Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Computationally efficient implementation of combustion chemistry in parallel PDF calculations

Liuyan Lu [a,*], Steven R. Lantz [b], Zhuyin Ren [a], Stephen B. Pope [a]

[a] Sibley School of Mechanical and Aerospace Engineering, Cornell University, Upson Hall 245, Ithaca, NY 14853, USA
[b] Center for Advanced Computing, Cornell University, Ithaca, NY 14853, USA

ARTICLE INFO

Article history:
Received 23 November 2008
Accepted 20 April 2009
Available online 6 May 2009

PACS:
07.05.Mh
46.15.–x
47.11.–j

Keywords:
ISAT
Combustion chemistry
Parallel calculation
Distribution strategy
Load balance

ABSTRACT

In parallel calculations of combustion processes with realistic chemistry, the serial *in situ* adaptive tabulation (ISAT) algorithm [S.B. Pope, Computationally efficient implementation of combustion chemistry using *in situ* adaptive tabulation, Combustion Theory and Modelling, 1 (1997) 41–63; L. Lu, S.B. Pope, An improved algorithm for *in situ* adaptive tabulation, Journal of Computational Physics 228 (2009) 361–386] substantially speeds up the chemistry calculations on each processor. To improve the parallel efficiency of large ensembles of such calculations in parallel computations, in this work, the ISAT algorithm is extended to the multi-processor environment, with the aim of minimizing the wall clock time required for the whole ensemble. Parallel ISAT strategies are developed by combining the existing serial ISAT algorithm with different distribution strategies, namely purely local processing (PLP), uniformly random distribution (URAN), and preferential distribution (PREF). The distribution strategies enable the queued load redistribution of chemistry calculations among processors using message passing. They are implemented in the software *x2f_mpi*, which is a Fortran 95 library for facilitating many parallel evaluations of a general vector function. The relative performance of the parallel ISAT strategies is investigated in different computational regimes via the PDF calculations of multiple partially stirred reactors burning methane/air mixtures. The results show that the performance of ISAT with a fixed distribution strategy strongly depends on certain computational regimes, based on how much memory is available and how much overlap exists between tabulated information on different processors. No one fixed strategy consistently achieves good performance in all the regimes. Therefore, an adaptive distribution strategy, which blends PLP, URAN and PREF, is devised and implemented. It yields consistently good performance in all regimes. In the adaptive parallel ISAT strategy, the type and extent of redistribution is determined "on the fly" based on the prediction of future simulation time. Compared to the PLP/ISAT strategy where chemistry calculations are essentially serial, a speed-up factor of up to 30 is achieved. The study also demonstrates that the adaptive strategy has acceptable parallel scalability.

© 2009 Elsevier Inc. All rights reserved.

1. Introduction

Numerical calculations of reactive flows with realistic chemical kinetics are computationally expensive. At the same time, they are becoming increasingly important both in understanding the physical processes and in the design and development of practical systems, such as engines and combustors. The computational difficulty is caused by the large number of chemical

* Corresponding author.
E-mail address: lu.liuyan@gmail.com (L. Lu).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.04.037

Nomenclature

Roman symbols

$\mathbf{A}$	mapping gradient matrix with components $A_{ij} \equiv \partial f_i / \partial x_j$
A^*	critical number of ISAT table entries
A	maximum number of table entries per processor allowed
a	number of table entries in a serial calculation
a_i	total number of tabulated table entries in all processors in group i in the adaptive strategy
a_L^*	maximum number of table entries per processor allowed on the Lth pairing stage in the adaptive strategy
$\mathbf{f}(\mathbf{x})$	function of $\mathbf{x}$ of dimension n_f
$\mathbf{f}^l$	linear approximation to $\mathbf{f}(\mathbf{x})$
g	number of processors in each group in the adaptive strategy
$\mathbf{L}$	overlap matrix with components $L_{ij} \equiv P_{ij}/P_{ii}$
M_s	total number of pairing stages, $M_s = log_2(N_p)$
M_r	number of partially stirred reactors in the simulation
n_s	number of species
n_ϕ	dimension of composition $\boldsymbol{\phi}$
n_x	dimension of $\mathbf{x}$
n_f	dimension of $\mathbf{f}$
N	number of particles in a partially stirred reactor
N_{Fij}	average number of particles per processor requiring function evaluation when groups i and j are paired
N_i	average number of particles per processor in group i in the adaptive strategy
N_g	number of groups in the simulation
N_{ij}	average number of particles processed on each processor when groups i and j are paired
$N_{i,\alpha}$	number of particles on processor α in group i
N_p	total number of processors in a simulation
P_{ij}	probability of a particle composition from group i being able to be retrieved from the ISAT table(s) in group j in the adaptive strategy
$\widehat{P}_{\alpha\beta}$	probability of a particle composition from processor α being able to be retrieved from the ISAT table on processor β in the adaptive strategy
p_A	probability of a query resulting in an add
$p_A(q,a)$	probability of add on the qth query when there are a table entries
$p_A(a)$	probability of add when there are a table entries
p_{Ai}	probability of a query resulting in an add for group i
p_D	probability of a query resulting in a discarded evaluation (DE)
p_F	probability of a query resulting in a function evaluation, $p_F = p_A + p_G + p_D$
p_{Fi}	probability of a query resulting in a function evaluation for group i
p_{fd}	threshold value of the frequency of the add and grow events below which an ISAT table is considered fully developed (see Eq. (C.1))
p_G	probability of a query resulting in a grow
p_R	probability of a query resulting in a retrieve ($p_R = 1 - p_F$)
Q_α	number of queries performed on processor α
q, Q	number of queries performed
$q_f(A)$	query on which ISAT table becomes full (i.e., ISAT fills A table entries)
$q(a)$	number of queries resulting in a table entries
$\mathbf{R}(\boldsymbol{\phi})$	reaction mapping
r	exponent in the observed power law Eq. (A.4)
$\mathbf{S}$	chemical source term Eq. (3)
s	ratio between A and A^*, i.e., $s = A/A^*$
T	average wall clock time spent in reaction fractional step for one block of particles
T_i'	estimated wall clock time spent in reaction fractional step for one block of particles for group i
T_{ij}'	estimated wall clock time spent in reaction fractional step for one block of particles for the hypothetical pairing between group i and $j (i \neq j)$
T_{np}'	estimated wall clock time spent in reaction fractional step for one block of particles with no pairing performed
T_p'	estimated wall clock time spent in reaction fractional step for one block of particles with the optimal pairing
t_F	average CPU time for a function evaluation
$t_{F,w}$	average wall clock time for a function evaluation
$t_{Fi,w}$	average wall clock time for a function evaluation on group i
t_Q	average CPU time for a query
$t_{Q,w}$	average wall clock time for a query
t_R	average CPU time for a retrieve

$t_{R,w}$	average wall clock time for a retrieve
$t_{Ri,w}$	average wall clock time for a retrieve on group i
$t_{Rij,w}$	average wall clock time for a retrieve when groups i and j are paired
$t_{Rj\to i,w}$	average wall clock time per particle for particles from group j attempting to retrieve from ISAT tables on group i
$\mathbf{x}$	vector of dimension n_x
Greek symbols	
Δt	time step in reaction fractional step
ε_{tol}	user-specified error tolerance for ISAT
ε	incurred local error in ISAT Eq. (6)
τ_{mix}	specified mixing time scale in a PaSR
τ_{res}	specified residence time scale in a PaSR
τ_{pair}	specified pairing time scale in a PaSR
$\boldsymbol{\phi}$	particle composition
Calligraphic	
$\mathcal{A}$	add region
$\mathcal{D}$	particle composition distribution
$\mathcal{G}$	grow region
$\mathcal{P}_k$	kth feasible pairing
$\mathcal{R}$	retrieve region
Superscripts	
$'$	estimated quantity
Abbreviations	
ISAT	*in situ* adaptive tabulation
PLP	purely local processing
PREF	preferential distribution
URAN	uniform random distribution
PaSR	partially stirred reactor
ODE	ordinary differential equation
EOA	ellipsoid of accuracy
ROA	region of accuracy

species and the wide range of time scales involved in chemical kinetics. A realistic description of combustion chemistry for hydrocarbon fuels typically involves tens to thousands of chemical species [3,4], and the time scales usually range from 10^{-9} s to over 1 s [5,6]. The above considerations motivate the well-recognized need for the development of methodologies that radically decrease the computational burden imposed by the direct use of realistic chemistry in reactive flow calculations. Among such methodologies are storage/retrieval approaches including structured look-up tabulation [7], repro-modelling [8], artificial neural networks (ANN) [9,10], *in situ* adaptive tabulation (ISAT) [1,2], piecewise reusable implementation of solution mapping (PRISM) [11,12], and high dimension model representations (HDMR) [13].

The ISAT [1] algorithm is currently particularly fruitful and it has been widely used to incorporate reduced or detailed chemical mechanisms in probability density function (PDF) [14] calculations of turbulent nonpremixed flames [15–20]. While the computational efficiency of the ISAT algorithm is greatest in statistically stationary reactive flows, such as the Sandia turbulent jet flames where a speed-up factor of 100–1000 is achieved, ISAT has also been applied to the calculation of transient processes such as combustion in IC engines [17] where a speed-up factor of more than 10 is reported. Recently, ISAT has been incorporated in the LES/FDF approach [21,22] that offers the benefits of both large eddy simulation (LES) to treat the turbulent flow and the PDF approach to treat turbulence–chemistry interactions. The ISAT algorithm has also been applied to incorporate detailed chemical kinetics in the direct numerical simulation (DNS) of reactive flow [23,24]. Besides the wide applications in the field of combustion, the applications of ISAT in other areas have been reported in [25–27].

When ISAT is employed to speed up chemistry calculations in computational fluid dynamics (CFD), which can be direct numerical simulation (DNS), large eddy simulation (LES) or a probability density function (PDF) method, a reaction fractional step is used to separate the chemical reactions from other processes such as convection and molecular diffusion. The task performed by ISAT in the reaction fractional step is to determine the thermo-chemical compositions after a computational time step (either variable or constant) due to chemical reactions. In the context of PDF methods [14], where the system within the solution domain is represented by a large number of computational particles, the task for ISAT in the reaction step is to determine the particle compositions after reaction. We call a particle "resolved" when its composition after reaction has been obtained. By tabulating useful information in binary trees called ISAT tables and reusing it, ISAT can substantially reduce the number of chemical kinetic calculations required and therefore provide significant speed-up for chemistry calculations.

Despite the seemingly unending progress in microprocessor performance indicated by Moore's law, large-scale computations of turbulent reactive flows with realistic chemistry demand that we pursue the additional factors of tens, hundreds, or thousands in total performance which may be obtained by harnessing a multitude of processors for a single calculation. For example, the terascale direct numerical simulations of three-dimensional turbulent temporally evolving plane CO/H_2 jet flames with an 11 species skeletal mechanism reported by Hawkes et al. [28] are performed on massively parallel processors.

One common type of platform to perform large-scale computations is a distributed memory system using some implementation of the message passing interface (MPI) to perform message passing between processors. The computation is most often parallelized using domain decomposition on the coordinate grid that represents the spatial configuration of the flow: the whole computational domain is decomposed into sub-domains and each processor performs the computation for one sub-domain.

When ISAT is employed to speed up the chemistry calculations in parallel PDF computations, each processor typically maintains its own ISAT table. During the reaction fractional step, each processor has an ensemble of particles whose compositions at the end of the reaction step need to be determined. However the original ISAT algorithm by Pope [1] is serial in the following sense: during the reaction fractional step each processor performs its own chemistry calculations without message passing or load redistribution. Due to the nonuniform intensity of chemical reactions or nonuniform distribution of computational particles among the sub-domains, there is usually significant load imbalance in the chemistry calculations. For example, some sub-domains may have intense reaction activity, so the chemistry calculations are more challenging and require more computational resources; whereas others may be essentially inert (e.g., pure air or pure fuel) and the chemistry calculations are trivial. Previous calculations using spatial domain decomposition [21,22] show that even for a simple two-dimensional, spatially developing, reacting, plane mixing layer, it is hard to achieve good load balance in chemistry calculations if ISAT is used without any message passing. Hence even though ISAT substantially speeds up the chemistry calculations on each processor, the overall load imbalance in the chemistry calculations among the processors severely affects the parallel efficiency and provides further opportunities to develop algorithms for more efficient chemistry calculations. However, it should be noted that (as described in Section 4) ISAT poses a non-standard load balancing problem that cannot be solved readily by any common load balancing technique or software. Moreover load balance is not truly the right target for optimization: wall clock time is. As revealed in previous studies [21,22], the optimal algorithm – the one that minimizes the wall clock time for the chemistry calculations – may not necessarily give the best load balance.

The above observation motivates the development of parallel ISAT strategies with the objective of minimizing the wall clock time taken to complete a reaction fractional step on all processors. There are several viable approaches for developing parallel ISAT strategies such as parallelizing the current serial ISAT algorithm or developing distribution strategies to be used in combination with the serial ISAT algorithm. The approach taken in this study is the latter, and it works as follows. In the parallel calculations of reactive flows, each processor maintains its own ISAT table. During the reaction fractional step, the particles on one processor may be distributed to one or more other processors, and be resolved by the ISAT tables there. Particles are distributed by message passing before and after ISAT, not within ISAT. Different distribution strategies have been developed and implemented in software *x2f_mpi* [22], which is a Fortran 95 library developed for facilitating many parallel evaluations of a general vector function. The strategies discussed here are called purely local processing (PLP), uniformly random distribution (URAN), and preferential distribution (PREF). For PLP, there is no message passing during the chemistry calculations, and particles on one processor are locally processed via the local ISAT table. For URAN, the particles in a group of processors are randomly distributed uniformly among all the processors in the group using message passing. For PREF, the particles have preference to some processors: for example, particles can only be passed to those processors that they have visited during a previous step, or have not already visited during the current step.

The distribution strategies developed for parallel ISAT can be applied in either a fixed or adaptive manner. For parallel ISAT with a fixed distribution strategy, the particular strategy (e.g., PLP, URAN, or PREF) is specified by the user before a simulation and does not change. For the adaptive strategy, the type of distribution strategy can be changed on the fly based on a comparison of predictions of future performance. In this study, the performance of the various fixed and adaptive parallel ISAT strategies is investigated in parallel PDF calculations of the oxidation of methane/air mixtures in multiple partially stirred reactors (PaSR) on a distributed memory system.

The outline of the paper is as follows. In Section 2, the test case of partially stirred reactors (PaSR) burning methane/air mixtures is described. In Section 3, the ISAT algorithm is briefly reviewed, and serial ISAT performance is characterized in terms of regimes related to table size. The parallel calculation of reactive flows using ISAT is outlined in Section 4, and the different distribution strategies in the software *x2f_mpi* are detailed. In Section 5, parallel ISAT with various distribution strategies is described and demonstrated, and the idea of a multi-stage process is introduced. In Section 6, the methodology, the algorithm, and the performance of the adaptive strategy are presented. In Section 7, the relative performance of the parallel ISAT strategies in different computational regimes is investigated. The effect of the number of processors on the parallel ISAT performance is discussed in Section 8. Section 9 discusses the implications of the results and outlines possible directions for future work, and conclusions are drawn in Section 10.

2. Partially stirred reactor (PaSR)

The partially stirred reactor (PaSR) was used previously by Pope [1] to investigate the performance of ISAT in serial computations. It has the advantage of simplicity in terms of controlling the distribution of particle compositions, and therefore allows the performance of ISAT to be explored in different computational regimes (as demonstrated below). Moreover, the amount of computational work spent outside of ISAT (i.e., the reaction fractional step) in a PaSR calculation is negligible, which provides a more efficient use of computational resources for the study of ISAT performance. Due to its simplicity, the PaSR has been widely used to investigate combustion models and numerical algorithms [1,29–32]. It is similar to a single grid cell embedded in a large PDF computation of turbulent combustion.

In the stochastic simulation of a PaSR based on Monte Carlo methods, at time t, the reactor consists of an even number of particles, N, with the ith particle having composition $\boldsymbol{\phi}^i(t)$. The composition is taken to be the species specific moles (mass fractions over molecular weights) and the sensible enthalpy of the mixture. The particles are arranged in pairs: particles 1 and 2, 3 and $4,\ldots,N-1$ and N are partners. With Δt being the specified time step, at the discrete times $k\Delta t$ (k integer), events occur corresponding to outflow, inflow and pairing, which can cause $\boldsymbol{\phi}^i(t)$ to change discontinuously. Between these discrete times, the composition evolves by a mixing fractional step and a reaction fractional step. The mixing fractional step consists of pairs (p and q, say) evolving by

$$\frac{d\boldsymbol{\phi}^p}{dt} = -(\boldsymbol{\phi}^p - \boldsymbol{\phi}^q)/\tau_{mix}, \tag{1}$$

$$\frac{d\boldsymbol{\phi}^q}{dt} = -(\boldsymbol{\phi}^q - \boldsymbol{\phi}^p)/\tau_{mix}, \tag{2}$$

where τ_{mix} is a specified mixing time scale. In the reaction fractional step, each particle evolves by the reaction equation

$$\frac{d\boldsymbol{\phi}^i}{dt} = \mathbf{S}(\boldsymbol{\phi}^i), \tag{3}$$

where $\mathbf{S}$ is the rate of change of composition given by the chemical kinetics.

With τ_{res} being the specified residence time, at the discrete times $k\Delta t$, outflow and inflow consist of selecting $\frac{1}{2}N\Delta t/\tau_{res}$ pairs at random and replacing their compositions with inflow compositions, which are drawn from a specified distribution. With τ_{pair} being the specified pairing time scale, $\frac{1}{2}N\Delta t/\tau_{pair}$ pairs of particles (other than the inflowing particles) are randomly selected for pairing. Then these particles and the inflowing particles are randomly shuffled so that (most likely) they change partners. Between the discrete times, i.e., over a time step Δt, the composition evolves by one mixing step of Δt, followed by one reaction step of Δt.

The fuel considered in this study is methane. The pressure is atmospheric throughout. The specified time scales are $\tau_{res} = 1\times 10^{-2}$ s, $\tau_{mix} = 1\times 10^{-3}$ s, $\tau_{pair} = 1\times 10^{-3}$ s, and the time step is constant with $\Delta t = 4\times 10^{-5}$ s.

In the serial PaSR calculations which are used to characterize the serial ISAT performance below, we consider both a 16-species skeletal mechanism [33] and the GRI3.0 mechanism [3] (without nitrogen chemistry) consisting of 36 species. There are three inflowing streams: air (79% N_2, 21% O_2 by volume) at 300 K; methane at 300 K; and a pilot stream consisting of the adiabatic equilibrium products of a stoichiometric fuel/air mixture at a temperature of 2600 K, corresponding to an unburnt temperature of 1113 K. The mass flow rates of these streams are in the ratio 0.85:0.1:0.05. The number of particles in the reactor, N, is 100. Initially, all particle compositions are set to be the pilot stream composition. In order to explore ISAT performance in the statistically stationary state, a statistically stationary solution is first obtained, then long-run simulations are performed starting from this solution.

In this study, to investigate the parallel ISAT performance, the above serial PaSR is naturally extended to the multi-processor environment through the creation of a multiple PaSR test case. In the parallel simulation of the multiple PaSR, M_r independent reactors are distributed among the N_p processors with each processor having M_r/N_p reactor(s), with M_r being an integer multiple of N_p. For simplicity, all the cases considered below have the number of reactors equal to the number of processors, i.e., $M_r = N_p$.

All the parallel calculations performed employ the GRI3.0 mechanism without nitrogen chemistry. Each reactor has three inflowing streams: air, fuel, and pilot with the mass flow rates being in the ratio 0.85:0.1:0.05. For one class of test cases considered below, all the reactors are statistically identical. There are three inflowing streams: air (79% N_2, 21% O_2 by volume) at 300 K; methane at 300 K; and a pilot stream consisting of the adiabatic equilibrium products of a stoichiometric fuel/air mixture at a temperature of 2600 K, corresponding to an unburnt temperature of 1113 K. For another class of cases presented below, to make the composition distributions disjoint among the processors, by design, the above three inflowing streams on each reactor are diluted by a specified amount of Argon, i.e., on the αth reactor (with $\alpha = 1,2,\cdots,N_p$), each stream is diluted so that the fraction of Ar (by mass) is $(\alpha-1)/(\alpha-1+721/50)$. In other words, on the αth reactor, the air stream is diluted with Ar such that the ratio (by volume) of N_2, O_2 and Ar is 79:21:5$(\alpha-1)$, and the fuel and pilot streams are correspondingly modified such that the fractions of Ar (by mass) in these two streams are the same as that of the air stream. Also while keeping the unburnt temperature of the pilot stream unchanged (i.e., 1113 K), the temperatures of the inflowing air and fuel stream on different processors change linearly, i.e., on the αth reactor, the temperatures of the fuel and air streams are specified at $(300 + 50\times(\alpha-1))$ K. (These settings are chosen so that all the PaSR reactors yield burning solutions.) For the case with a uniform number of queries, each reactor has 5000 particles. For the nonuniform cases, the reactor on the first processor has $N_p \times 5000$ particles while the other reactors have 5000 particles each.

All the results from multiple PaSR test cases presented below are from long-run simulations restarting from pre-obtained statistically stationary solutions (with empty ISAT tables). The ISAT error tolerance ε_{tol} and the maximum number of entries allowed A are given below for each case presented.

