A. Schwer, P. Lu, W.H. Green Jr., Combust. Flame 133 (4) (2003) 451–465.
[2] M.A. Singer, S.B. Pope, H.N. Najm, Combust. Theory Modelling 10 (2006) 199–217.
[3] S.B. Pope, Combust. Theory Modelling 1 (1997) 41–63.
[4] H.N. Najm, O.M. Knio, J. Sci. Comput. 25 (2005) 263–287.
[5] G.P. Smith, D.M. Golden, M. Frenklach, N.W. Moriarty, B. Eiteneer, M. Goldenberg, C.T. Bowman, R.K Hanson, S. Song, W.C. Gardiner Jr., V.V. Lissianski, Z. Qin, available at: http://www.me.berkeley.edu/gri_mech/.
[6] H.N. Najm, P.S. Wyckoff, Combust. Flame 110 (1–2) (1997) 92–112.
[7] H.N. Najm, P.H. Paul, C.J. Mueller, P.S. Wyckoff, Combust. Flame 113 (3) (1998) 312–332.
[8] H.N. Najm, O.M. Knio, P.H. Paul, P.S. Wyckoff, Combust. Theory Modelling 3 (1999) 709–726.
[9] J.G. Verwer, Appl. Numer. Math. 22 (1–3) (1996) 359–379.
[10] M.A. Singer, S.B. Pope, Combust. Theory Modelling 8 (2004) 361–383.
[11] S.B. Pope, S.R. Lantz, `x2f_mpi`: Software for the efficient load distribution of function evaluations in a multi-processor MPI environment, in preparation.
[12] O.M. Knio, H.N. Najm, P.S. Wyckoff, J. Comput. Phys. 154 (2) (1999) 428–467.
[13] R.J. Kee, J.F. Grcar, M.D. Smooke, J.A. Miller, Sandia Report SAND85-8240, Sandia National Laboratories, 1985.
[14] R.J. Kee, F.M. Rupley, J.A. Miller, Sandia Report SAND89-8009, Sandia National Laboratories, 1989.
[15] P.N. Brown, G.D. Byrne, A.C. Hindmarsh, SIAM J. Sci. Stat. Comput. 10 (5) (1989) 1038–1051.
[16] H.N. Najm, personal communication, 2005.