3. *In situ* adaptive tabulation (ISAT) for combustion chemistry

In this section, we first outline the essential concepts in the original ISAT algorithm [1]. The recent augmentations made in the new implementation of ISAT, denoted as ISAT5, are detailed in [2]. Then we characterize the performance of ISAT (i.e., ISAT5) in the serial PDF calculation of the combustion process in a statistically stationary PaSR.

3.1. ISAT concepts

The *in situ* adaptive tabulation algorithm (ISAT) introduced by Pope [1] is a storage and retrieval method. Briefly stated, ISAT is used to tabulate a function $\mathbf{f}(\mathbf{x})$, where $\mathbf{f}$ and $\mathbf{x}$ are vectors of length n_f and n_x, respectively.

Consider the application of ISAT for chemistry calculations in PDF calculations of the combustion process in an isobaric PaSR. At time t, the thermo-chemical composition of the ith particle is represented by the $n_\phi = n_s + 1$ variables $\boldsymbol{\phi}^i(t)$, where n_s is the number of chemical species. The evolution of particle composition due to reaction is treated in a separate fractional step, where the particle composition evolves (at fixed pressure and enthalpy) according to Eq. (3), i.e.,

$$\frac{d\boldsymbol{\phi}(t)}{dt} = \mathbf{S}(\boldsymbol{\phi}(t)). \tag{4}$$

The task in the reaction fractional step is to determine the reaction mapping $\mathbf{R}(\boldsymbol{\phi}^0) \equiv \boldsymbol{\phi}(t_0 + \Delta t)$, which is the solution to Eq. (4) after a time step Δt from the initial condition $\boldsymbol{\phi}^0 = \boldsymbol{\phi}(t_0)$ at time t_0. Here, for simplicity, Δt is taken to be a constant. Hence in the context of numerical calculations of the reaction fractional step using ISAT, $\mathbf{x}$ is the particle composition prior to the reaction fractional step, $\boldsymbol{\phi}^0$, and $\mathbf{f}$ is the particle composition after the reaction fractional step, i.e., the reaction mapping $\mathbf{R}(\boldsymbol{\phi}^0) = \boldsymbol{\phi}(t_0 + \Delta t)$. Thus n_x and n_f are both vectors of length $n_s + 1$. A *function evaluation* obtains the reaction mapping by integrating Eq. (4).

ISAT uses the ODE solver DDASAC [34] to integrate Eq. (4) and stores the relevant information in a binary tree, with each termination node (or leaf) representing a record consisting of (among other information) the tabulation point $\mathbf{x}$, the reaction mapping $\mathbf{f}$, and the mapping gradient matrix $\mathbf{A}$ (or sensitivity matrix), defined as $A_{ij} = \partial f_i / \partial x_j$. For a given query composition $\mathbf{x}^q$ close to a tabulated point $\mathbf{x}$, from the tabulated quantities at $\mathbf{x}$, a linear approximation to $\mathbf{f}(\mathbf{x}^q)$, denoted as $\mathbf{f}^l(\mathbf{x}^q)$, can be obtained, i.e.,

$$\mathbf{f}^l(\mathbf{x}^q) \equiv \mathbf{f}(\mathbf{x}) + \mathbf{A}(\mathbf{x})(\mathbf{x}^q - \mathbf{x}). \tag{5}$$

The incurred local error is simply defined as the scaled difference between the exact mapping and the linear approximation, i.e.,

$$\varepsilon = |\mathbf{B}(\mathbf{f}(\mathbf{x}^q) - \mathbf{f}^l(\mathbf{x}^q))|, \tag{6}$$

where $\mathbf{B}$ is a scaling matrix [1].

In addition to $\mathbf{x}$, $\mathbf{f}$, and matrix $\mathbf{A}$, at each leaf, an ellipsoid of accuracy (EOA) is also stored. An EOA is a hyperellipsoid used to approximate the region of accuracy (ROA), which is defined to be the connected region in composition space containing $\mathbf{x}$ in which the incurred local error ε (defined by Eq. (6)) does not exceed the user-specified error tolerance ε_{tol}.

For a given query $\mathbf{x}^q$, ISAT traverses the tree until a leaf representing some $\mathbf{x}$ is reached. This value of $\mathbf{x}$ is intended to be close to $\mathbf{x}^q$. One of the following events is invoked to obtain an approximation to the corresponding function $\mathbf{f}(\mathbf{x}^q)$.

- Retrieve. If the query point falls within the ellipsoid of accuracy (EOA) of $\mathbf{x}$, a linear approximation to $\mathbf{f}(\mathbf{x}^q)$ is returned through Eq. (5). This outcome is denoted as a retrieve.
- Grow. Otherwise (i.e., $\mathbf{x}^q$ is outside of the EOA), a function evaluation is performed to determine $\mathbf{f}(\mathbf{x}^q)$, which is exact and returned. Moreover the error in the linear approximation is measured through Eq. (6). If the computed error is within the user-specified tolerance ε_{tol}, the EOA of the leaf node $\mathbf{x}$ is grown to include the query point. This outcome is called a grow.
- Add. In the previous (grow) process, if the computed error is greater than ε_{tol} and the table is not full (i.e., the ISAT table has not reached the allowed memory limit), a new entry associated with $\mathbf{x}^q$ is added to the ISAT table. This is called an add.
- Discarded evaluation. If, however, the computed error is larger than ε_{tol} and the table is full, then $\mathbf{f}(\mathbf{x}^q)$ obtained by the function evaluation is returned without further action. (Hence the function evaluation has no effect on the ISAT table.) This outcome is called a discarded evaluation.

(It is worth emphasizing that the above are the basic ISAT processes in the original ISAT algorithm [1]. The up-to-date version of ISAT is detailed in [2]; the further innovations in that version do not affect our present discussion of the parallel algorithm.)

Notice that one event of grow, add, or discarded evaluation involves one and only one function evaluation. The average CPU time to perform a function evaluation (denoted as t_F) is typically several orders of magnitude larger than the average CPU time to perform a retrieve (denoted as t_R). ISAT speeds up the chemistry calculations by obtaining the reaction mapping using retrieve whenever possible. Moreover, in a large-scale calculation, the grow and add events are in general likely only during the table building period, which typically accounts for only a small fraction of the whole simulation.

3.2. Characterization of serial ISAT performance

The parallel adaptive strategy (to be described below) is based on predictions of how well ISAT will perform much later during a given simulation. Hence, in the remainder of Section 3.2 we characterize the performance of serial ISAT to provide a basis for such predictions.

When ISAT is employed for chemistry calculations in simulating reactive flows, there are many factors affecting its performance, e.g.: the stationarity of the simulation; the length of the simulation; the dimensionality of $\mathbf{x}$ and $\mathbf{f}$; the cost of evaluating $\mathbf{f}(\mathbf{x})$; the particular implementation of the ISAT algorithm; the user-specified ISAT error tolerance ε_{tol}; and the user-specified memory allowed for the ISAT table.

An ISAT task is defined by the function $\mathbf{f}(\mathbf{x})$, the total number of queries Q, the error tolerance ε_{tol}, the given implementation of the ISAT algorithm, and the distribution $\mathcal{D}(\mathbf{x})$ from which the ith query $\mathbf{x}_i$ is drawn. In this study, the distribution $\mathcal{D}(\mathbf{x})$ considered is stationary (i.e., independent of i), and the simulation results in a very large number of ISAT queries, Q. We consider the case where the physical memory limits the number of ISAT table entries, A, that can be tabulated. Given an ISAT task, to understand the ISAT performance, it is important to investigate the probabilities of different ISAT events and their dependence on the allowed table entries.

To characterize the ISAT performance, we consider serial PDF calculations of the statistically stationary nonpremixed methane/air combustion in a PaSR. Each is a long-run calculation resulting in a very large number of ISAT queries Q.

3.2.1. Probability of function evaluation after many queries

When ISAT is used to facilitate chemistry calculations, initially the ISAT table is empty. During the calculation, the ISAT table is built and developed through grows and adds. For a given ISAT task, during the calculation, the probabilities of different events depend on the allowed number of table entries, A, and the number of queries performed, q. Let $p_R(q,A), p_G(q,A), p_A(q,A)$ and $p_D(q,A)$ denote the probabilities of retrieve, grow, add, and discarded evaluation on the qth query with the allowed table entries A, respectively. We have

$$p_R(q,A) + p_G(q,A) + p_A(q,A) + p_D(q,A) = 1. \tag{7}$$

In the calculations, these probabilities can be estimated from the recorded ISAT statistics.

Fig. 1 shows the probabilities of different events against the number of queries from a PaSR calculation. In the early stage of the simulation, the number of add and grow events are significant and the sum of their probabilities can be more than 10%. In contrast, in the late stage of the simulation, the probability of add and grow decreases monotonically. Conceptually, the operation of ISAT in the simulation can therefore be thought of in terms of a *building phase*, in which the ISAT table is built and developed by grows and adds; and a *retrieving phase* in which adds and grows are negligible or non-existent, and essentially all queries are resolved by retrieves or discarded evaluations (if the table is full). For the very long-run calculation considered, the cost of the building phase is likely a negligible fraction of the cost of the whole simulation.

Function evaluation is assumed to be very expensive compared to retrieve, so a fundamental quantity in developing an understanding of ISAT performance is the probability of function evaluation $p_F(q,A)$, which is defined to be the sum of the probabilities of grow, add, and discarded evaluation, i.e.,

$$p_F(q,A) \equiv p_G(q,A) + p_A(q,A) + p_D(q,A) = 1 - p_R(q,A). \tag{8}$$

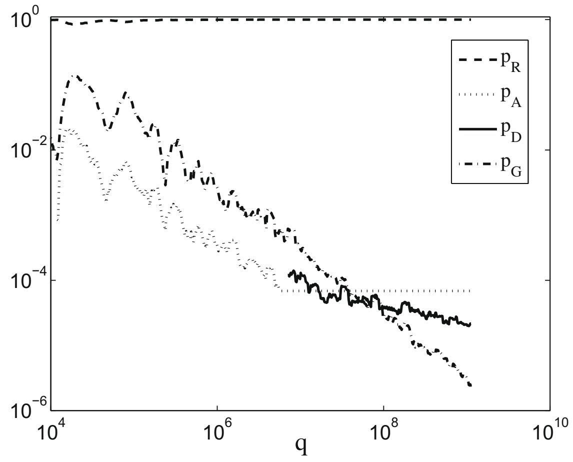


Fig. 1. The probabilities of retrieve p_R, add p_A, discarded evaluation p_D and grow p_G at query q against the number of queries q in the PaSR calculation with the skeletal mechanism, with $\varepsilon_{tol} = 1 \times 10^{-3}$ and $A = 2.0 \times 10^3$. The probability of discarded evaluation is nonzero only after the table is full. The probability of add is zero after the table is full, although it is drawn as a flat line for the sake of illustration. The turning point where the p_A curve becomes flat indicates the point where the ISAT table becomes full.

(Recall that each event of grow, add, or discarded evaluation involves one function evaluation.) Notice that before the ISAT table is full, p_D is zero and hence $p_F = p_A + p_G$; after the table is full, p_A is zero and hence $p_F = p_D + p_G$. Let $q_f(A)$ denote the query on which the ISAT table becomes full. As shown in Appendix A, for a given table size A, the probability of function evaluation p_F after an infinite number of queries $p_F(\infty, A)$ is approximately equal to the probability of add when the table becomes full, $p_A(q_f(A), A)$. (As revealed by the notation, $p_A(q_f(A), A)$ depends only on A.) The empirical relation between $p_A(q_f(A), A)$ and A can be approximated by an inverse power law (see Eq. (A.4)).

3.2.2. Estimate of the average query time t_Q

We are now ready to assemble the above insights regarding ISAT performance into a long-term prediction that is based on current ISAT statistics. For a long-run calculation, the cost of the building phase is in general negligible, and in the *retrieving phase* essentially all queries are resolved either by retrieves or by discarded evaluations. Hence in the retrieving phase, the average CPU time for a query, t_Q, can be well approximated as

$$t_Q = t_R p_R(\infty, A) + t_F p_F(\infty, A) = t_R(1 - p_F(\infty, A)) + t_F p_F(\infty, A) = t_R + p_F(\infty, A)(t_F - t_R), \tag{9}$$

where t_R is the average CPU time to perform a retrieve, and t_F is the average CPU time to perform a function evaluation. On the first two lines of Eq. (9), the first terms on the right hand side are the contributions from retrieve, and the second terms are the contributions from function evaluation. (Recall that $p_F(\infty, A) = p_D(\infty, A)$.) The ideal ISAT performance is attained when $p_F(\infty, A) = 0$, i.e., when essentially all the queries are resolved by retrieves. Under this circumstance, the average time for a query, t_Q, is equal to the retrieve time t_R.

The variable t_R is subject to fluctuations over the course of a calculation because it depends on the configuration of the ISAT table as it develops. But as shown in Fig. 2, to a good approximation, t_R is a constant when the table is fully developed (i.e., after the building phase). The average CPU time for a function evaluation t_F depends solely on the distribution $\mathcal{D}(\mathbf{x})$ from

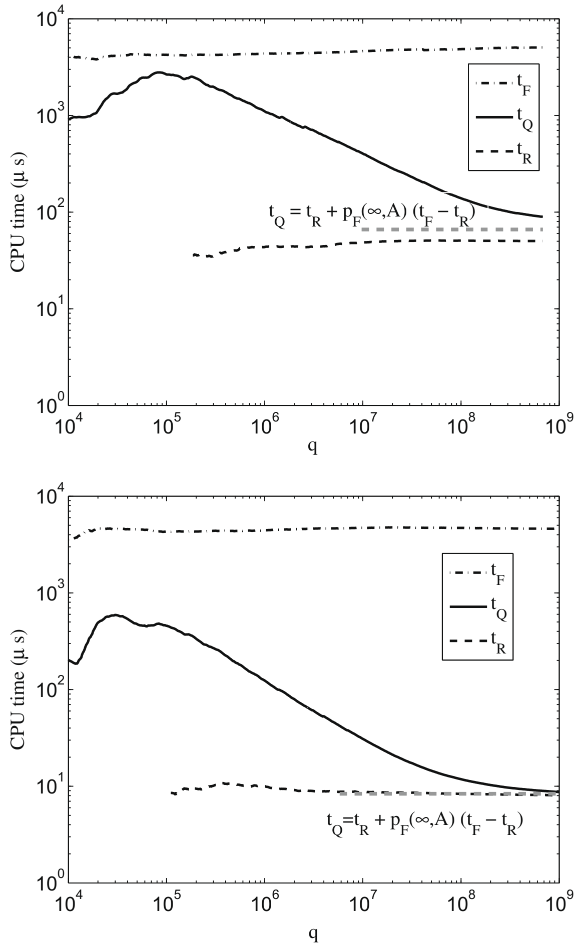


Fig. 2. Average CPU time for a function evaluation t_F, a query t_Q, and a retrieve t_R against the number of queries from the PaSR calculation with the skeletal mechanism. Top plot: $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 6 \times 10^4$; bottom plot: $\varepsilon_{tol} = 1 \times 10^{-3}$ and $A = 2 \times 10^3$. Also shown (as the gray dashed lines) are the predicted query times using Eq. (9). In the prediction, $p_F(\infty, A)$ is estimated using the probability of add when the table becomes full (see Appendix A for more detail).

which $\mathbf{x}$ is drawn. To a good approximation, t_F is a constant along the simulation as shown in Fig. 2. (However the CPU time for a single function evaluation may vary significantly over the distribution $\mathcal{D}(\mathbf{x})$, e.g., by an order of magnitude.) In general, the function evaluation time t_F is much larger than the retrieve time t_R, e.g., by several orders of magnitude.

Fig. 2 shows the average CPU times t_R and t_F against the number of queries for two cases. As may be seen, to a good approximation, both t_R and t_F are constant with $t_R \approx 35$ μs and $t_F \approx 6 \times 10^3$ μs for the first case (top plot of Fig. 2). For the second case (bottom plot of Fig. 2), t_F is the same, but because of the smaller table size, t_R is reduced to less than 10 μs. For these particular cases, t_F is more than two orders of magnitude larger than t_R. Also plotted in the figure are the query times from both the simulation and the prediction (for $q \to \infty$) according to Eq. (9). In the prediction, $p_F(\infty, A)$ is estimated using the probability of add when the table becomes full. For a large number of queries, Eq. (9) provides a reasonable estimate (or at least asymptote) for the average query time.

3.2.3. Supercritical and subcritical ISAT regimes

One final aspect of ISAT performance remains to be discussed, which will turn out to have a significant bearing on why different parallelization strategies are more or less effective in speeding up a given long-run simulation. With Eq. (9), two different computational regimes can be identified, namely a supercritical regime and a subcritical regime. In the supercritical regime, the particles can be almost always successfully retrieved from the ISAT table and the contribution from retrieve to the query time is dominant. In contrast, in the subcritical regime, the contribution from function evaluation is dominant. For a given ISAT task (with a given ε_{tol}), which computational regime a long-run calculation is in depends solely on the allowed table size A. To be more rigorous, we define the critical number of ISAT table entries, A^*, implicitly by

$$p_F(\infty, A^*) = \frac{t_R}{t_F - t_R}. \tag{10}$$

Thus with A^* table entries, retrieves and function evaluations contribute equally to the average query time. Given that $p_F(\infty, A)$ is a monotonically decreasing function of A, there is a unique value of A^* satisfying this equation. With this definition, the average query time can be re-expressed as

$$\frac{t_Q}{t_R} = 1 + \frac{p_F(\infty, A)}{p_F(\infty, A^*)}. \tag{11}$$

Evidently the *storage ratio* $s \equiv A/A^*$ determines the effectiveness of ISAT. In the supercritical regime, defined by $s \geqslant 1$, ISAT is very effective and $t_Q/t_R \leqslant 2$, i.e., within a factor of 2 of the ideal performance. In the subcritical regime, defined by $s < 1$, the time spent on function evaluations is significant and $t_Q \approx p_F(\infty, A)t_F \geqslant 2t_R$.

The above discussion highlights the significance of the allowed table size A to the ISAT performance. An increase in A can effectively move the calculation from the subcritical regime to the supercritical regime, and hence greatly enhance the computational efficiency of the chemistry calculation. Fig. 3 shows the average query time from two PaSR calculations with the same settings except the allowed table size A. As may be seen, with an increase in A from 2×10^4 to 6×10^4, the average query time decreases from about 300 μs to 100 μs, and the calculation shifts from the subcritical regime to the supercritical regime.

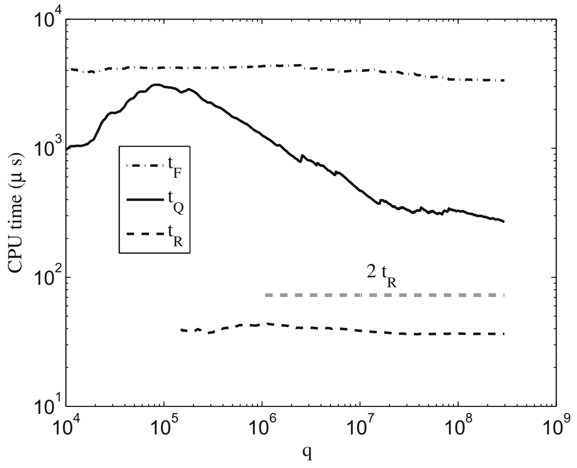
4. Parallel computations of turbulent combustion

In this study, the target platform for performing parallel calculations is a distributed memory system with N_p processors. For CFD of an inhomogeneous reactive flow with domain decomposition, the whole computational domain is decomposed into N_p sub-domains and each processor performs the computation for one sub-domain. In the PaSR tests considered here, each of the N_p processors is assigned its own PaSR. Message passing among the processors is performed using MPI 1.1 [36].

When ISAT is used for the combustion chemistry calculations, each processor has its own ISAT table. The same ISAT error tolerance ε_{tol} and allowed table size A are specified on each of the processors. We consider the case in which the physical memory limits the maximum number of ISAT table entries A that can be tabulated on each processor. During the reaction fractional step, each processor has an ensemble of particles whose compositions after the reaction step need to be determined. In other words, each processor has an ensemble of particles that needs to be resolved. For each processor, the particles originally located on the processor are referred to as local particles. In parallel computations, the following ISAT processes can be invoked to attempt to resolve a particle:

- attempt to retrieve from the local ISAT table,
- attempt to retrieve from the ISAT tables on remote processors,
- function evaluation (through one of the events grow, add or discarded evaluation) on the local processor,
- function evaluation (through one of the events grow, add or discarded evaluation) on a remote processor.

Notice that the processes performed on remote processors incur extra message passing time. The retrieve attempts do not guarantee to resolve a particle, whereas function evaluation does. Another important difference between these different processes is the associated computational cost. The retrieve time may be several orders of magnitude smaller than the function evaluation time.



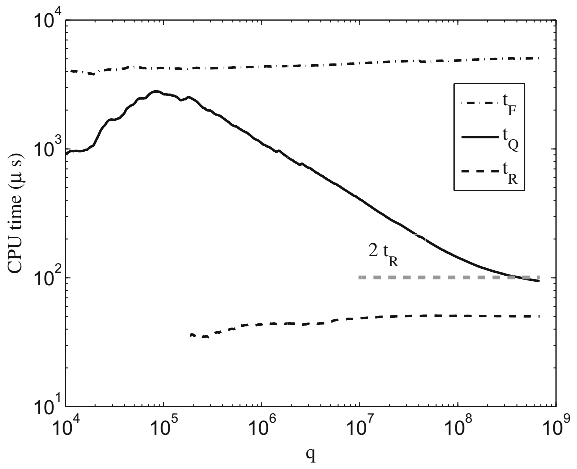


Fig. 3. Average CPU time for a function evaluation t_F, a query t_Q and a retrieve t_R against the number of queries. Top plot: a subcritical ($t_Q > 2t_R$) case with $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 2 \times 10^4$; bottom plot: a supercritical ($t_Q < 2t_R$) case with $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 6 \times 10^4$. Also shown are the gray dashed lines of $2t_R$.

The computational load of chemistry calculation on a processor depends strongly on the number of queries and the composition distribution on the processor. Load imbalance of chemistry calculations can be caused by the nonuniform distributions of queries and compositions among processors. However, it should be noted that ISAT poses a non-standard load balancing problem that cannot be solved readily by common load balancing techniques or software due to the following reasons:

- The unit operation to be performed (i.e., resolution of a query) takes a random, highly-variable amount of CPU time to perform (e.g., by several orders of magnitude).
- The amount of time a query takes is not known *a priori*, and there is no computationally cheap test to determine how much it will cost to resolve the query.
- The amount of time a query takes on a given processor depends on the whole history of previous queries on that processor; as a consequence, a given query can take very different times to resolve on different processors.

It should also be noted that, as stated previously, load balance is not truly the right target for optimization: wall clock time is. The optimal algorithm that minimizes the wall clock time for the chemistry calculations may not necessarily give the best load balance.

4.1. Effects of query and composition distributions

As mentioned, the ISAT performance depends strongly on the number of queries and the composition distributions among different processors. Let Q_α denote the number of queries on processor α. Due to the possible nonuniform distribution of computational particles among the sub-domains, Q_α may vary significantly among processors. Furthermore, due to the possible nonuniform reaction activity among the sub-domains, the composition distribution may also vary significantly from processor to processor. Let $\mathcal{D}_\alpha(\mathbf{x})$ denote the composition distribution on processor α. The two extremes that may arise in a multi-processor calculation are: *coincident query distributions*, in which $\mathcal{D}_\alpha(\mathbf{x})$ is identical for all processors; and *disjoint query distributions*, in which $\mathcal{D}_\beta(\mathbf{x})$ is disjoint from $\mathcal{D}_\alpha(\mathbf{x})$ for all $\alpha \neq \beta$.

The important concept we use to describe the similarities of the composition distribution $\mathcal{D}_\alpha(\mathbf{x})$ among the processors is *query overlap*. Consider a parallel computation that relies on purely local processing to resolve particles using ISAT, i.e., particles are resolved using the local ISAT table without message passing and load redistribution. (This approach is denoted as PLP/ISAT, where PLP stands for "purely local processing".) Let $\widehat{P}_{\alpha\beta}$ denote the probability that a query from processor α could (hypothetically) be retrieved using the ISAT table on processor β. By definition $\widehat{P}_{\alpha\alpha}$ denotes the probability of normal, local retrieval. The *query overlap* in a calculation can be quantified by the *overlap matrix* $\mathbf{L}$ with the component $L_{\alpha\beta}$ defined by

$$L_{\alpha\beta} = \widehat{P}_{\alpha\beta}/\widehat{P}_{\alpha\alpha}, \tag{12}$$

where the summation convention does not apply. In general, when ISAT tables are built using the PLP/ISAT strategy, queries from one processor are far more likely to be retrievable from the local ISAT table than from the ISAT tables on remote processors. Hence it is reasonable to expect $\widehat{P}_{\alpha\beta} \leqslant \widehat{P}_{\alpha\alpha}$ and therefore $0 \leqslant L_{\alpha\beta} \leqslant 1$. For the two extremes, we have $L_{\alpha\beta} = 1$ for coincident query distributions, and $L_{\alpha\beta} = \delta_{\alpha\beta}$ for disjoint query distributions.

Given the above, four extreme computational regimes can be identified based on the composition distributions $\mathcal{D}_\alpha(\mathbf{x})$ and the number of queries Q_α among the processors, namely: coincident and uniform; coincident and nonuniform; disjoint and uniform; and disjoint and nonuniform. As the name indicates, in the coincident and uniform regime, the query distributions among the processors are coincident and the number of queries is uniform among the processors.

The main goal of this study is to explore strategies that will result in good parallel ISAT performance, not just for one of the four extreme computational regimes, but for all of them. For the investigation, we use the multiple PaSR test cases described in Section 2. For the coincident cases, all the reactors have identical inflowing streams; for the disjoint cases presented, each reactor has different inflowing streams by design to make the composition distributions disjoint among the processors. The multiple PaSR test has the advantage of simplicity in terms of controlling the distribution of particle compositions $\mathcal{D}(\mathbf{x})$ and the number of queries on each processor. Therefore it allows one to explore the ISAT performance in the above different computational regimes.

4.2. Software x2f_mpi

To parallelize ISAT, the simplest approach is PLP/ISAT in which particles are resolved using the local ISAT table without message passing and load redistribution. However, this simple PLP/ISAT strategy is not the computationally-optimal strategy for all chemistry calculations. In parallel calculations, even though the straightforward PLP/ISAT strategy substantially speeds up the combustion chemistry calculations on each processor, the parallel computational efficiency of PLP/ISAT can be severely affected by the load imbalance of chemistry calculations caused by the nonuniform distributions of queries and compositions among processors. For example, Fig. 4 shows the wall clock time and CPU time per particle step in the reaction fractional step from a nonuniform coincident PaSR calculation. (For a given processor, the wall clock time and CPU time per particle step are defined as the total wall clock time and the total CPU time on that processor, normalized by the average number of particles on all processors.) As may be seen, for PLP/ISAT, there is significant load imbalance due to the nonuniformity of queries among the processors. For the case considered, the CPU time spent by the first processor, which has the 8

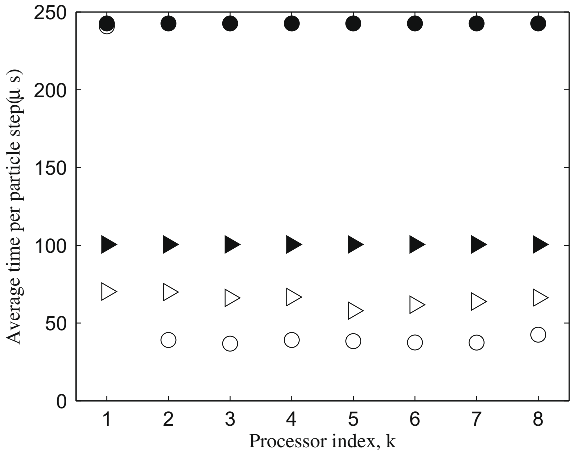


Fig. 4. The wall clock time and CPU time per particle step (in microseconds) for each processor from the nonuniform coincident PaSR calculation with $\varepsilon_{tol} = 5 \times 10^{-4}$ and $A = 1 \times 10^3$. For a given processor, the wall clock time and CPU time per particle step are defined as the total wall clock time and the total CPU time on that processor normalized by the average number of queries on all processors. Solid symbol: wall clock time; open symbol: CPU time. Symbol ○: results from PLP/ISAT; ▷: results from URAN/ISAT. The calculations result in an average of 1.9×10^8 queries per processor.

times the number of queries as the other processors, is about 6 times that on the other processors, and thus the other processors have a significant amount of idle time. One can imagine this imbalance in query distributions being exacerbated by nonuniform composition distributions. Yet even with uniform distributions of queries and compositions (see Fig. 8) and consequently good load balance among processors, the simple PLP/ISAT strategy may still not be the optimal strategy that minimizes the wall clock time for the chemistry calculations.

These observations motivate the development of more sophisticated parallel ISAT strategies to further improve parallel efficiency. The objective is to minimize the wall clock time spent in chemistry calculations. We consider the scenario where the communication (message passing) time per particle t_C is much smaller than the average function evaluation time t_F. If not, then the PLP/ISAT strategy is optimal and there is no reason to use the parallel ISAT strategies that involve message passing. Note that even when t_C is small, we do not need to assume that it is entirely negligible. The main way in which we can reduce the communication time per particle is through aggregating many small messages (e.g., individual particles) into a larger message in order to reduce the overall latency penalty. This technique is commonly known as "message batching", and it is one of the keys to achieving good performance in all the software described below.

In this study, parallel ISAT algorithms are developed by developing distribution strategies to be used in combination with serial ISAT as follows. In the parallel calculation of reactive flows, each processor has its own ISAT table. During each reaction fractional step, the ensemble of particles to be resolved on one processor may be distributed to one or more other processors using different distribution strategies, resolved by the ISAT tables there, then sent back to the original processor. The message passing happens before and after the serial ISAT algorithm is invoked, not within ISAT. Different distribution strategies have been developed and implemented in the software *x2f_mpi*, namely, purely local processing (PLP), uniformly random distribution (URAN), and preferential distribution (PREF). For PLP, there is no message passing in the chemistry calculations, and particles on one processor are locally processed by the local ISAT table. For URAN, the particles in a group of processors are randomly distributed uniformly among all the processors in the group. For PREF, the particles have preference to some processors: that is, particles can only be passed to those processors that they have visited during a previous reaction step, or have not yet visited during the current reaction step. (For more details about PREF, see Appendix B.) It should be noted that the various distribution strategies can be used in combination as shown below.

Compared to the PLP strategy, the URAN and PREF strategies require message passing and hence extra message passing time. Also these strategies may incur synchronization penalties. However, considering that passing particles among the processors may result in much less computational cost for the resolution of particles, the strategies with message passing may still have computational advantages over PLP as far as the wall clock time for the reaction fractional step is concerned.

Besides the above three distribution strategies, there is one additional mode called *quick try* (QT), in which a retrieve attempt for all the particles is made based on the local ISAT table before using the distribution strategies in *x2f_mpi*. Only the particles unresolved by QT are passed to *x2f_mpi*, and therefore the number of particles requiring message passing can be dramatically reduced.

Fig. 5 illustrates how parallel ISAT strategies are used in calculations of reactive flows. In a parallel calculation with N_p processors, with domain decomposition the whole solution domain is divided into N_p sub-domains and each processor performs the computation of one sub-domain. During the reaction fractional step, each sub-domain has an ensemble of particles to be resolved. At the start of each reaction fractional step, QT may be invoked depending on the user's setting. Then all the particles unresolved by QT are partitioned into one or more blocks. The blocks are looped and each block of particles is distributed among some or all of the processors based on the distribution strategy specified, and resolved using the ISAT tables there. This process continues until the particles in all the blocks are resolved. We refer to the processes to resolve each block of particles (i.e., the processes inside the "loop over blocks" in Fig. 5) as a "block sub-step". As illustrated in Fig. 5, during each block sub-step, the particles in the blocks are redistributed by *x2f_mpi* among the processors, resolved there, and then passed back to the original processors.

The number of blocks required depends on the available physical memory and the amount of data in the unresolved particles. This is because temporary storage is required, the amount of which scales linearly with the block size. In general, to minimize interprocessor communication, the size of each block should be large. As mentioned earlier, passing particles among processors in small blocks or even singly increases the overall latency penalty. The use of numerous small blocks also increases the likelihood of synchronization delays. For small or medium scale calculations, a single block is in general sufficient.

5. Parallel ISAT with fixed distribution strategies

5.1. Parallel ISAT strategies: PLP/ISAT and URAN/ISAT

For parallel evaluation of ensembles of particles with ISAT, purely local processing (PLP) lies at one extreme of the range of possible strategies, because it has no message passing and no load redistribution: by definition, PLP/ISAT uses only the local ISAT tables. At the other extreme is URAN/ISAT, which combines uniform random distribution (URAN) with ISAT. In this strategy, during the reaction fractional step, the particles on each processor are randomly distributed uniformly (to within one particle) among all of the processors in the simulation, so that each processor has an equal number of particles to process. The major characteristics of the two extreme ISAT strategies are as follows:

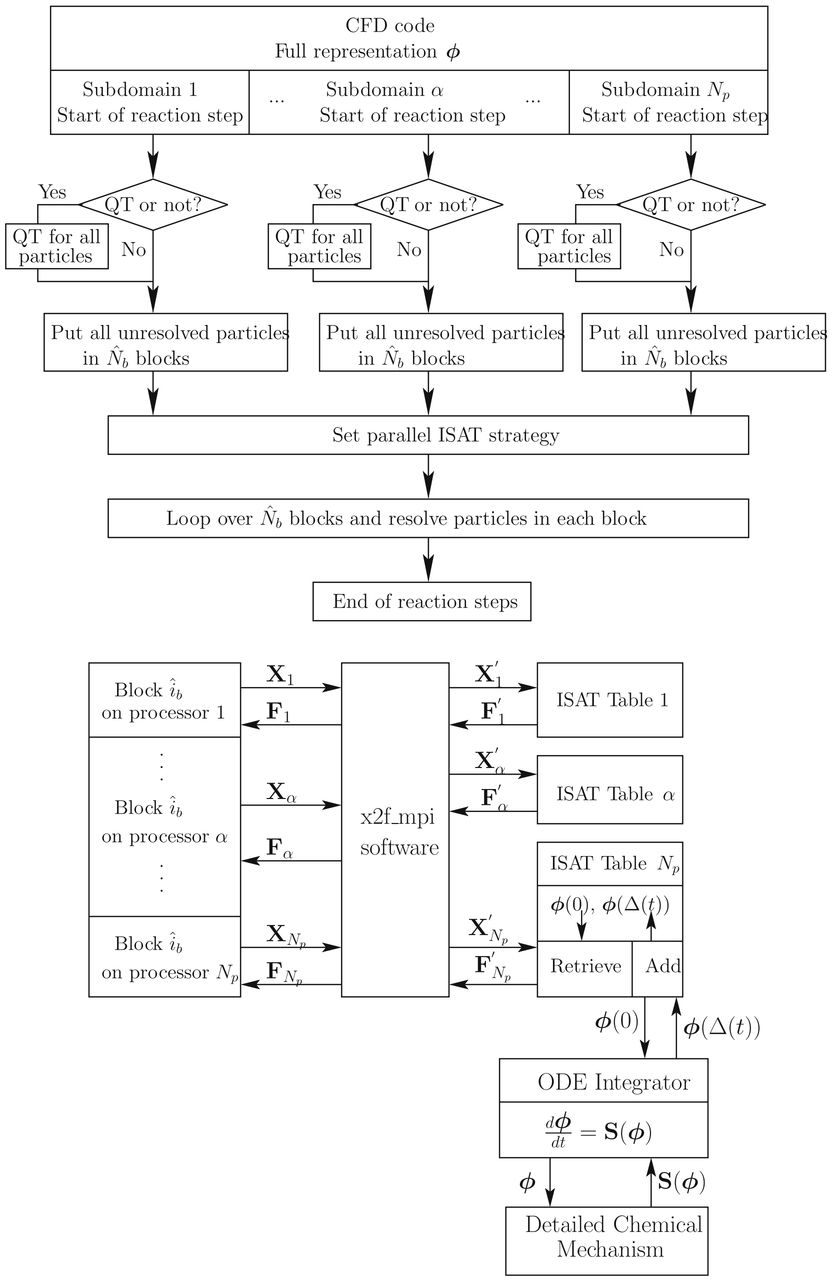

CFD code
Full representation ϕ
Subdomain 1
Start of reaction step
Subdomain α
Start of reaction step
Subdomain N_p
Start of reaction step
QT or not?
Yes
No
QT for all particles
Put all unresolved particles in $\hat{N}_b$ blocks
Set parallel ISAT strategy
Loop over $\hat{N}_b$ blocks and resolve particles in each block
End of reaction steps
Block $\hat{i}_b$ on processor 1
Block $\hat{i}_b$ on processor α
Block $\hat{i}_b$ on processor N_p
$\mathbf{X}_1$ $\mathbf{F}_1$ $\mathbf{X}_\alpha$ $\mathbf{F}_\alpha$ $\mathbf{X}_{N_p}$ $\mathbf{F}_{N_p}$
x2f_mpi software
$\mathbf{X}'_1$ $\mathbf{F}'_1$ $\mathbf{X}'_\alpha$ $\mathbf{F}'_\alpha$ $\mathbf{X}'_{N_p}$ $\mathbf{F}'_{N_p}$
ISAT Table 1
ISAT Table α
ISAT Table N_p
$\phi(0)$, $\phi(\Delta(t))$
Retrieve
Add
$\phi(0)$
$\phi(\Delta(t))$
ODE Integrator
$\frac{d\phi}{dt} = \mathbf{S}(\phi)$
ϕ
$\mathbf{S}(\phi)$
Detailed Chemical Mechanism

Fig. 5. Sketch showing the use of the parallel ISAT algorithm in a calculation of a reactive flow with N_p processors. With domain decomposition, the whole solution domain is divided into N_p sub-domains and each processor performs the computation of one sub-domain. The bottom subplot illustrates the processes in each block sub-step, where the particles in the blocks are redistributed by *x2f_mpi* among the processors, resolved there, and then passed back to the original processors.

- PLP/ISAT: no message passing; the local ISAT table depends on the local particle composition distribution $\mathcal{D}_k(\mathbf{x})$; load imbalance is possible due to the nonuniform intensity of chemical reactions or nonuniform distribution of computational particles.
- URAN/ISAT: much message passing; the ISAT tables on the N_p processors are statistically identical (and independent) and depend on the union of the composition distributions on all the processors, i.e., $\cup_\alpha \mathcal{D}_\alpha(\mathbf{x})$; the load balancing is perfect.

Fig. 4 shows the measured wall clock time and CPU time per particle step in the reaction fractional step from the nonuniform, coincident PaSR calculation, in which the first processor has 8 times the number of particles as the other processors. Due to the nonuniform distribution of the number of particles among the processors, the PLP/ISAT strategy exhibits a significant

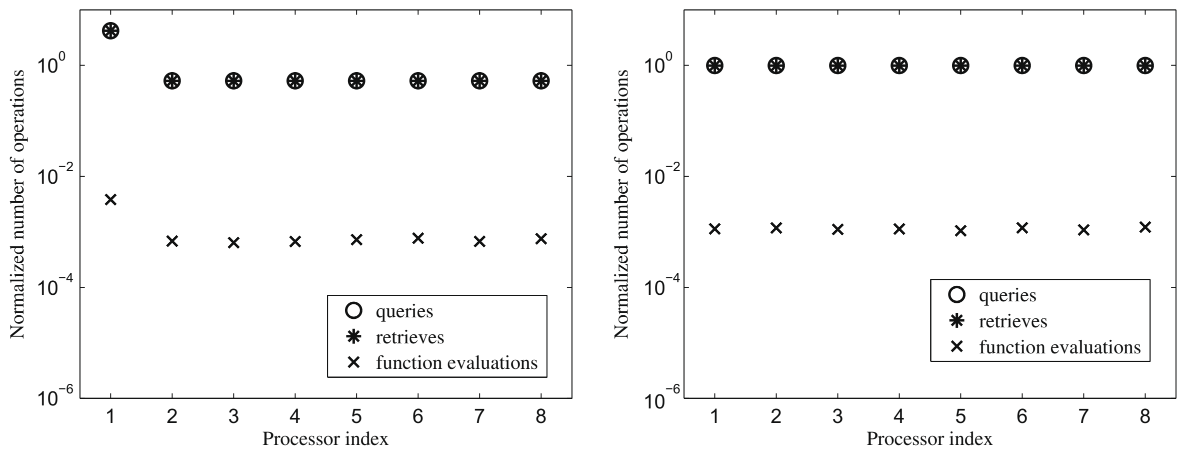


Fig. 6. Normalized number of operations for different events performed by ISAT on each processor from the nonuniform coincident PaSR calculations with $\varepsilon_{tol} = 5 \times 10^{-4}$ and $A = 1 \times 10^3$. The number of operations is normalized by the average number of queries among all the processors. Left plot: PLP/ISAT; right plot: URAN/ISAT. Symbol $\circ$: queries; $*$: retrieves; $\times$: function evaluations. The calculations result in an average of 1.9×10^8 queries per processor.

load imbalance. However, in the URAN/ISAT strategy, by using the load redistribution among the processors, good load balancing is achieved. This is confirmed in Fig. 6 which shows the number of different operations in ISAT on each processor given by the two different parallel ISAT strategies. For the PLP/ISAT strategy, the first processor has a larger number of queries to resolve, so the computation on this processor becomes the bottleneck of the whole simulation. In contrast, the URAN/ISAT strategy distributes the work evenly among all the processors. Compared to PLP/ISAT, even with the extra time spent in message passing, URAN/ISAT achieves a parallel speed-up factor of 2.4 for this particular case as far as the wall clock time is concerned. For this case, the average message passing time (two-way) per particle, t_C, is in the same order as retrieve time and $t_C \approx 16$ µs. The message passing time is measured by passing particles using *x2f_mpi* without performing any computational work.

Although the URAN/ISAT strategy guarantees good load balancing among processors, and in some computational regimes it achieves better performance than the PLP/ISAT strategy, this simple URAN/ISAT strategy is in general not the optimal strategy for minimizing the wall clock time. In the following, more sophisticated strategies are proposed based on the ideas of domain decomposition in composition space or a multi-stage process.

5.2. Domain decomposition in composition space

One reason that the chemistry computations can be subject to a load imbalance is that the particles are primarily assigned to processors based on their positions in physical coordinate space, rather than on any chemical properties. Thus, if the spatial distribution of particles is nonuniform, so is the computational load. It would therefore seem advantageous to define a different domain decomposition to apply to particles during the reaction fractional step, in order to group together particles of similar composition on the same processor. Each processor may then proceed to develop a specialized ISAT table that is particularly effective in evaluating its assigned types of particles. Even though this strategy necessarily involves substantial communication, comparable perhaps to URAN, the resulting enhancement in the probability of retrieve p_R may more than compensate for the penalty of constantly shuttling large numbers of particles between physical and compositional sub-domains.

In practice such a strategy turns out to be problematic, because by definition ISAT tables evolve as a simulation progresses, and they evolve in different ways when they tabulate different parts of the composition space. For example, if the composition space for a combustion process is partitioned according to mixture fraction, then some processors will collect particles that are either mostly air or mostly fuel. These particles undergo little or no reaction at all, and their final states are quickly tabulated. Other processors will gather "burning particles" whose final states may depend sensitively on their initial states. Such particles are difficult to tabulate completely, resulting in many grow and add operations. Even when the mixture-fraction partitions are allowed to adjust dynamically, the outcome is that large numbers of retrieves on some processors must be balanced against comparatively few function evaluations on other processors. This balance turns out to be rather difficult to achieve, because on the processors that receive "burning particles", statistical variations as well as systematic changes in the numbers of grow and add operations during a given step will continually throw off the expected workload. Therefore, the strategy of domain decomposition in composition space was deemed to be less than optimal at an early stage in this work [21,22].

5.3. Multi-stage process

As mentioned in Section 4, various ISAT processes can be invoked in attempting to resolve particles: retrieve attempt from the local ISAT table; retrieve attempt from the ISAT table on a remote processor; function evaluation on the local processor;

function evaluation on a remote processor. In the multi-stage procedure, a sequence of the above different processes is invoked across all processors in attempting to resolve particles with the minimum computational cost (i.e., the minimum wall clock time). The computationally cheap processes (e.g., retrieve attempts) are tried first: if the retrieve attempts fail, then computationally more expensive processes (e.g., function evaluations) are invoked to resolve particles. At each stage in a multi-stage process, a different distribution strategy such as URAN or PREF can be used to redistribute the unresolved particles among the processors.

It is significant to note that at each stage the ISAT processes with comparable computational cost are employed among the processors, e.g., either all perform retrieve attempts or all use function evaluations. This is necessary due to the difficulty previously encountered in balancing huge numbers of retrieve attempts against a few function evaluations, as described in the preceding subsection. Consequently good load balancing is in general achieved at each stage and therefore in the chemistry calculations among the processors.

5.4. Multi-stage parallel ISAT strategies: QT/URAN/ISAT and PREF/URAN/ISAT

The simplest parallel ISAT strategy that employs the multi-stage process idea is called QT/URAN/ISAT, where QT stands for "quick try". During the reaction step, in the QT stage, a retrieve attempt for particles is made based on the local ISAT table; then in the URAN stage, the particles unresolved by QT are randomly distributed uniformly among all the processors and are resolved there either by retrieves or by function evaluations. Note that in QT/URAN/ISAT, the ISAT table that develops on each processor is statistically identical and depends on the union of the composition distributions on all the processors (as in URAN/ISAT). This is because only the URAN stage affects the ISAT table building on each processor, and in this stage all the unresolved particles are independent and identically distributed (i.i.d.) among all the processors.

In QT/URAN/ISAT, by performing QT on the local ISAT table, most particles are successfully resolved, hence the number of the particles that need to be redistributed by URAN is substantially reduced, and so also is the message passing time. Furthermore, the QT/URAN/ISAT strategy puts more effort into trying computationally cheap retrieve attempts: queries that cannot be resolved by retrieves from the local ISAT table experience another retrieve attempt from another ISAT table on another processor instead of directly resorting to the computationally expensive function evaluation.

Notice that in PLP/ISAT, URAN/ISAT and QT/URAN/ISAT strategies, for each particle, retrieve attempts are made on only one or (at most) two processors. Recall that the computational cost of a function evaluation is several orders of magnitude larger than that of a retrieve. Computationally it may be worthwhile to put more effort into sending the particles among the processors and trying more attempts of retrieve. If particles can be resolved by retrieves instead of function evaluations, the wall clock time for resolving particles may still be smaller, even at the expense of extra message passing and retrieve attempts.

Based on the above reasoning, another parallel ISAT strategy, denoted as PREF (n_r)/URAN/ISAT, is developed, which allows for more retrieve attempts. In this strategy, for each particle, retrieve attempts are made on at most n_r processors, where $n_r \leqslant N_p$ is a user-specified parameter. Specifically, during each of the n_r retrieve stages, a retrieve attempt is made for unresolved particles; then particles resolved by this retrieve attempt are passed back to the original processor; and the remaining unresolved particles are passed to another processor using PREF for another retrieve attempt in the next retrieve stage. The same process continues until all particles have been resolved, or the number of retrieve attempts reaches the designated number n_r. In the URAN stage, all the unresolved particles are randomly distributed uniformly among all the processors and are resolved there by retrieves or function evaluations. (As discussed in Appendix B, in the first retrieve attempt, if the number of particles to be resolved is uniform or close to uniform among the processors, PREF forces the particles to try the first retrieve attempt from their local ISAT table; otherwise if the number of particles among the processors are significantly nonuniform, PREF distributes particles uniformly among the processors and the first retrieve attempt for particles is not necessarily made on the local ISAT table.) In PREF (n_r)/URAN/ISAT, the ISAT tables on the processors are statistically identical (but not independent) and depend on the union of all the composition distributions on all the processors. This is because only the URAN stage affects the ISAT table building on each processor and in this stage all the unresolved particles are independent and identically distributed (i.i.d.) among all the processors. However, an important observation is that the ISAT tables are not independent. This is because the compositions added to one table are those that could not be resolved on the n_r processors visited.

It is worth mentioning that the URAN/ISAT strategy described before is actually a special case (with $n_r = 0$) of the whole class of PREF/URAN/ISAT strategies. If the number of particles to be resolved is uniform or close to uniform among the processors, PREF(1)/URAN/ISAT performs similarly to QT/URAN/ISAT.

Fig. 7 shows the measured wall clock time and CPU time per particle step from the uniform, coincident PaSR calculations. For this particular case, with the PLP/ISAT strategy, each processor has a significant fraction of function evaluations (about 1.2%). In contrast, with the multi-stage process, the QT/URAN/ISAT and PREF(8)/URAN/ISAT strategies make more retrieve attempts, and the wall clock time decreases (by factors of 1.5 and 2.7, respectively) even though there is more message passing and unsuccessful retrieve attempts. This is because more particles are resolved by cheap retrieves instead of expensive function evaluations. This is confirmed by the recorded number of different operations in ISAT on each processor from the three different parallel ISAT strategies. On each processor, the fractions of function evaluations for QT/URAN/ISAT and PREF(8)/URAN/ISAT are about 0.9% and 0.5%, respectively. Compared to PLP/ISAT, PREF(8)/URAN/ISAT achieves a parallel speed-up factor of about 3 for this particular case.

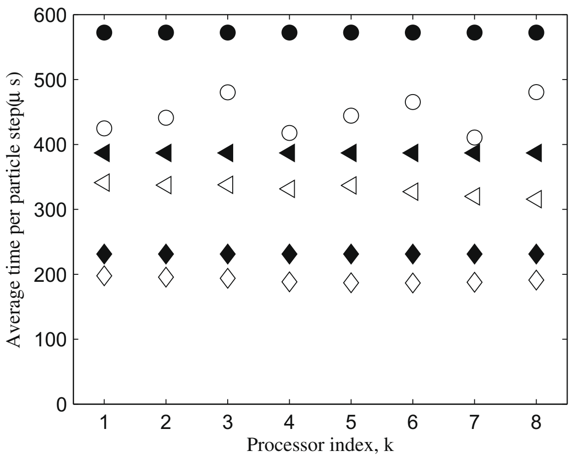


Fig. 7. Wall clock time and CPU time per particle step (in microseconds) for each processor from the uniform coincident PaSR calculations with $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 2 \times 10^3$. Solid symbol: wall clock time; open symbol: CPU time. Symbol ∘: PLP/ISAT; ◁: QT/URAN/ISAT; ◇: PREF(8)/URAN/ISAT. The calculations result in an average of 1.0×10^8 queries per processor.

6. Adaptive parallel ISAT strategy

As found in [21,22], none of the parallel ISAT implementations with fixed distribution strategies consistently achieves good performance in all the computational regimes. The optimal distribution strategy depends on the computational regime a calculation is in. To address this challenge, an adaptive parallel ISAT strategy is developed, in which the distribution strategy is determined on the fly based on a prediction of future calculation time in combustion chemistry, drawing on the results obtained in Section 3. The adaptive strategy is developed based on assumed statistical stationarity (at least approximately) of a calculation.

6.1. Overview

When applying the adaptive parallel ISAT strategy for a reactive flow calculation, the number of processors N_p must be an integer power of 2, and each processor maintains its own ISAT table. At the beginning of the simulation, the adaptive strategy involves up to M_s pairing stages with $M_s = \log_2(N_p)$. Initially the N_p processors in the simulation are partitioned into $N_g = N_p$ groups, with each group containing a single processor. In each pairing stage, the simulation runs until either (a) the ISAT tables on all the processors are "fully developed" (as described in Appendix C), or (b) the number of table entries in all the tables in one of the groups reaches a specified fraction of the number of allowed table entries. Then, based on a prediction of future calculation time, the adaptive strategy either maintains the existing grouping or forms a new grouping by pairing all of the existing groups. Thus the number of processors g in each group may double after each pairing stage. If a pairing of groups is performed at every pairing stage, then after the M_s-th stage there is only a single group containing all processors in the simulation.

With the adaptive ISAT strategy, at any given moment, the simulation has N_g group(s) with $g = N_p/N_g$ processor(s) in each group. During the reaction step, the following processes are invoked to resolve particles:

- Retrieve attempt(s). The ensemble of particles from each group is distributed among the processor(s) within the group and retrieve is attempted using one or more tables within the group. The distribution strategy employed is the preferential distribution (PREF). The maximum number of retrieve attempts for the unresolved particles is the number of processors in the group, g. Synchronization within each group occurs after each retrieve attempt.
- Function evaluation (through the events grow, add or discarded evaluation). Those particles that have not been resolved by retrieves, are randomly distributed evenly using the URAN strategy either within each group or among all the processors in the simulation. The unresolved particles are distributed among all the processors in the simulation to achieve good load balancing in workload only if the following conditions are satisfied: all the m pairing stages have been performed; and all of the ISAT tables on all the processors are fully developed. Otherwise, the unresolved particles are distributed evenly among the processors in each group so that ISAT tables can continue to be developed based on queries from within the group.

It is worth mentioning some extreme limits of the adaptive strategy. If no group pairing is performed in any pairing stage, then after the mth stage, there are still N_p groups with each containing a single processor. In this limit, if the unresolved particles are not distributed evenly among all the processors in the simulation during the URAN stage, the adaptive parallel ISAT strategy is equivalent to the PLP/ISAT strategy. At the other extreme, if the pairing of groups is performed in every pairing stage, then there is only one single group containing all the processors in the simulation after the pairing stages. In this limit, after all of the ISAT tables are fully developed, the adaptive parallel ISAT strategy mimics the PREF/URAN/ISAT strategy with up to N_p retrieve attempts. (There are subtle differences due to the difference in building the ISAT tables.)

In the following, we elaborate on the grouping algorithm.

6.2. Grouping algorithm

The adaptive strategy involves up to $M_s = \log_2(N_p)$ pairing stages. During the Lth pairing stage (with $1 \leqslant L \leqslant M_s$), the simulation has N_g groups of processors and the number of processors in each group is $g(= N_p/N_g \leqslant 2^{L-1})$. For the L-th stage, the simulation runs until (a) the ISAT tables on all the processors are fully developed or (b) the number of table entries on each processor of one group reaches a_L^*, where a_L^* is the maximum number of table entries allowed on each processor during the Lth stage. In the current implementation, a_L^* is specified as

$$a_L^* = A \times \begin{cases} \left[\left(\frac{1}{2}\right)^1 + \left(\frac{1}{2}\right)^2 + \cdots \left(\frac{1}{2}\right)^L\right] = 1 - \left(\frac{1}{2}\right)^L & \text{if } 1 \leqslant L < M_s \\ 1 \text{ if } L = M_s \end{cases}. \tag{13}$$

At the end of the Lth stage, either

1. the existing grouping is maintained (so that N_g is unchanged), or
2. a new grouping is formed by pairing all existing groups (so that N_g is halved).

It is worth mentioning that the above specification of a_L^* is tentative. Exploring other specifications and identifying the optimal one are certainly necessary and important for further improving the adaptive strategy.

The decision on whether and how to perform pairings is based on an estimation of wall clock time per block sub-step for a very long-run simulation assuming the use of all the allowed table entries. We denote by T_i' the estimated time for group i to accomplish the combustion chemistry calculations required in a block sub-step (in a long-run simulation using all of the allowable table entries) when the groups remain unpaired. Then the estimated wall clock time per block sub-step for the simulation, with the assumption of no pairing, is

$$T_{np}' = \max(T_i'). \tag{14}$$

We denote by T_{ij}' the estimated time per block sub-step (for one block of particles for a long-run simulation using all of the allowable table entries) for the hypothetical pairing of groups i and $j(i \neq j)$. The pairing of the existing groups is not unique. Let $\mathcal{P}_k$ denote the kth possible pairing, and the estimated wall clock time per block sub-step for the pairing is

$$T_{p,k}' = \max_{(i,j)\in\mathcal{P}_k}\left(T_{ij}'\right), \tag{15}$$

where $(i,j) \in \mathcal{P}_k$ denotes all the pairs of groups (i and j) in the pairing $\mathcal{P}_k$. The optimal pairing among the groups is the pairing with the minimum value of $T_{p,k}'$. (See Appendix D for more details about the algorithm for determining the optimal pairing.) The estimated wall clock time per block sub-step for the simulation with the optimal pairing is

$$T_p' = \min_k(T_{p,k}'). \tag{16}$$

If T_p' is less than T_{np}', the optimal pairing is used to form the new grouping for the next stage, and the number of processors in each group doubles. Otherwise, the existing grouping is maintained.

The details of how the estimates T_i' and T_{ij}' are made are in Appendix E.

7. Investigation of different parallel ISAT strategies in extreme computational regimes

Here we focus on investigating the relative performance of different parallel ISAT strategies in different extreme computational regimes: namely, coincident and disjoint query distributions both with uniform and nonuniform numbers of queries. These extreme circumstances correspond to extreme nonuniform distributions of reaction activity and computational particles among the sub-domains. The parallel ISAT strategies investigated here are PLP/ISAT, URAN/ISAT, QT/URAN/ISAT, PREF(8)/URAN/ISAT and the adaptive strategy. The decisions of the adaptive strategy in each stage for each case considered are listed in Table 1.

7.1. Coincident query distributions, uniform number of queries

In this regime, the composition distributions $\mathcal{D}(\mathbf{x})$ are identical among the processors. Suppose that the chemistry calculation is performed locally, i.e., without message passing and load redistribution among the processors. As in Section 3, we can define a critical number of table entries A^*, which is identical on each processor. Based on A^* and the number of processors N_p, this regime can be further categorized as

Table 1
Decision of adaptive strategy in each stage for different cases.

Case				Stage 1	Stage 2	Stage 3
Particle distribution	Composition distribution	ε_{tol}	A			
Uniform	Coincident	8×10^{-4}	2000	Pair	Pair	Not pair
Uniform	Coincident	5×10^{-4}	1000	Pair	Pair	Not pair
Uniform	Coincident	1×10^{-4}	2000	Pair	Pair	Pair
Nonuniform	Coincident	8×10^{-4}	2000	Pair	Pair	Pair
Nonuniform	Coincident	5×10^{-4}	1000	Pair	Pair	Pair
Nonuniform	Coincident	1×10^{-4}	2000	Pair	Pair	Pair
Uniform	Disjoint	8×10^{-4}	2000	Not pair	Not pair	Not pair
Uniform	Disjoint	5×10^{-4}	1000	Pair	Pair	Not pair
Uniform	Disjoint	1×10^{-4}	2000	Pair	Pair	Not pair
Nonuniform	Disjoint	8×10^{-4}	2000	Not pair	Not pair	Not pair
Nonuniform	Disjoint	5×10^{-4}	1000	Pair	Not pair	Not pair
Nonuniform	Disjoint	1×10^{-4}	2000	Pair	Pair	Pair

- Locally supercritical. The table size is locally supercritical if $A > A^*$. In this regime, the ISAT table on each processor is very effective, and the particles can almost always be successfully retrieved from the local ISAT table. Thus, the PLP/ISAT strategy is almost certainly optimal among all the ISAT strategies.
- Locally subcritical but globally supercritical. The table size is defined to be globally supercritical if the product $N_pA > A^*$. This means that, among all the processors there is sufficient storage to tabulate the most-accessed compositions in the simulation. This can hold true even if the table size is locally subcritical, i.e., one can have $A^*/N_p < A < A^*$. For this regime, a multi-stage strategy should be more efficient than PLP/ISAT; the latter will require many more function evaluations that can be avoided by using multi-stage retrieves from other processors.
- Globally subcritical. The table size is globally subcritical if the product $N_pA < A^*$. Again PLP/ISAT is inefficient because a substantial number of function evaluations will be required. However, the wall clock time can still be substantially reduced by attempting to retrieve from more ISAT tables on other processors, even if many of the attempts are unsuccessful.

Fig. 8 shows the results from the calculations of the uniform coincident PaSR cases with different specifications of the ISAT error tolerance and the allowed number of ISAT table entries. The calculations from the top to the bottom in the figure are designed to be from relatively easy to hard by varying ε_{tol} and A. Consequently, as shown, for the PLP/ISAT strategy, the normalized number of function evaluations gradually increases from the top to the bottom.

For all the cases considered here, URAN/ISAT gives comparable CPU time to PLP/ISAT, but requires a little more wall clock time because of message passing. With quick try, QT/URAN/ISAT substantially improves the performance (by more than 30%) compared to URAN/ISAT. The easiest case investigated here is close to the locally supercritical regime, and the ISAT table on each processor is sufficiently effective. Thus the performance of PLP/ISAT is comparable (within 20%) to the other parallel ISAT strategies (i.e., QT/URAN/ISAT, PREF/URAN/ISAT and the adaptive strategy). However, as the problem becomes harder, the performance of PLP/ISAT becomes worse. This is simply because the PLP/ISAT strategy does not take advantage of ISAT tables on other processors, thus it results in a relatively large number of function evaluations. For the hardest problem investigated here, the wall clock time by PLP/ISAT is about 3 times that of PREF(8)/URAN/ISAT or of the adaptive strategy. Another observation is that PREF(8)/URAN/ISAT and the adaptive strategy yield comparable performance (within 5%) for the cases considered here. In this regime, the adaptive strategy performs pairing twice for the two relatively easy cases and three times for the hardest case.

7.2. Coincident query distributions, nonuniform number of queries

In this regime the major factor affecting the computational efficiency is the load balancing issue due to the nonuniform number of queries among the processors. For the cases presented below, the first processor has 8 times the number of particles as the other processors. Similar to the uniform coincident regime, this regime can be further categorized as: locally supercritical, globally supercritical or globally subcritical. Now, even in the locally supercritical regime, due to the load imbalance, PLP/ISAT is not necessarily optimal among all the ISAT strategies.

Fig. 9 shows the results from calculations of nonuniform coincident PaSR cases with different specifications of the error tolerance and of the number of ISAT table entries allowed. Again, the calculations from the top to the bottom run from relatively easy to hard, as evidenced by the fact that the probability of function evaluations in PLP/ISAT gradually increases. In this regime, due to the nonuniform distribution of particles among the processors, the PLP/ISAT strategy exhibits a significant load imbalance and performs poorly: the first processor takes much more CPU time than the other processors, so the other processors have substantial idle time. The performance of PLP/ISAT worsens as the problem becomes harder. As expected, the strategies with load redistribution among the processors significantly improve the computational performance. For example, even for the easiest case, URAN/ISAT, PREF(8)/URAN/ISAT and the adaptive strategy achieve comparable performance, which is

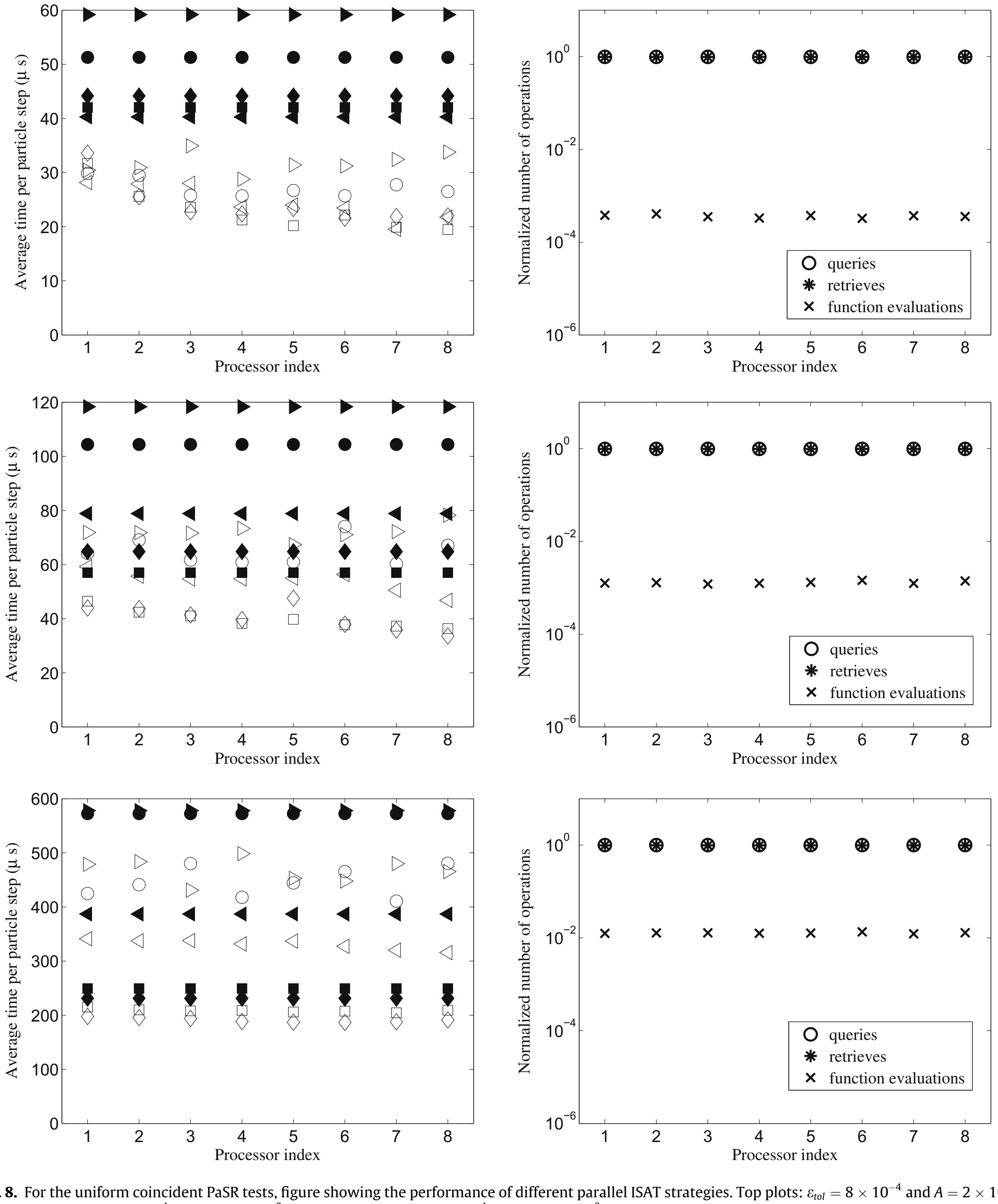


Fig. 8. For the uniform coincident PaSR tests, figure showing the performance of different parallel ISAT strategies. Top plots: $\varepsilon_{tol} = 8 \times 10^{-4}$ and $A = 2 \times 10^3$; middle plots: $\varepsilon_{tol} = 5 \times 10^{-4}$ and $A = 1 \times 10^3$; bottom plots: $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 2 \times 10^3$. Left column: wall clock time (solid symbols) and CPU time (open symbols) in the reaction fractional step (in microseconds per particle step) for each processor. Symbol ∘: PLP/ISAT; ▹: URAN/ISAT; ◃: QT/URAN/ISAT; ◇: PREF(8)/URAN/ISAT; □: adaptive. Right column: normalized number of operations for different events performed by ISAT on each processor from PLP/ISAT. Symbol ∘: queries; ∗: retrieves; ×: function evaluations. Each calculation results in an average of 1.0×10^8 queries per processor.

about 80% faster than PLP/ISAT. (For this highly nonuniform case, PREF(8)/URAN/ISAT and the adaptive strategy uniformly distribute particles among processors even in the first retrieve attempt.) Also, as shown for the easiest case, QT/URAN/ISAT gives poor performance (similar to PLP/ISAT), and is notably worse than URAN/ISAT. This is due to the load imbalance in the quick try mode, and is in contrast to the beneficial effect of QT in the uniform case (Fig. 8). As the problem gets harder

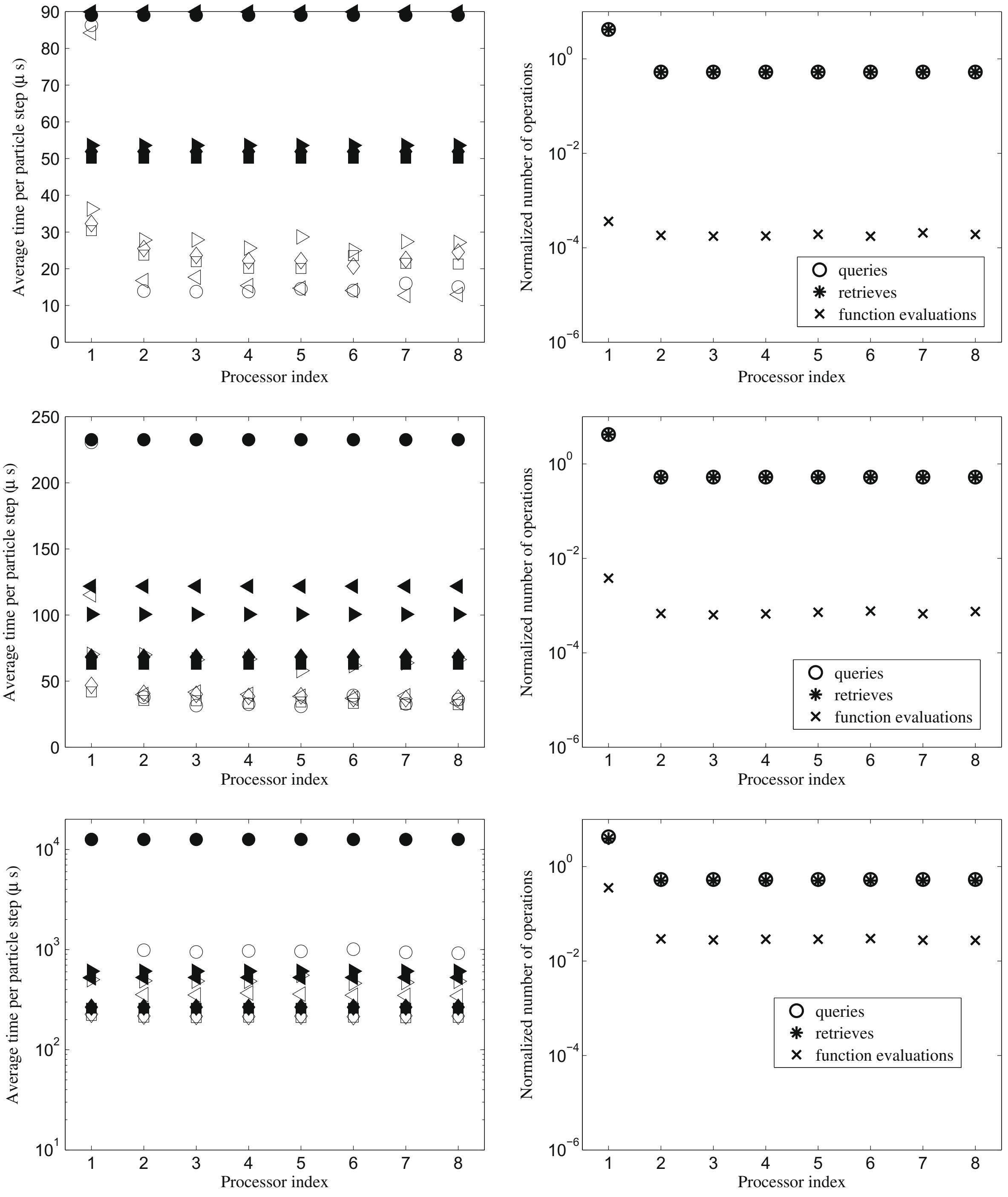


Fig. 9. For the nonuniform coincident PaSR tests, figure showing the performance of different parallel ISAT strategies. Top plots: $\varepsilon_{tol} = 8 \times 10^{-4}$ and $A = 2 \times 10^3$; middle plots: $\varepsilon_{tol} = 5 \times 10^{-4}$ and $A = 1 \times 10^3$; bottom plots: $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 2 \times 10^3$. Left column: wall clock time (solid symbols) and CPU time (open symbols) in the reaction fractional step (in microseconds per particle step) for each processor. Symbol ∘: PLP/ISAT; ▹: URAN/ISAT; ◃: QT/URAN/ISAT; ◇: PREF(8)/URAN/ISAT; □: adaptive. Right column: normalized number of operations for different events performed by ISAT on each processor from PLP/ISAT. Symbol ∘: queries; ∗: retrieves; ×: function evaluations. Each calculation results in an average of 1.0×10^8 queries per processor.

and the contribution of function evaluations becomes significant, compared to PLP/ISAT and URAN/ISAT, the performance of QT/URAN/ISAT is significantly improved due to the extra retrieve attempt, which results in more particles being resolved by cheap retrieves.

Also as the problem gets harder, PREF(8)/URAN/ISAT and the adaptive strategy show greater advantages over URAN/ISAT and QT/URAN/ISAT which attempt to retrieve from only one or two (at most) tables. As can be seen from Table 1, in this regime, for all three cases considered, the adaptive strategy performs pairing three times and there is only one single group containing all 8 processors after the pairing stages. For all three cases considered, PREF(8)/URAN/ISAT and the adaptive strategy yield similar performance, and for the hardest case, they yield a speed-up factor of about 30 compared to PLP/ISAT.

7.3. Disjoint query distributions, uniform number of queries

The essential feature of having disjoint query distributions is that an ISAT table built on one processor using PLP/ISAT has no value in enabling queries from other processors to be retrieved. Also the function evaluation time may vary significantly from processor to processor. When using the PLP/ISAT strategy, load imbalance may arise due to the nonuniform reaction activity among the processors.

Fig. 10 shows the results from the calculations of the uniform disjoint PaSR cases with different specifications of the error tolerance and of the number of ISAT table entries allowed. Again, the calculations from the top to the bottom run from relatively easy to hard, as evidenced by the fact that the probability of function evaluations in PLP/ISAT gradually increases. For the disjoint cases considered here, the conditions of inflowing streams for each reactor are set so that each reactor has intense reaction. However, particle compositions on each reactor come from different parts of the composition space and there is no overlap. In this regime, as may be seen from the figure, PLP/ISAT yields relatively good performance since an ISAT table only processes the local particles and only tabulates a particular part of composition region based on the local particles. In contrast, the URAN/ISAT strategy needs to tabulate a much wider composition region in each table with limited memory since each processor can encounter some particles from other processors. As may be seen, the performance of URAN/ISAT becomes very poor (about 4 times slower than PLP/ISAT) for the hardest problem considered here. Similarly, QT/URAN/ISAT performs poorly for all the cases considered. The figure also shows that PREF(8)/URAN/ISAT performs poorly, especially for the relatively easy cases. The reason is similar to the one for URAN/ISAT: it needs to tabulate a wider composition region on each table with limited memory.

Another important observation is that in this regime, by properly grouping processors and possibly taking advantage of ISAT table(s) from other processor(s), the adaptive strategy shows consistently good performance for all three cases considered here. For the easiest case, no pairing is performed (see Table 1) and the adaptive strategy approaches the PLP/ISAT limit. For the two relatively difficult cases considered, the adaptive strategy performs pairing in the first two pairing stages.

7.4. Disjoint query distributions, nonuniform number of queries

The nonuniform disjoint cases are constructed by specifying the first processor to have 8 times the number of particles as the other processors with the stream settings being the same as the uniform disjoint cases. The calculations from the top to the bottom in Fig. 11 again run from relatively easy to hard.

Fig. 11 shows the results from the calculations of the nonuniform disjoint PaSR cases with different specifications of the error tolerance and the number of table entries allowed. In this regime, as may be seen from the figure, PLP/ISAT, URAN/ISAT, and QT/URAN/ISAT generally have poor performance. In contrast, PREF(8)/URAN/ISAT generally yields the best performance. For the easiest case considered, PREF(8)/URAN/ISAT outperforms the other three strategies by saving 25% to 35% wall clock time. Another key observation is that the performance of the adaptive strategy gradually improves when the problem becomes harder. For the easiest case, no pairing is performed in the adaptive strategy and it achieves performance comparable to PLP/ISAT. As the problem becomes harder, to distribute the expensive function evaluations among the processors and speed up the calculation, the adaptive strategy performs pairing once for the intermediate case and three times for the hardest case, respectively. For the hardest case considered, the adaptive strategy slightly outperforms PREF(8)/URAN/ISAT and both are about 3 and 4 times faster than URAN/ISAT and PLP/ISAT, respectively.

As mentioned, for the easiest case, no pairing is performed in the adaptive strategy. Throughout the calculation, ISAT tables are not fully developed based on the current criterion adopted. Based on the current procedure for the URAN stage (see Section 6.1), during the URAN stage, expensive function evaluations are performed locally without redistribution among the processors. (For this case, the adaptive strategy is equivalent to PLP/ISAT.) This significantly degrades the performance due to the load imbalance of expensive function evaluations in the URAN stage as shown in Fig. 11. To determine the optimal procedure for the URAN stage and hence to further improve the adaptive strategy is left to future work.

To further investigate whether or not the adaptive strategy makes the right pairing decision for this easiest case, the following three calculations are performed, where forced pairing is made at the first pairing stage, at both first and second pairing stages, and at all three pairing stages. The calculations then continue to run with the forced pairing for a sufficiently long time. Fig. 12 shows the performance of these three calculations together with the performance of the adaptive strategy (which results in no pairing). As shown, the computational performance degrades as more pairings are performed, which implies that the adaptive strategy makes the right pairing decision for this easiest case.

Another interesting observation is that the performance of the calculation with three forced pairings is much worse than that of PREF(8)/URAN/ISAT. Recall that the only differences in the calculations with three forced pairings and with PREF(8)/URAN/ISAT are due to the difference in building the tables. This implies that the performance of a particular parallel ISAT strategy may strongly depend on how the ISAT tables are built among the processors. To understand the effect of table building and hence to further improve the adaptive strategy is left to future work.

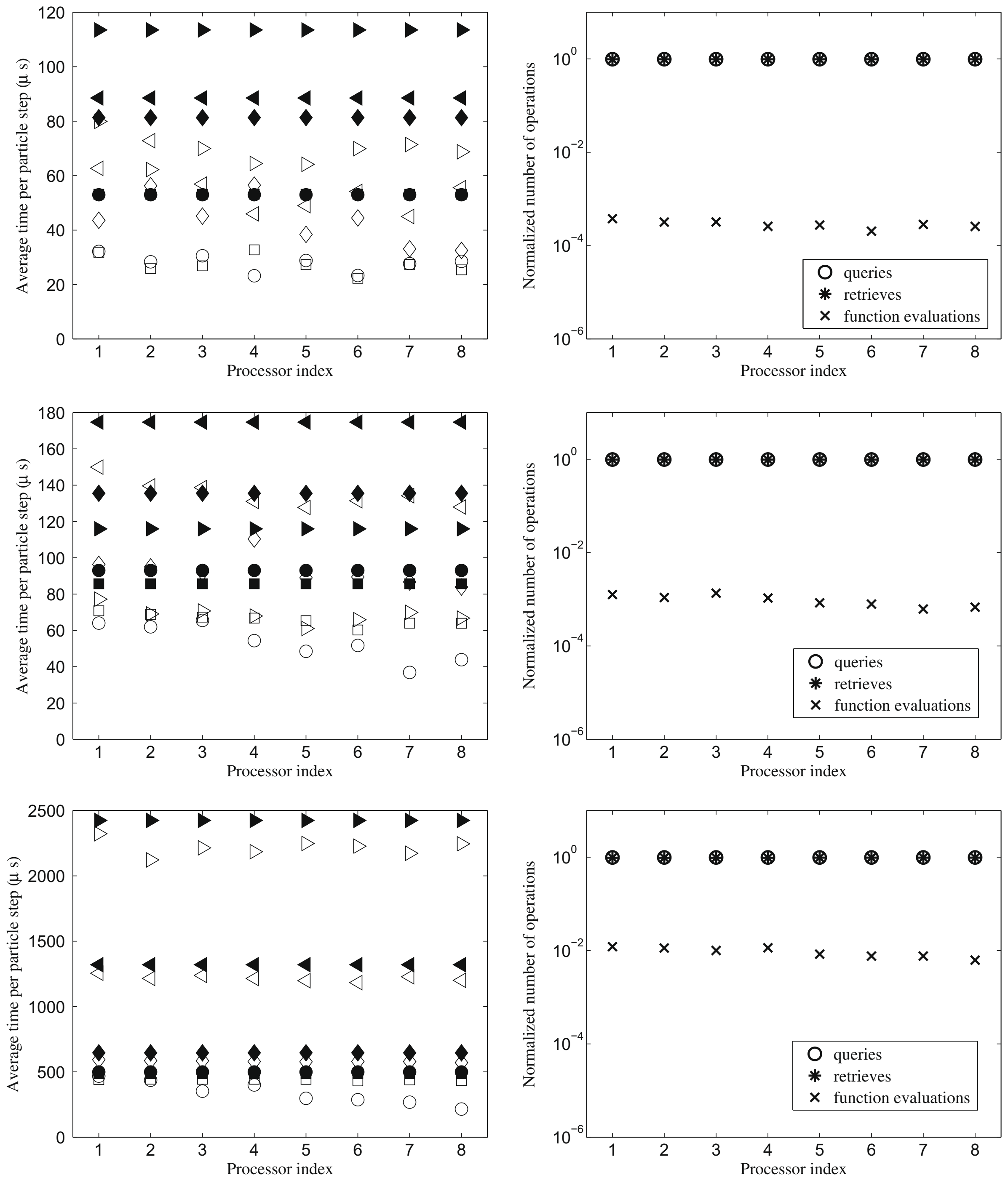


Fig. 10. For the uniform disjoint PaSR tests, figure showing the performance of different parallel ISAT strategies. Top plots: $\varepsilon_{tol} = 8 \times 10^{-4}$ and $A = 2 \times 10^3$; middle plots: $\varepsilon_{tol} = 5 \times 10^{-4}$ and $A = 1 \times 10^3$; bottom plots: $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 2 \times 10^3$. Left column: wall clock time and CPU time in reaction fractional step (in microseconds per particle step) for each processor. Solid symbol: wall clock time; open symbol: CPU time. Symbol $\circ$: PLP/ISAT; $\triangleright$: URAN/ISAT; $\triangleleft$: QT/URAN/ISAT; *diamondsuit*: PREF(8)/URAN/ISAT; $\square$: adaptive. Right column: normalized number of operations for different events performed by ISAT on each processor from PLP/ISAT. Symbol $\circ$: queries; $*$: retrieves; $\times$: function evaluations; Each calculation results in an average of $1.0 \times \ 10^8$ queries per processor.

8. Effect of the number of processors on the parallel ISAT performance

In this section, we investigate the dependence of the parallel ISAT performance on N_p, i.e., the number of processors used in the multi-PaSR calculations. In particular, we investigate the parallel scaling of the adaptive ISAT strategy and the PLP/ISAT

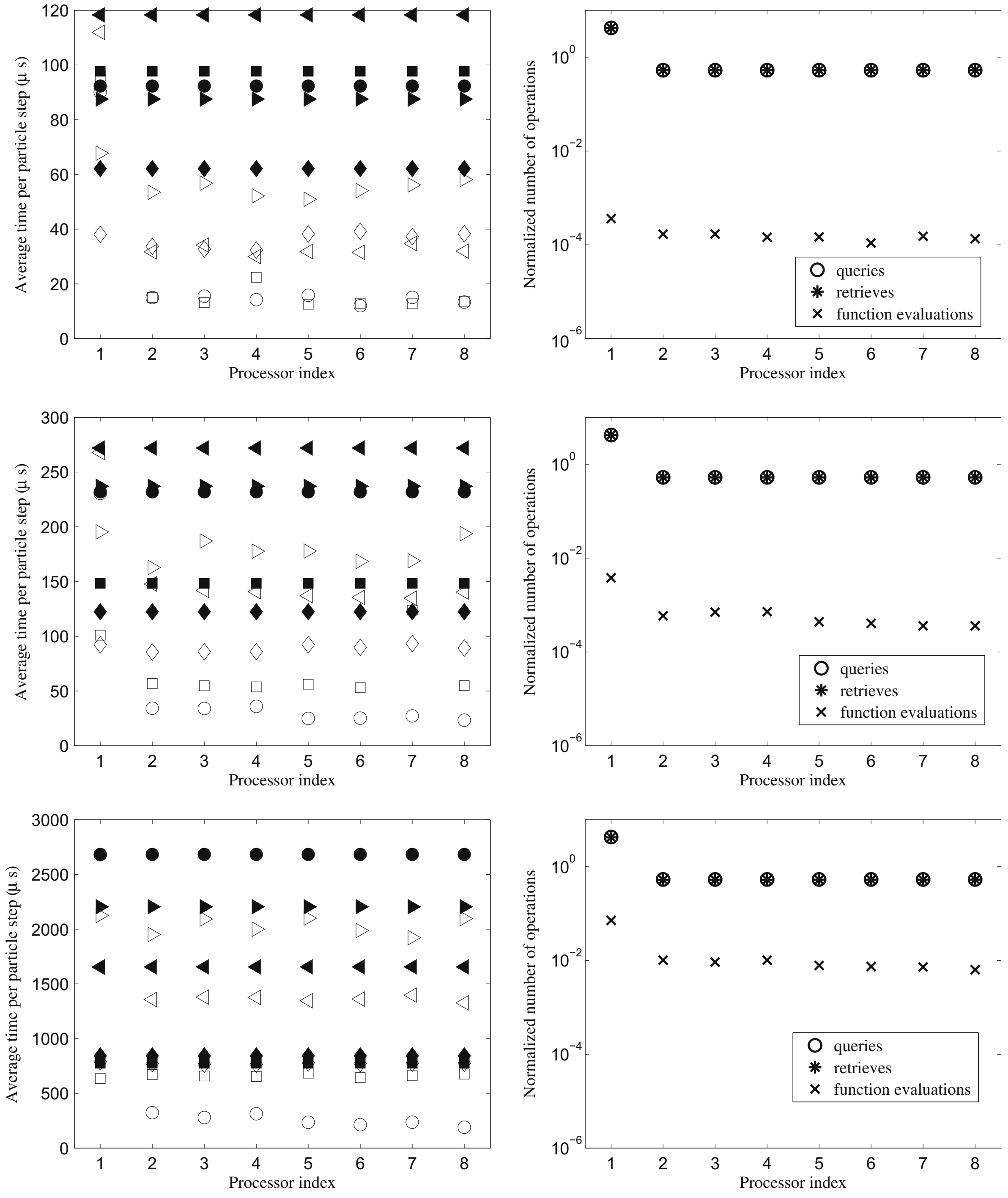


Fig. 11. For the nonuniform disjoint PaSR tests, figure showing the performance of different parallel ISAT strategies. Top plots: $\varepsilon_{tol} = 8 \times 10^{-4}$ and $A = 2 \times 10^3$; middle plots: $\varepsilon_{tol} = 5 \times 10^{-4}$ and $A = 1 \times 10^3$; bottom plots: $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 2 \times 10^3$. Left column: wall clock time and CPU time in reaction fractional step (in microseconds per particle step) for each processor. Solid symbol: wall clock time; open symbol: CPU time. Symbol $\circ$: PLP/ISAT; $\triangleright$: URAN/ISAT; $\triangleleft$: QT/URAN/ISAT; $\diamond$: PREF(8)/URAN/ISAT; $\square$: adaptive. Right column: normalized number of operations for different events performed by ISAT on each processor from PLP/ISAT. Symbol $\circ$: queries; $*$: retrieves; $\times$: function evaluations; Each calculation results in an average of 1.0×10^8 queries per processor.

strategy on the Velocity 3 cluster at the Center for Advanced Computing at Cornell University. The cluster consists of 170 commodity rack-mounted servers with 2 CPUs per sever, connected via a standard switched network (gigabit Ethernet and a fully nonblocking switch). The parallel calculations are performed by maintaining the problem size on each processor

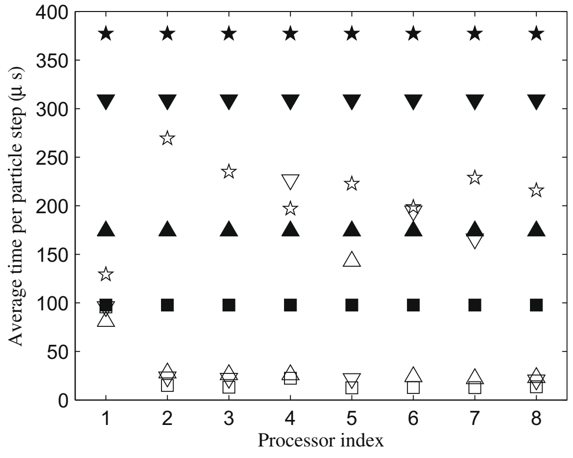


Fig. 12. For the nonuniform disjoint PaSR test with $\varepsilon_{tol} = 8 \times 10^{-4}$ and $A = 2 \times 10^3$, figure showing the ISAT performance with forced pairing in the adaptive strategy. Solid symbols: wall clock time; open symbols: CPU time. Symbols □: adaptive with no pairing; △: adaptive with single pairing forced at the first pairing stage; ▽: adaptive with two pairings forced at the first and second pairing stages; ★: adaptive with three pairings forced at all three pairing stages. Each calculation results in an average of 1.0×10^8 queries per processor.

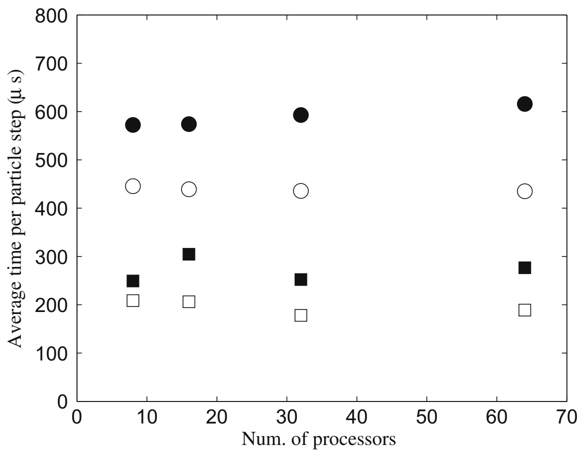


Fig. 13. For the uniform coincident PaSR calculations with $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 2 \times 10^3$, figure showing the average wall clock time and CPU time in reaction fractional step (per particle step in microseconds) against the number of processors; Solid symbols: wall clock time; open symbols: CPU time. Symbol ○: PLP/ISAT; □: adaptive.

constant (i.e., a fixed number of reactors on each processor, which is 1 in all the calculations) while varying the number of processors. Calculations were conducted for 8, 16, 32 and 64 processors.

Fig. 13 shows the wall clock time and CPU time per particle step against the number of processors in both the PLP/ISAT and adaptive strategies for the uniform coincident cases. Unsurprisingly, the CPU time and wall clock time for the PLP/ISAT remain almost constant over the number of processors considered. However it is interesting to observe that the CPU time and wall clock time for the adaptive strategy also remain almost constant. The reason is that the performance of the adaptive strategy is mainly controlled by the number of pairings performed (i.e., the number of processors in each group), not the number of processors used in the calculation. For the calculations run on 8, 16, 32, and 64 processors, the adaptive strategy performs 3, 3, 4, and 4 pairings, respectively.

As may be seen, the wall clock time ratio between these two strategies (about 2) remains almost constant over the number of processors considered, which implies that the adaptive strategy has good scalability up to at least 64 processors for the case considered here.

9. Discussions and future work

9.1. Primary retrieve and secondary retrieve

In ISAT5 [2], retrieve attempts consist of two different retrieve procedures, namely *primary retrieve* and *secondary retrieve*. During a retrieve attempt, ISAT first traverses the binary tree and possibly scans two list-based data structures [2,35] (which store the most frequently used and the most recently used tabulated entries) to check if the query point lies within the EOA of a tabulated point. The query is resolved if it is in any identified EOA, and then a linear approximation to the solution is returned. This outcome is a *primary retrieve*. Otherwise, ISAT tests more EOAs of tabulated points, attempting to find an EOA containing the query point. The query is resolved if it is in any identified EOA, and then a linear approximation to the solution is returned. This outcome is a *secondary retrieve*. The amount of secondary retrieving attempted is controlled so that the CPU time spent on it (in a given query) is less than a specified fraction of the average CPU time per function evaluation. It should be noted that the computational cost associated with *primary retrieve* and *secondary retrieve* may be quite different, e.g., the CPU time per retrieve in *secondary retrieve* may be orders of magnitude higher than that in *primary retrieve*.

In the multi-stage parallel ISAT strategies, due to the difference in computational cost, it is possible to introduce load imbalance in the PREF rounds when primary and secondary retrieves are used in combination in the retrieve attempts, which may significantly degrade the parallel ISAT performance. (Such phenomena are not observed in the current PaSR calculations due to the small tables used and the relatively easy problems considered.)

The approach to solve the above issue is to separate *primary retrieve* attempts and *secondary retrieve* attempts into different PREF rounds. Instead of performing retrieve attempts with *primary retrieve* and *secondary retrieve* combined together, an alternative way is to perform attempts of *primary retrieve* first and then perform attempts of *secondary retrieve* with both using the PREF distribution strategy.

9.2. Quick try (QT)

As mentioned, the advantage of performing QT is that it can significantly reduce the number of unresolved particles requiring message passing. However if the time spent in QT among processors is significantly different, then it can introduce significant load imbalance during this process. The load imbalance in QT may be caused by a nonuniform distribution of particles among processors (as shown in Fig. 9) or a nonuniform distribution of primary and secondary retrieve attempts in QT.

Currently it is still not clear how the optimal QT should be performed. Should it be based on *primary retrieve* or the combination of *primary retrieve* and *secondary retrieve*? Is it possible to adaptively determine how much QT is optimal (i.e., none, *primary retrieve*, or the combination of *primary retrieve* and *secondary retrieve*)?

For the results reported above, the combination of *primary retrieve* and *secondary retrieve* is used in QT. In the current parallel ISAT implementation, QT needs to be manually specified by users (either based on *primary retrieve* or the combination of *primary retrieve* and *secondary retrieve*). As long as memory is not a primary consideration, QT should not be invoked for the PREF strategies and the adaptive strategy. This is due to the following reasons. In the first retrieve attempt in PREF, if the number of particles to be resolved is uniform or close to uniform among the processors, PREF forces the particles to try the first retrieve attempt from their local ISAT table. Hence it is equivalent to QT. Otherwise if particles among the processors are significantly nonuniform, PREF distributes particles uniformly among the processors and the first retrieve attempt for particles is not necessarily made on the local ISAT table. Under this circumstance, QT should not be invoked as PREF effectively does this, taking into account load balancing.

9.3. The number of table entries allowed on the Lth stage a_L^*

The number of table entries allowed for each processor on the Lth stage, a_L^*, is an important parameter in the adaptive strategy which may affect parallel ISAT performance. There are various ways to specify it. With the current specification given by Eq. (13), most of the table is built based on local particles during the first pairing stage, which are considered more useful than those tabulated during later stages for a statistically stationary problem. The advantage of this specification is as follows: the local tables contains significant local composition information; thus instead of passing large numbers of particles and trying to retrieve from remote ISAT tables on the other processors, significant fractions of particles can be resolved by using the local ISAT table; hence the number of particles requiring message passing is significantly reduced. The disadvantage of this specification is that the ISAT tables may contain significant amounts of redundant information, hence the assigned ISAT storage has not been fully utilized.

An alternative way to build the ISAT tables that makes better usage of the allowed memories, is as follows. In the Lth stage of the $M_s = \log_2(N_p)$ pairing stages, the simulation runs until (a) the ISAT tables on all the processors are fully developed or (b) the number of table entries on each processor of one group reaches $a_L^* = A$, i.e., the table is full. At the end of the Lth stage, after the pairing decision is made, all the ISAT tables are deleted and then rebuilt by using PREF (g)/URAN/ISAT with g determined from the adaptive strategy. The disadvantage of this specification is that it spends a significant amount of extra time in building ISAT tables. However considering extremely long-run simulations where the time spent in building tables is negligible, rebuilding ISAT tables for better future usage may still be beneficial. This option is worthy of further investigation.

9.4. Application to a very large number of processors

The current study demonstrates good scalability for the adaptive strategy up to 64 processors. Whether this strategy is able to perform consistently well with a larger number of processors (e.g., 10^4 or 10^5) or with more complex chemistry remains in question and needs to be further investigated. Several limitations in the current adaptive algorithm are:

- The adaptive strategy requires the computations of P_{ij} and $t_{Rj \to i,w}$. To compute them requires at least $O\left(N_p^2\right)$ operations. Consequently, for sufficiently large N_p, the cost of computing P_{ij} and $t_{Rj \to i,w}$ will dominate that of most other processes which scale linearly with N_p.
- The adaptive strategy requires pairing among groups. The algorithm currently used (see Appendix D) for determining optimal pairing among groups runs in $O\left(N_p^3 \log(N_p)\right)$ operations. Consequently, for sufficiently large N_p, the cost of the paring process will dominate that of most other processes which scale linearly with N_p.
- For large N_p, with a large number of pairing stages, the pairing phase in the adaptive strategy may take a significant fraction of the simulation time.

One possible approach to solve the above limitations is that for a simulation with large N_p, at the beginning of the simulation one can partition the N_p processors into several groups with each group having a smaller number of processors (e.g., 64) and then apply the adaptive strategy in each group.

Another limitation with the current adaptive strategy is that the number of processors used in the simulation, N_p, needs to be an integer power of 2. Extension of the strategy to the general number of processor needs to be further considered.

10. Conclusion

The *in situ* adaptive tabulation (ISAT) algorithm [1] has been widely used for the efficient calculation of combustion chemistry in simulations of reactive flows. When it is employed for combustion chemistry in a parallel calculation, even though ISAT substantially speeds up the chemistry calculation on each processor, the overall load imbalance in the chemistry calculations among the processors severely affects the computational efficiency. To further improve the efficiency in chemistry calculations in parallel calculations, in this work, the original serial ISAT algorithm is extended to the multi-processor environment. The platform considered to perform parallel calculations is a distributed memory system. Message passing among the processors is performed using MPI. The whole computational domain is decomposed into sub-domains using domain decomposition, and each processor performs the computation for one sub-domain.

Parallel ISAT strategies have been developed via the use of different distribution strategies combined with the serial ISAT algorithm. In calculations with a parallel ISAT strategy, each processor has its own ISAT table. During the chemistry calculation, particles on one processor may be distributed to one or more other processors by distribution strategies, and be resolved by the ISAT tables there. Particles are distributed by message passing before and after the serial ISAT algorithm, not within ISAT. Three different distribution strategies have been developed and implemented in the software *x2f_mpi*, namely, purely local processing (PLP), uniformly random distribution (URAN), and preferential distribution (PREF). For PLP, there is no message passing in the chemistry calculations, and particles on each processor are locally processed by the local ISAT table. For URAN, the particles in a group of processors are randomly distributed uniformly among all the processors in the group using message passing. For PREF, the particles have preference to some processors, that is, particles can only be passed to processors that they have not visited previously during that reaction step. The above three different distribution strategies can be used in combination.

Different parallel ISAT strategies are developed with fixed or adaptive distribution strategies. For parallel ISAT with fixed distribution strategies, the distribution strategy is pre-specified by the user before a calculation, whereas for the adaptive strategy, the optimal distribution strategy is determined on the fly based on the prediction of future simulation time. For all the parallel ISAT strategies, computational efficiency in the chemistry calculations is achieved through the use of the multi-stage processes and load redistribution among processors using MPI.

The ISAT performance for chemistry calculations depends strongly on both the distribution of queries and the distribution of compositions among the processors. The relative performance of different parallel ISAT strategies is investigated in four extreme computational regimes, namely, coincident composition distributions with uniform and nonuniform numbers of queries, and disjoint composition distributions with uniform and nonuniform numbers of queries. To fulfill this purpose, the serial PaSR is extended to the multi-processor environment, and the parallel PDF calculations of multiple partially stirred reactors burning methane/air mixtures are performed. The multiple PaSR test has the advantage of simplicity in terms of controlling the distribution of particle compositions $\mathcal{D}(\mathbf{x})$ and the number of queries on each proces- sor.

The results show that the performance of the parallel ISAT strategies with fixed distribution strategies depends strongly on the computational regimes in which the problem is located; no single fixed strategy consistently achieves good performance in all computational regimes. For example, PREF/URAN/ISAT and URAN/ISAT perform poorly in the uniform disjoint regime, whereas the PLP/ISAT performs poorly in the nonuniform coincident regime. In contrast, the adaptive ISAT strategy yields reasonably good performance in all regimes. Compared to the PLP/ISAT strategy, where the chemistry calculation is essentially serial, a speed-up factor of up to 30 is achieved by the adaptive parallel ISAT strategy. The study also demonstrates that the adaptive strategy has very good parallel scalability.

It is worth mentioning that the software *x2f_mpi*, with different distribution strategies implemented, can be used for efficient general function evaluations in parallel calculations.

Acknowledgment

This research was conducted using the resources of the Cornell University Center for Advanced Computing, which receives funding from Cornell University, New York State, the National Science Foundations, and other leading public agencies, foundations, and corporations. This work is supported by the National Science Foundation under Grant No. CBET-0426787.

Appendix A. Probability of function evaluation after many queries

A fundamental quantity in developing an understanding of ISAT performance is the probability of function evaluation $p_F(q,A)$, which is defined to be the sum of the probabilities of grow, add, and discarded evaluation, i.e.,

$$p_F(q,A) \equiv p_G(q,A) + p_A(q,A) + p_D(q,A) = 1 - p_R(q,A). \tag{A.1}$$

The purpose of this appendix is to predict the probability of function evaluation p_F for a very long-run simulation with an allowed number of table entries A, i.e., $p_F(\infty,A)$, based on information available at the point when the table becomes full.

A.1. Asymptotic behavior of $p_F(\infty,A)$

For a given ISAT task, during the calculation, suppose that geometrically the whole composition space can be partitioned into three regions: retrieve region, $\mathcal{R}$; grow region, $\mathcal{G}$; and add region $\mathcal{A}$. During the calculation, these three regions $\mathcal{R}, calG$, and $\mathcal{A}$ evolve, and they depend on the number of allowed table entries A and the number of queries performed q. The categorization is based on how a query with a particular value of composition $\mathbf{x}^q$ would be resolved. That is, if a query $\mathbf{x}^q$ results in a retrieve, then $\mathbf{x}^q$ is by definition in the retrieve region $\mathcal{R}$; if it results in a grow, it is in the grow region $\mathcal{G}$; and if it results in an add, it is in the add region $\mathcal{A}$. Let $\mathcal{R}(q,A), \mathcal{G}(q,A)$ and $\mathcal{A}(q,A)$ denote the retrieve region, grow region and add region on the qth query with allowed table entries A. The probabilities of different events $p_R(q,A), p_G(q,A)$, and $p_A(q,A)$ are just the probabilities of the qth query $\mathbf{x}^q$ being in the regions $\mathcal{R}(q,A), \mathcal{G}(q,A)$, and $\mathcal{A}(q,A)$, respectively.

Let $q_f(A)$ be the query on which the ISAT table becomes full. Consider stopping adding at the query point $q = q_f(A)$ and continuing for an infinite number of queries. We consider the following simple model. Due to growing, the retrieve region $\mathcal{R}$ grows as much as it can so that for an infinite number of queries, the retrieve region $\mathcal{R}$ is the union of the retrieve region and the grow region when the table became full, and the grow region at $q = \infty$ becomes empty, i.e.,

$$\mathcal{R}(\infty,A) = \mathcal{R}(q_f(A),A) \cup \mathcal{G}(q_f(A),A), \qquad \mathcal{G}(\infty,A) = \emptyset \tag{A.2}$$

Notice that this simple model assumes that growing does not cause the EOAs to extend into the add region, and hence any query in the add region results in a discarded evaluation. According to the model (with $p_A(\infty,A) = 0$), we have

$$\begin{aligned} &p_R(\infty,A) = p_R(q_f(A),A) + p_G(q_f(A),A),\\ &p_G(\infty,A) = 0,\\ &p_D(\infty,A) = p_A(q_f(A),A),\\ &p_F(\infty,A) = p_G(\infty,A) + p_D(\infty,A) = p_D(\infty,A) = p_A(q_f(A),A). \end{aligned} \tag{A.3}$$

Hence, for a very long-run simulation with allowed table entries A, the probability of function evaluation $p_F(\infty,A)$ is equal to the probability of discarded evaluation $p_D(\infty,A)$, which can be estimated using the probability of add $p_A(q_f(A),A)$ when the table becomes full.

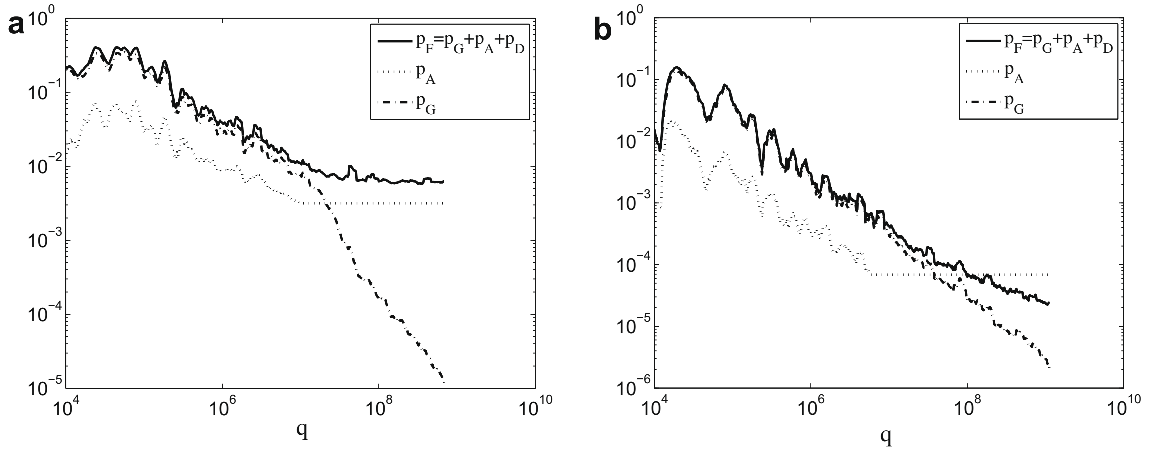


Fig. 14. The probabilities of function evaluation p_F, add p_A, and grow p_G against the number of queries q from PaSR calculations with the skeletal mechanism (a) $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 6 \times 10^4$, (b) $\varepsilon_{tol} = 1 \times 10^{-3}$ and $A = 2 \times 10^3$. In the plots, after the tables are full, to facilitate comparison, the values of p_A (which are zero) are artificially set to be the values at the points where the tables become full. Hence the turning points where the p_A curves become flat indicate the points where the ISAT tables become full.

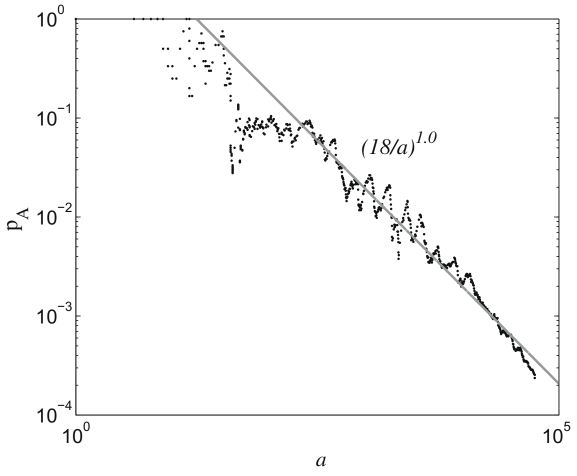


Fig. 15. Probability of add p_A against the number of tabulated table entries a. Symbols: results from the PaSR calculation with the skeletal mechanism and $\varepsilon_{tol} = 2 \times 10^{-4}$; line: the inverse power law $(c/a)^r$ with $c = 18$ and $r = 1.0$.

Fig. 14 shows the probability of function evaluation p_F, the probability of add p_A, and the probability of grow p_G from two PaSR calculations with different settings. As may be seen from the figure, the probability of grow p_G becomes very small for a large number of queries, q. Hence for a very long-run simulation, the probability of function evaluation $p_F(\infty, A)$ is indeed close to the probability of discarded evaluation $p_D(\infty, A)$.

Another important observation is that the quality of the estimation of the probability of function evaluation $p_F(\infty, A)$ using the probability of add $p_A(q_f(A), A)$ is case-dependent. The probability of function evaluation p_F can be overestimated or underestimated by the simple model presented above. Nevertheless, as shown in Section 6, it is sufficient to use this simple model to estimate p_F for a long-run simulation and make the right decisions in the adaptive strategy.

A.2. Power law for p_A

For a given ISAT task with a large number of queries, according to Eq. (A.3), the probability of function evaluation for an infinite number of queries $p_F(\infty, A)$ can be estimated using the probability of add $p_A(q_f(A), A)$. As revealed by the notation, $p_A(q_f(A), A)$ depends only on the number of tabulated table entries. Since A is arbitrary, the function $p_A(q_f(A), A)$ is simply one instance of the general function $p_A(q(a), a)$, or in short, $p_A(a)$, where a is the number of tabulated entries and $q(a)$ is the corresponding number of queries in a simulation. For a given calculation, the functional relation between $p_A(a)$ and a can be easily obtained from the ISAT statistics.

Fig. 15 shows the probability of add, p_A, against the number of tabulated table entries, a, from a PaSR calculation. For sufficiently large a, the probability of add p_A decreases almost monotonically with a (within statistical fluctuation). As shown in the figure, for a sufficiently large number of table entries, the curve of p_A is reasonably well represented by an inverse power law. For the particular case shown, the inverse power is 1. For all the cases investigated in the present study, the observed range of the inverse power is from 0.5 to 1.5. (In other studies [35] where PDF calculations of the oxidation of a premixed methane/air mixture in a PaSR are performed, an inverse power up to 1.7 is observed.) Based on these observations, the functional relation between $p_A(a)$ and a can be approximated by the following inverse power law:

$$p_A(a) \sim (c/a)^r, \tag{A.4}$$

where c and r are problem-dependent constants.

To sum up, for a given table size A, the probability of function evaluation after an infinite number of queries $p_F(\infty, A)$ is approximately equal to the probability of add when the table becomes full, $p_A(q_f(A), A)$. The latter, to a good approximation, can be estimated from the power law in Eq. (A.4).

Appendix B. Preferential distribution (PREF) algorithm

We consider B batches of particles, resident on P processors. (When PREF is used in the adaptive ISAT strategy, P is the number of processors in each group, i.e., g. Otherwise, P is the number of processors in the calculation, i.e., N_p.) There are N_b particles in batch b with $1 \leqslant b \leqslant B$. We define an indicator matrix K of dimension $B \times P$. The components of K have values 0 or 1. Batch b can be processed on processor p only if K_{bp} equals 1. (The components of matrix K are all set to be 1 initially, and are updated subsequently by tracking which processors the unresolved particles have been to.) The task is to assign particle batches to valid processors in such a way that (as nearly as possible) each processor is assigned the same number of particles.

We define the following quantities in the algorithm:

- $\overline{N} = \sum_{b=1}^{B} N_b/P$ is the average number of particles to be assigned to each processor
- M_b is the number of particles in batch b yet to be assigned; initially $M_b = N_b$
- V_p denotes the vacancies on processor p; initially $V_p = \overline{N}$
- $J_{bp} = 1$ if some particles in batch b can be assigned to processor p; initially $J = K$
- $F_p \equiv \sum_{b=1}^{B} J_{bp} M_b/V_p$ is the average number of particles per vacancy that could be assigned to processor p
- $A_b \equiv \sum_{p=1}^{P} J_{bp} V_p/M_b$ is the average number of available vacancies per particle in batch b.

The strategy employed is as follows.

(1) Selection of p. Processor p is first selected such that $F_p \leqslant F_{p'}$ for all p' such that $J_{bp'} = 1$ for some b. A processor with a small value of F_p means that the number of particles eligible to visit this processor is relatively small, and in order to distribute all the particles and at the same time (if possible) guarantee load balancing, it is desirable to assign particles to this processor first.
(2) Selection of b. Batch b is then selected such that $A_b \leqslant A_{b'}$ for all b' such that $J_{b'p} = 1$. A batch with a small value of A_b means that the number of processors eligible for the particles (in the batch) to visit is relatively small, and in order to distribute all the particles and at the same time (if possible) guarantee load balancing, it is desirable to distribute these particles first.

After selection of b and p, M_b, V_p and J_{bp} are updated as follows:

```
if $M_b \leqslant V_p$:
    assign all $M_b$ particles to processor $p$
    $V_p \leftarrow V_p - M_b$, adjust count of vacancies
    $M_b = 0$, no more particles to assign
    $J(b,:) = 0$, no more particles in batch $b$
    if $V_p = 0, J(:,p) = 0$ no more particles can be assigned to $p$
otherwise (i.e., $M_b > V_p$)
    assign $V_p$ particles from batch $b$ to processor $p$
    $M_b = M_b - V_p$, adjust the number of particles yet to be assigned
    $V_p = 0$, no more vacancies
    $J(p,:) = 0$, no more vacancies on processor $p$
```

The process is repeated until the maximum component in J is 0. Notice that for some cases, some particles may be unassigned at the end. In order to assign all of the particles, the vacancy size $\overline{N}$ on each processor is increased. Currently $\overline{N}$ is increased by 8% each time the strategy does not assign all of the particles.

A similar strategy can be defined by selecting b first, and then p (rather than vice versa). It is found that there is no significant difference in terms of performance between these two variants.

One thing worth mentioning is, in the first retrieve attempt, if the number of particles to be resolved is uniform or close to uniform among the processors, the current implementation of the PREF algorithm forces, to the extent possible, the particles to try the first retrieve attempt from their local ISAT tables to reduce the amount of message passing; otherwise if the numbers of particles among the processors are significantly nonuniform, the above algorithm is used to distribute particles uniformly among the processors, and thus the first retrieve attempt for particles is not necessarily made on the local ISAT tables. In current implementation, a case is deemed to be nonuniform if the number of particles on one processor is $\tilde{\alpha}$ (e.g., $\tilde{\alpha} = 2$) times more than the average number of particles among the processors, where $\tilde{\alpha}$ is a user-specified parameter (having default value $\tilde{\alpha} = 2$.)

Appendix C. Fully developed ISAT table

In the current implementation, we deem an ISAT table to be "fully developed" when

$$p_G + p_A < p_{fd} \tag{C.1}$$

with p_{fd} (e.g., $p_{fd} = 10^{-4}$) being a user-specified threshold value. This criterion is based solely on the frequency of the add and grow events, regardless of the computational time spent in different events.

Another viable criterion to be investigated is based on the fraction of time spent in grow and add. Under this criterion, an ISAT table is deemed to be "fully developed" when

$$(p_G + p_A)t_F < \alpha(p_D t_F + p_R t_R), \tag{C.2}$$

i.e.,

$$p_G + p_A < \alpha(p_D + p_R t_R/t_F), \tag{C.3}$$

where α is a user-specified parameter (e.g., $\alpha = 0.1$), t_R and t_F are the average retrieve time and function evaluation time, respectively. The left-hand side of Eq. (C.2) is an estimate for the fraction of time in grow and add, and the term $(p_D t_F + p_R t_R)$ on the right hand side is an estimate of the time in retrieve and discarded evaluation. Hence, by this criterion, the table is "fully developed" when the fraction of time spent in grow and add is less than the (relatively small) user-specified value α.

Appendix D. Algorithm for optimal pairing

We consider the lth stage of the adaptive strategy, in which the simulation has $2 \times N_g$ groups (of processors). Each of the groups i can be paired with one of the other groups j with an estimated wall clock time T'_{ij} for the hypothetical pairing. The objective here is to partition the groups into N_g pairs of new groups, such that this partition minimizes the maximum value of T'_{ij} over all i and j pairs corresponding to new groups.

This problem is closely related to the Maximum Cardinality Matching problem (MCM): Given an undirected (and unweighted) graph consisting of a set of vertices, V, and a set of edges, E, determine the largest possible matching. *A matching* is a set M of edges such that each vertex belongs to at most one member of M. *A matching* is said to be *perfect* if each vertex belongs to exactly one member of M (i.e., all the vertices are used). For the problem we consider here, the set of vertices corresponds to the groups in the simulation, the edges between vertices are hypothetical pairing between different groups, and a matching is to pair each one of the groups in the simulation with exactly one of the other groups.

The graph we have here is weighted with the edge weight corresponding to the estimated wall clock time for the hypothetical pairing and the goal is to find the *perfect matching* M that minimizes the maximum weight among all edges of M. The algorithm used here is proposed by Chew [37] which makes use of the MCM subroutine [38].

(1) Let $G = (V, E)$ be the graph. Sort the edges by weight. The idea is to use binary search on the sorted list of edges. Let w be the middle weight in the sorted list of edges.
(2) Form G', an unweighted graph by retaining only the edges with weight less than w.
(3) Run the MCM algorithm to see if G' has a perfect match. If it does, make w smaller (as in binary search). If it does not, make w larger (as in binary search).
(4) If the binary search range is down to a single weight, report that weight; otherwise go to step (2).

The rationale of the algorithm is to gradually exclude the heavy edges in the graph and then use MCM to determine if a perfect match exists within the remaining edges. The MCM subroutine used runs in time $O(n^3)$, where n is the number of vertices. Thus the above algorithm, based on the MCM subroutine and binary search, runs in time $O(n^3 \log(n))$. It is found that with the algorithm, optimal pairing can be obtained efficiently.

Appendix E. The estimated times in adaptive ISAT strategy

In this appendix, we discuss how the different estimated times required in the adaptive strategy are obtained. In order for this strategy to be effective, it is necessary to construct and compare estimates of wall clock time for the two cases of paring and not pairing the existing groups of processors.

E.1. The estimated time T'_i with no pairing

When there is no pairing, the existing grouping remains unchanged and the ISAT tables continue to develop based on the queries from within the group. During each block sub-step, the particles from each group are distributed among the processor(s) within the group and retrieve attempts are made using the PREF distribution strategy. Those particles that have not been resolved by retrieve attempts are distributed evenly using the URAN strategy within each group and resolved by either grow, add, or discarded evaluation. For a very long-run simulation with the existing grouping, one can estimate the time spent in retrieve attempts and function evaluations separately based on the available information. Then the estimated time per block sub-step in the ith group, T'_i, is just the sum of these two, i.e.,

$$T'_i = N'_i t'_{Ri,w} + N'_{Fi} t'_{Fi,w}, \tag{E.1}$$

where the first term on right hand side represents the contribution from retrieve attempts and the second term represents the contribution from function evaluation. In Eq. (E.1), N'_i is the average number of particles processed on each processor in group i in each block sub-step; N'_{Fi} is the estimated number of particles that need to be evaluated using function evaluation on each processor in group i in each block sub-step; $t'_{Ri,w}$ and $t'_{Fi,w}$ are the average wall clock time per particle spent in retrieve attempts and function evaluations, respectively.

The quantities $N'_i, t'_{Ri,w}, N'_{Fi}$ and $t'_{Fi,w}$ are estimated as follows. The variable N'_i is estimated as

$$N'_i = N_i, \tag{E.2}$$

where N_i is the average number of particles processed on each processor in group i in each block sub-step, i.e.,

$$N_i \equiv (N_{i,1} + N_{i,2} + \cdots + N_{i,k} + \cdots + N_{i,g})/g \tag{E.3}$$

with $N_{i,k}$ being the number of particles on the kth processor in the ith group in each block sub-step. The variable $t'_{Ri,w}$ is estimated as

$$t'_{Ri,w} = t_{Ri,w}, \tag{E.4}$$

where $t_{Ri,w}$ is obtained by dividing the measured wall clock time spent in retrieve attempts by the average number of particles treated per processor, i.e., N_i, during retrieve attempts in group i. Notice that $t_{Ri,w}$ includes not only the CPU time but also the communication and synchronization times. The variable $t'_{Fi,w}$ is estimated as

$$t'_{Fi,w} = \max(t_{Fi,1}, t_{Fi,2}, \cdots t_{Fi,k}, \cdots t_{Fi,g}), \tag{E.5}$$

where t_{Fi} is the average CPU time per function evaluation and is extracted from ISAT statistics during the simulation. The variable N'_{Fi} is estimated as

$$N'_{Fi} = N'_i p'_{Fi}(\infty, a'_i), \tag{E.6}$$

where $p'_{Fi}(\infty, a'_i)$ is the estimation of the probability of function evaluation in group i for a long-run simulation with a'_i table entries. We assume that for the long-run simulation all processors in group i have tabulated the maximum allowed number of entries A. Hence a'_i is given by

$$a'_i = gA. \tag{E.7}$$

From Eq. (A.3), the probability of function evaluation $p_{Fi}(\infty, a'_i)$ can be estimated using

$$p_{Fi}(\infty, a'_i) \approx p_{Ai}(a'_i). \tag{E.8}$$

The probability of add $p_{Ai}(a'_i)$ can be obtained through the power law Eq. (A.4)

$$\frac{p_{Ai}(a'_i)}{p_{Ai}(a_i)} = \left(\frac{a_i}{a'_i}\right)^r, \tag{E.9}$$

where a_i is the total number of tabulated entries in the current pairing stage in group i, and the probability of add $p_{Ai}(a_i)$ is obtained by dividing the counted number of adds by the total number of queries in group i at the step when the number of table entries in group i reaches a_i. The power r observed in the simulations to date is in the range of 0.5 and 1.5 and the value 1 is used in the current implementation. The effect of different values of r is investigated below.

E.2. The estimated time T'_{ij} with pairing

When groups i and group j are paired to form a new group, during each block sub-step, for particles from one group, retrieve attempts can be made using not only their original group's ISAT table(s), but also the one(s) in the other group. Thus the number of retrieve attempts allowed using the PREF distribution strategy doubles. Particles that have not been resolved by retrieves are distributed evenly using the URAN strategy within all the processors in the new group and resolved by function evaluations (through the events grow, add, or discarded evaluation). For a very long-run simulation, with this new grouping, we similarly estimate the time spent in retrieve attempts and function evaluations separately, and the estimated time per block sub-step, T'_{ij}, is the sum of these two, i.e.,

$$T'_{ij} = N'_{ij} t'_{Rij,w} + N'_{Fij} t'_{Fij,w} \tag{E.10}$$

where N'_{ij} represents the estimated average number of particles processed in retrieve attempts per processor when groups i and j are paired in each block sub-step; N'_{Fij} is the estimated number of particles per processor that need to be resolved by function evaluation in each block sub-step; and $t'_{Rij,w}$ and $t'_{Fij,w}$ are the estimated average wall clock times per particle spent in retrieve attempts and function evaluation when groups i and j are paired, respectively.

In the current implementation, N'_{ij} is obtained as

$$N'_{ij} = (N_i + N_j)/2, \tag{E.11}$$

which is the average number of particles processed on each processor when groups i and j are paired; and $t'_{Fij,w}$ is obtained as

$$t'_{Fij,w} = \max(t'_{Fi,w}, t'_{Fj,w}), \tag{E.12}$$

where $t'_{Fi,w}$ and $t'_{Fj,w}$ are obtained through Eq. (E.5).

The estimations of N'_{Fij} and $t'_{Rij,w}$ are more subtle and are elaborated in the following.

E.2.1. The estimate N'_{Fij} of the number of function evaluations required

When groups i and j are paired to form a new group, for a long-run simulation, we denote by a'_i and a'_j the total number of table entries tabulated in the new group based on particles from group i and group j, respectively, regardless of whether these entries are in the ISAT tables of group *i* or group *j*. For example, the table entries based on compositions from group *i* are not necessarily tabulated on the processors in the former group *i*, and may be tabulated on the processors in the former group *j*. We assume

(1) that a'_i and a'_j are proportional to a_i and a_j, respectively, i.e.,

$$\frac{a'_i}{a'_j} = \frac{a_i}{a_j}, \tag{E.13}$$

where a_i and a_j are the numbers of table entries in group *i* and group *j*, respectively, immediately prior to the current pairing stage.

(2) that the maximum allowed number of table entries in the new group is used in a long-run simulation, i.e.,

$$a'_i + a'_j = 2gA, \tag{E.14}$$

where $2g$ is the number of processors in the new group.

With these two assumptions, we have

$$a'_i = 2gA\frac{a_i}{a_i + a_j} \tag{E.15}$$

and

$$a'_j = 2gA\frac{a_j}{a_i + a_j}. \tag{E.16}$$

When groups *i* and *j* are paired to form a new group, whether or not particles from one group (say group *i*) are able to be retrieved using the table entries added based on compositions from the other group (say group *j*) depends on the composition distributions $\mathcal{D}(\mathbf{x})$ on these two groups. For example, if the particle compositions in groups *i* and *j* are disjoint, then the table entries added based on compositions from group *j* are useless for retrieving particle compositions from group *i*. Let $\tilde{a}_i$ denote the *effective* number of tabulated table entries that are useful for retrieving particle compositions from group *i* when groups *i* and *j* are paired to form a new group. If the particle compositions in groups *i* and *j* are disjoint then

$$\tilde{a}_i = a'_i; \tag{E.17}$$

at the other extreme, if they are coincident then

$$\tilde{a}_i = a'_i + a'_j. \tag{E.18}$$

In general, we model $\tilde{a}_i$ as

$$\tilde{a}_i = a'_i + a'_j\frac{P_{ij}}{\max(P_{ij}, P_{jj})}, \tag{E.19}$$

where P_{ij} is the probability that a particle from group *i* can be resolved by retrieve from the ISAT table(s) in group *j*, and P_{jj} is the probability that a particle from group *j* can be resolved by retrieve from the ISAT table(s) in group *j*. Both P_{ij} and P_{jj} are obtained from the statistics in the current pairing stage. (Notice that for coincident cases, P_{ij} may be far less than one due to the small number of table entries allowed. Simply using $\tilde{a}_i = a'_i + a'_jP_{ij}$ may underestimate the effective number of table entries $\tilde{a}_i$ available for group *i*. The above normalization in Eq. (E.19) is expected to provide a reasonable prediction of $\tilde{a}_i$.) Similarly, we model $\tilde{a}_j$ as

$$\tilde{a}_j = a'_j + a'_i\frac{P_{ji}}{\max(P_{ji}, P_{ii})}. \tag{E.20}$$

Then for a long-run simulation in which groups *i* and *j* are paired, with Eq. (A.3) and the power law in Eq. (A.4), the probability of a group *i* particle composition requiring function evaluation is

$$p'_{Fi}(\infty, \tilde{a}_i) = p_{Ai}(a_i)\left(\frac{a_i}{\tilde{a}_i}\right)^r, \tag{E.21}$$

where a_i is the number of tabulated entries immediately prior to the current pairing stage in group *i*, and the current probability of add, $p_{A_i}(a_i)$, is estimated by dividing the counted number of adds by the total number of queries in group *i* at the step when the number of tabulated table entries in group *i* reaches a_i. Similarly,

$$p'_{Fj}(\infty, \tilde{a}_j) = p_{Aj}(a_j)\left(\frac{a_j}{\tilde{a}_j}\right)^r. \tag{E.22}$$

The unresolved particles after the retrieve attempts should be evenly distributed among the two groups' processors and resolved by function evaluation, thus N'_{Fij} can be estimated as

$$N'_{Fij} = \frac{N_i p'_{Fi}(\infty, \tilde{a}_i) + N_j p'_{Fj}(\infty, \tilde{a}_j)}{2}, \tag{E.23}$$

where N_i and N_j are the average number of particles per processor on group i and group j, respectively.

E.2.2. The estimate $t'_{Rij,w}$ of the retrieving time

When groups i and j are paired, retrieve attempts can be made for particles using not only their original group's ISAT table(s) but also the one(s) paired with the group. The average wall clock time per particle spent in retrieve attempts, $t_{Rij,w}$, for a long-run simulation after pairing is difficult to estimate. There is no straightforward functional relation between $t_{Rij,w}$ and $t_{Ri,w}$ and $t_{Rj,w}$, where $t_{Ri,w}$ and $t_{Rj,w}$ are the measured average wall clock times per particle spent in retrieve attempts for particles within group i and within group j in the current pairing stage, respectively.

To illustrate the complicated relationship between $t_{Ri,w}, t_{Rj,w}$ and $t_{Rij,w}$, we perform the following PaSR tests. For each PaSR test, two initially identical simulations (each having two groups with one processor in each group) run until the criterion for pairing is reached. Then the two simulations, one with forced pairing between these two groups and the other one with no pairing, continue to run for a sufficiently long time. At the same time, the wall clock time spent in retrieve attempts is measured, and by normalizing the wall clock time by the average number of particles processed, the average retrieve time per particle can be obtained. Fig. 16 shows measurements of $t_{Ri,w}, t_{Rj,w}$ and $t_{Rij,w}$ from two uniform PaSR tests, one being coincident and the other one being disjoint. The figure shows only the results subsequent to the pairing/no-pairing branch point. As may be seen from the figure, for the coincident case, $t_{Rij,w}$ is slightly larger (about 25%) than $t_{Ri,w}$ and $t_{Rj,w}$. However for the disjoint case, $t_{Rij,w}$ is much larger (about 60%) than $t_{Ri,w}$ and $t_{Rj,w}$.

Hence when groups i and j are paired, the retrieving processes among the processors in these two groups are nontrivial and consequently $t_{Rij,w}$ is difficult to estimate. Currently, we estimate $t_{Rij,w}$ by modelling the retrieving processes through the following two-step process.

(1) In the first step, retrieve attempts are made on half of the particles from one group (say group i) and half of the particles from the other group (say group j), using the ISAT tables in the group (say group i). The non-local particles are sent by message passing. Specifically, in the first step, the number of particles that need to be processed by each processor in group i and group j is

$$N_{Ri,1} = N_{Rj,1} = \frac{N_i + N_j}{2}. \tag{E.24}$$

The computational time spent in this step in group i can be estimated as

$$I_i = \frac{N_i}{2} t_{Ri,w} + \frac{N_j}{2} t_{Rj \to i,w} + \frac{N_j}{2} t_C, \tag{E.25}$$

where $t_{Rj \to i,w}$ is the measured average wall clock time per particle for particles from group j spent in retrieve attempts in group i which is obtained from statistics in the current pairing stage, and t_C is the average two-way communication time per particle. Similarly for I_j, we have

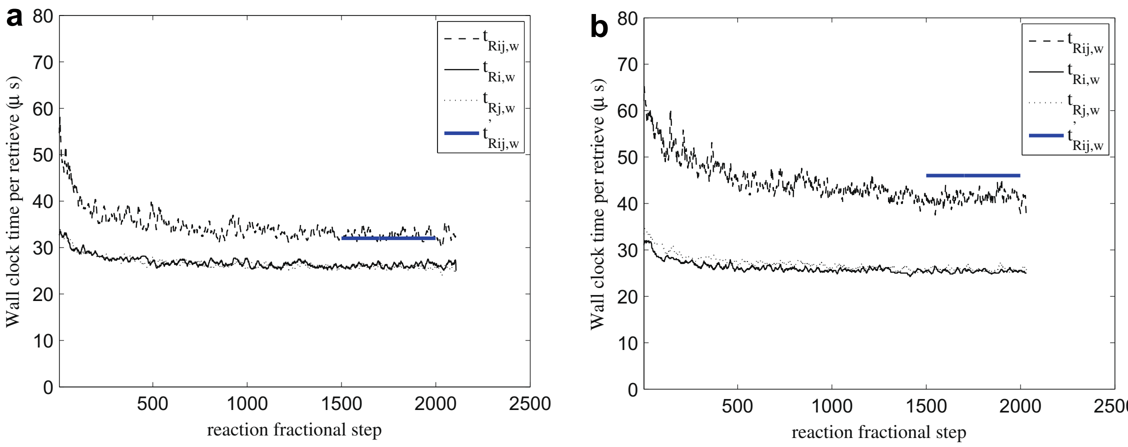


Fig. 16. Wall clock time per particle in retrieve attempts from the uniform PaSR calculations with $N_p = 2, \varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 2 \times 10^3$; (a) coincident case (b) disjoint case. Black dashed line: $t_{Rij,w}$, measured wall clock time for the case with group i and j being paired; black solid line: $t_{Ri,w}$, measured wall clock time for group i without pairing performed; black dot line: $t_{Rj,w}$, measured wall clock for group j with no pairing performed; horizontal solid line: $t'_{Rij,w}$, the estimated wall clock time per particle in retrieve attempts for the hypothetical pairing between group i and j. Each calculation results in an average of 1.0×10^7 queries per processor. The figure shows only the results subsequent to the paring/no-pairing branch point.

$$I_j = \frac{N_j}{2} t_{Rj,w} + \frac{N_i}{2} t_{Ri\to j,w} + \frac{N_i}{2} t_C. \tag{E.26}$$

(2) In the second step, retrieve attempts are made on the unresolved particles from the first step based on the ISAT tables in the other group. Hence in this step each group in the potential pairing needs to process the unresolved particles from the other group. The number of particles treated per processor in group i in this step is

$$N_{Ri,2} = \frac{N_i}{2}(1 - P_{ij}) + \frac{N_j}{2}(1 - P_{jj}), \tag{E.27}$$

where P_{ij} is the probability that a particle composition from group i can be retrieved from ISAT table(s) in group j. Similarly for $N_{Rj,2}$ we have

$$N_{Rj,2} = \frac{N_j}{2}(1 - P_{ji}) + \frac{N_i}{2}(1 - P_{ii}). \tag{E.28}$$

The computational time spent in this step on group i can be estimated as

$$II_i = \frac{N_i}{2}(1 - P_{ij}) t_{Ri,w} + \frac{N_j}{2}(1 - P_{jj}) t_{Rj\to i,w} + \frac{N_j}{2}(1 - P_{jj}) t_C, \tag{E.29}$$

where the first two terms on the right hand side are the contribution from the retrieve attempts and the last term is the message passing time. Similarly, we have

$$II_j = \frac{N_j}{2}(1 - P_{ji}) t_{Rj,w} + \frac{N_i}{2}(1 - P_{ii}) t_{Ri\to j,w} + \frac{N_i}{2}(1 - P_{ii}) t_C. \tag{E.30}$$

The average wall clock time per particle spent in retrieve attempts, $t'_{Rij,w}$, can be obtained as

$$t'_{Rij,w} = \frac{\max(I_i, I_j) + \max(II_i, II_j)}{(N_i + N_j)/2}. \tag{E.31}$$

Also shown in Fig. 16 are $t'_{Rij,w}$, the estimated wall clock times per particle in retrieve attempts for the hypothetical pairing between group i and j based on the above two-step empirical procedure for the two uniform cases. As may be seen, for both the coincident and disjoint cases considered, there is good agreement between the measured and estimated wall clock time in retrieve attempts when two groups are paired. For the test cases considered, the above two-step procedure to compute $t'_{Rij,w}$ provides a sufficiently good estimate for $t_{Rij,w}$.

It is worth mentioning that the above two-step process is the first attempt to model the complicated retrieving processes among the processors in two groups when they are paired. When computational particles among the processors are significantly nonuniform, during each retrieve attempt, PREF distributes the unresolved particles uniformly among the processors in the two groups; and to a good approximation, half of the unresolved particles from one group are processed by local ISAT tables and the other half are processed by remote ISAT tables on the processors in the other group. Therefore for the highly nonuniform cases, the retrieve attempts are realistically (to some extent) represented by the above two-step process. However for uniform (or close to uniform) cases, PREF forces particles to attempt retrieving from the local table first, and the assumption in the two-step procedure "half of the particles from one group are processed by remote ISAT tables in the other group" is far from realistic. Nevertheless, while not realistic, as shown in Fig. 16, for the uniform test cases considered, the two-step procedure gives a sufficiently good estimate of $t_{Rij,w}$. Further exploring and understanding the complicated retrieving processes among the processors in two groups is certainly a fruitful direction for further study.

E.3. Appraisal of the estimates T'_p and T'_{np}

In this subsection, we appraise the above approaches for estimating T_p and T_{np}, which are the wall clock times per block sub-step when there is pairing and no pairing, respectively, by making measurements in long-run simulations. We perform the following PaSR tests. For each PaSR test, two initially identical simulations (each having two groups with one processor in each group) run until the criterion for pairing is reached. We then estimate T_p and T_{np} based on the approaches described in Section 6.2. Then the two simulations, one with forced pairing between these two groups and the other one with no pairing, continue to run for a sufficiently long time. Based on these two long runs, T_p and T_{np} are measured. The estimates are then appraised by comparing the estimated times T'_p and T'_{np} with the measured times T_p and T_{np}, respectively.

It should be appreciated that the estimates T'_p and T'_{np} are used solely to make the binary decision of whether to pair or not. Consequently, it is not essential for the estimates to be quantitatively accurate. Rather, it is important that if T_p is significantly greater or less than T_{np}, then, correspondingly, T'_p is greater or less than T'_{np}. In this way, the correct pairing decision is made in the circumstances where the decision makes a significant difference.

Fig. 17 shows the measured and estimated wall clock times in the reaction fractional step from two uniform PaSR calculations, one being coincident and the other one being disjoint. The figure shows the results only after pairing or not (i.e., not prior to the pairing criterion being satisfied). As may be seen, for the coincident case, the estimated quantities T'_p and T'_{np} are lower than the measured ones. The main reason is that the estimated percentages of function evaluation are lower than the

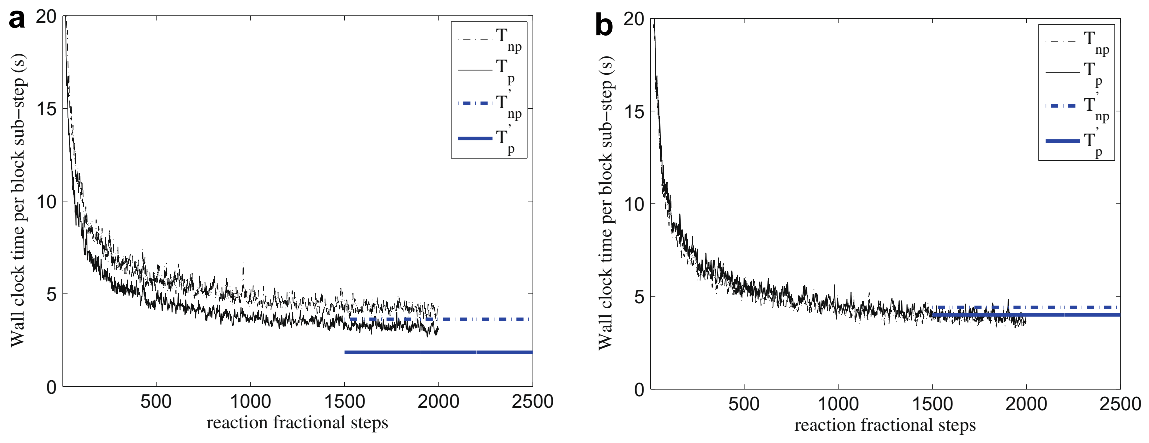


Fig. 17. Wall clock time per block sub-step from the uniform PaSR calculations with $N_p = 2$, $\varepsilon_{tol} = 1 \times 10^{-4}$, and $A = 2 \times 10^3$; Top plot: coincident case; bottom plot: disjoint case. Black dashed line: T_{np}, measured wall clock time (per block sub-step) with no pairing; black solid line: T_p, measured wall clock time with pairing; horizontal dashed dot line: T'_{np}, estimated wall clock time with no pairing; horizontal solid line: T'_p, estimated wall clock time with pairing. Figure shows the results only after pairing or not (i.e., not prior to the pairing criterion being satisfied).

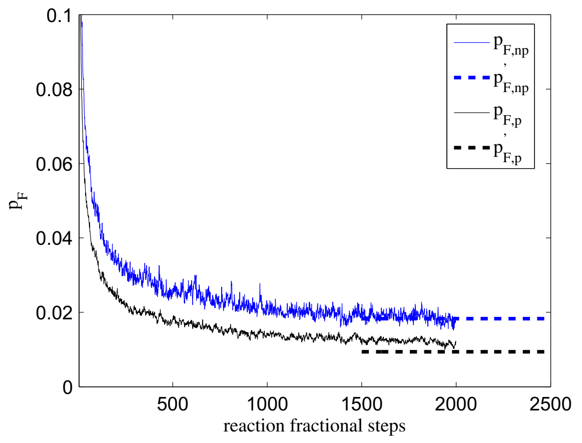


Fig. 18. The estimated and measured percentages of function evaluations (averaged over processors) from the uniform coincident PaSR calculations with no pairing and pairing. In the calculations, $N_p = 2$, $\varepsilon_{tol} = 1 \times 10^{-4}$, and $A = 2 \times 10^3$. Blue solid line: $p_{F,np}$, measured with no pairing; blue (horizontal) dashed line: $p'_{F,np}$, estimated with no pairing; black solid line: $p_{F,p}$, measured with pairing; black (horizontal) dashed line: $p'_{F,p}$, estimated with pairing. Figure shows the results only after pairing or not (i.e., not prior to the pairing criterion being satisfied).

Table E.1
Effect of r on estimated wall clock time(s) per block sub-step from uniform coincident PaSR calculation with $\varepsilon_{tol} = 1 \times 10^{-4}$ and $A = 2 \times 10^3$.

	$T'_{np}(s)$	$T'_p(s)$
Measured	4.2	3.1
Estimated ($r = 0.5$)	4.2	2.9
Estimated ($r = 1.0$)	3.0	1.6
Estimated ($r = 1.5$)	2.1	0.9

observed ones. This is confirmed in the Fig. 18 which shows the estimated and measured percentages of function evaluations (averaged over the processors) for both pairing and no pairing. Nevertheless, the estimation procedure correctly predicts that pairing leads to significantly reduced wall clock time (compared to not pairing).

For the disjoint case, there is good agreement between the estimated and measured times. For this particular case, the decision on whether to pair or not is not significant because the calculations with and without pairing achieve comparable computational performance.

Recall that in the adaptive strategy, one key ingredient is the power law, i.e., Eq. (A.4), which is used to estimate the percentage of function evaluations for a long-run simulation. The exponent r in the power law is case-dependent and is not known *a priori*. For all the cases investigated in the present study, the observed range of the exponent is from 0.5 to 1.5. In the current implementation of the adaptive strategy, we fix r to be 1.

Table E.1 shows the measured and estimated (with different values of r) wall clock times per block sub-step from a uniform coincident PaSR calculation. As may be seen from the table, a particular specification of r affects the estimates of T_p and T_{np}, but nevertheless it does not affect the decisions on pairing in the adaptive strategy: for all the values of r, T_{np} is larger than T_p and the adaptive strategy makes the correct pairing decision.

References

[1] S.B. Pope, Computationally efficient implementation of combustion chemistry using *in situ* adaptive tabulation, Combustion Theory and Modelling 1 (1997) 41–63.
[2] L. Lu, S.B. Pope, An improved algorithm for *in situ* adaptive tabulation, Journal of Computational Physics 228 (2009) 361–386.
[3] 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, V.V. Lissianski, Z. Qin, <http://www.me.berkeley.edu/gri_mech/>.
[4] H.J. Curran, P. Gaffuri, W.J. Pitz, C.K. Westbrook, A comprehensive modeling study of iso-octane oxidation, Combustion and Flame 129 (2002) 253–280.
[5] U. Maas, S.B. Pope, Simplifying chemical-kinetics: intrinsic low-dimensional manifolds in composition space, Combustion and Flame 88 (1992) 239–264.
[6] Z. Ren, S.B. Pope, The geometry of reaction trajectories and attracting manifolds in composition space, Combustion Theory and Modelling 10 (2006) 361–388.
[7] J.-Y. Chen, W. Kollmann, R.W. Dibble, PDF modelling of turbulent nonpremixed methane jet flames, Combustion Science and Technology 64 (1989) 315–346.
[8] T. Turányi, Parameterization of reaction mechanisms using orthonormal polynomials, Computers and Chemistry 18 (1994) 45–54.
[9] F.C. Christo, A.R. Masri, E.M. Nebot, Artificial neural network implementation of chemistry with PDF simulation of H_2/CO_2 flames, Combustion and Flame 106 (1996) 406–427.
[10] J.A. Blasco, N. Fueyo, C. Dopazo, J. Ballester, Modelling the temporal evolution of a reduced combustion chemical system with an artificial neural network, Combustion and Flame 113 (1998) 38–52.
[11] S.R. Tonse, N.W. Moriary, N.J. Brown, M. Frenklach, PRISM: piecewise reusable implementation of solution mapping an economical strategy for chemical kinetics, Israel Journal of Chemistry 39 (1999) 97–106.
[12] J.B. Bell, N.J. Brown, M.S. Day, M. Frenklach, J.F. Grcar, R.M. Propp, S.R. Tonse, Scaling and efficiency of PRISM in adaptive simulations of turbulent premixed flames, Proceedings of the Combustion Institute 28 (2000) 107–113.
[13] H. Rabitz, Ö. Alis, General foundations of high-dimensional model representations, Journal of Mathematical Chemistry 25 (1999) 197–233.
[14] S.B. Pope, PDF methods for turbulent reactive flows, Progress in Energy and Combustion Science 11 (1985) 119–192.
[15] J. Xu, S.B. Pope, PDF calculations of turbulent nonpremixed flames with local extinction, Combustion and Flame 123 (2000) 281–307.
[16] Q. Tang, J. Xu, S.B. Pope, PDF calculations of local extinction and NO production in piloted-jet turbulent methane/air flames, Proceedings of the Combustion Institute 28 (2000) 133–139.
[17] M. Embouazza, D.C. Haworth, N. Darabiha, Implementation of detailed chemical mechanisms into multidimensional CFD using in situ adaptive tabulation: application to HCCI engines, SAE Paper 2002-01-2773, 2002.
[18] R.L. Gordon, A.R. Masri, S.B. Pope, G.M. Goldin, A numerical study of auto-ignition in turbulent lifted flames issuing into a vitiated co-flow, Combustion Theory and Modelling 11 (3) (2007) 351–376.
[19] B.J.D. Liu, S.B. Pope, The performance of in situ adaptive tabulation in computations of turbulent flames, Combustion Theory and Modelling 9 (2005) 549–568.
[20] R. Cao, S.B. Pope, The influence of chemical mechanisms on PDF calculations of nonpremixed piloted jet flames, Combustion and Flame 143 (2005) 450–470.
[21] L. Lu, Z. Ren, V. Raman, S.B. Pope, H. Pitsch, LES/FDF/ISAT computations of turbulent flames, in: Proceedings of CTR Summer Program 2004, Center For Turbulence Research, (2004) 283–294.
[22] L. Lu, Z. Ren, S.R. Lantz, V. Raman, S.B. Pope, H. Pitsch, Investigation of strategies for the parallel implementation of ISAT in LES/FDF/ISAT computations, in: Fourth Joint Meeting of the US Sections of the Combustion Institute, Drexel University, Philadelphia, PA, March 20–23, 2005.
[23] M.A. Singer, S.B. Pope, Exploiting ISAT to solve the reaction-diffusion equation, Combustion Theory and Modelling 8 (2004) 361–383.
[24] M.A. Singer, S.B. Pope, H.N. Najm, Modeling unsteady reacting flow with operator-splitting and ISAT, Combustion and Flame 147 (2006) 150–162.
[25] L.G. Wang, R.O. Fox, Application of in situ adaptive tabulation to CFD simulation of nano-particle formation by reactive precipitation, Chemical Engineering Science 58 (2003) 4387–4401.
[26] J.D. Hedengren, T.F. Edgar, *In situ* adaptive tabulation for real-time control, Industrial Engineering Chemistry Research 44 (2005) 2716–2724.
[27] A. Varshney, A. Armaou, Multiscale optimization using hybrid PDE/kMC process systems with application to thin film growth, Chemical Engineering Science 60 (2005) 6780–6794.
[28] E.R. Hawkes, R. Sankaran, J.C. Sutherland, J.H. Chen, Scalar mixing in direct numerical simulations of temporally evolving plane jet flames with skeletal CO/H_2 kinetics, Proceedings of the Combustion Institute 31 (2007) 1633–1640.
[29] S.M. Correa, Turbulence-chemistry interactions in the intermediate regime of premixed combustion, Combustion and Flame 93 (1993) 41–60.
[30] S.M. Correa, M.E. Braaten, Parallel simulations of partially stirred methane combustion, Combustion and Flame 94 (1993) 469–486.
[31] J.-Y. Chen, Stochastic modeling of partially stirred reactors, Combustion Science and Technology 122 (1997) 63–94.
[32] B. Yang, S.B. Pope, An investigation of the accuracy of manifold methods and splitting schemes in the computational implementation of combustion chemistry, Combustion and Flame 112 (1998) 16–32.
[33] S. James, M.S. Anand, M.K. Razdan, S.B. Pope, In situ detailed chemistry calculations in combustor flow analyses, Journal of Engineering for Gas Turbines and Power 123 (2001) 747–756.
[34] M. Caracotsios, W.E. Stewart, Sensitivity analysis of initial-value problems with mixed ODEs and algebraic equations, Computers and Chemical Engineering 9 (1985) 359–365.
[35] L. Lu, S.B. Pope, A systematic investigation of in situ adaptive tabulation for combustion chemistry. Cornell University Report, FDA 07-01, 2007.
[36] http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html.
[37] L.P. Chew, Private Communication.
[38] ftp://dimacs.rutgers.edu/pub/challenge/graph/.