A Systematic Investigation of In Situ Adaptive Tabulation for Combustion Chemistry

Liuyan Lu, Stephen B. Pope

FDA 07-01 October 2007

Abstract

A systematic up-to-date investigation of different implementations of the *in situ* adaptive tabulation algorithm for chemistry calculations is performed through PDF calculations of non-premixed methane/air combustion in a partially stirred reactor. A parametric study is performed for the new implementation called ISAT5. In particular, the effects of the key parameters in the augmentations to ISAT5, such as affine space, ellipsoids of inaccuracy (EOI), and error checking and correction (ECC), are investigated. The study shows that EOI and ECC are very effective in controlling the incurred local error (or the retrieve error). For a given user-specified error tolerance ε_{tol}, the new implementation ISAT5 significantly reduces the incurred error compared to its predecessor, denoted as ISAT4. Hence to achieve the same incurred error, one can specify a much larger value of ε_{tol} in ISAT5 than in ISAT4. The study highlights that comparison of the computational performance of different implementations of ISAT must be made at a fixed incurred error. To achieve a given incurred errors, ISAT5 is advantageous compared to ISAT4, both in the computational efficiency and in table storage requirements. This significantly alleviates both the computational constraint and the storage constraint in applying ISAT to computationally challenging reactive flows.

Keywords: Turbulent combustion, ISAT algorithm, combustion chemistry, affine space, error checking and correction

1 Introduction

A major challenge in calculations of turbulent reactive flows is that typical detailed combustion mechanisms involve tens or hundreds of species, hundreds or thousands of reactions, and a wide range of time scales. The system of ordinary differential equations governing chemical reactions is thus large and extremely stiff, making the task of solving these equations computationally expensive. It is practically significant to develop methodologies that radically decrease the computational burden imposed by the direct use of detailed chemical kinetics in reactive flow calculations. Among several different types of such methodologies, one approach currently widely used is storage/retrieval methods, which substantially reduce the amount of chemical-kinetic computations required in a reactive flow calculation.

Several storage/retrieval methods have been developed in the past including structured look-up tabulation [1], repro-modelling [2], artificial neural networks (ANN) [3,4], *in situ* adaptive tabulation (ISAT) [5], piecewise reusable implementation of solution mapping (PRISM) [6,7], and high dimension model representations (HDMR) [8]. Among the existing approaches, the ISAT algorithm [5] is currently particularly fruitful. ISAT has been widely used to incorporate detailed or reduced chemical mechanisms in modelling reactive flows: examples are probability density function (PDF) [9] calculations of turbulent non-premixed flames [10–14], and large eddy simulations and direct numerical simulations of reactive flows [15–18]. ISAT is also used in the transported PDF method implemented in Fluent which has been used by Gordon et al. [19]. It has been shown to speed up the chemistry calculations by up to a factor of 1000 in the PDF calculations of turbulent flames [5]. Even for the challenging case of unsteady laminar flames, an overall speed-up factor of approximately 7.5 for chemistry has been demonstrated by Singer et al. [17,18]. Besides the wide applications in the field of combustion, the applications of ISAT in other areas have been reported

in [20–22].

Since its original development in the mid 1990s by Pope [5], the ISAT algorithm has been significantly improved in terms of accuracy, efficiency and parallel implementation, and variants have been proposed [15,16,23–26]. The performance of ISAT in accuracy, computational efficiency, and storage requirement depends on the particulars of the implementation. While the computational savings using ISAT are well demonstrated and reported in the literature, the effects of different implementations of ISAT on performance are not well studied and fully understood. Some existing studies include the investigation of ISAT accuracy in terms of controlling both local and global errors by Liu and Pope [27], and an analysis of the ISAT computational performance by Chen [24].

Recently, a new implementation of ISAT [28], denoted as ISAT5, is developed by Pope to succeed the previous implementation, denoted as ISAT4. In ISAT5, new algorithms, such as affine space, ellipsoid of inaccuracy, and error checking and correction are incorporated to improve the accuracy, efficiency, and the storage requirement of the ISAT algorithm. The purpose of this investigation is to provide a systematic parametric study of the key parameters in ISAT5, and to perform an up-to-date investigation of the relative performance of different implementations of ISAT. For simplicity, the reacting flow chosen for this study is nonpremixed methane/air combustion in idealized partially stirred reactor (PaSR).

The paper is organized as follows. The PDF calculation of the nonpremixed methane/air combustion in the PaSR is outlined in Section 2. Then a brief description of the ISAT algorithm is provided in Section 3, followed by its recent augmentations given in Section 4. In Section 5, a parametric study of the key parameters in ISAT5 is performed. We present the comparison of different implementations of ISAT in Section 6. Conclusions are drawn in Section 7.

2 PaSR test case

To investigate the performance of different implementations of ISAT, we consider the PDF calculations of the combustion of CH_4/air mixtures in an adiabatic, isobaric, partially stirred reactor (PaSR) (with the pairwise mixing model), which is used previously by Correa [29] and Pope [5]. Due to its simplicity, the PaSR has been widely used to investigate combustion models and numerical algorithms [5,29–33]. It is similar to a single grid cell embedded in a large PDF computation of turbulent combustion.

In the stochastic simulation of the PaSR based on Monte Carlo methods, at any time t, the PaSR consists of an even number N of particles. At time t, the thermochemical composition of the ith particle is represented by the $n_\phi = n_s + 1$ variables $\boldsymbol{\phi}^i(t)$, which are taken to be the n_s species specific moles (mass fractions over molecular weights) and the sensible enthalpy of the mixture. As described in [5], "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 mixing and reaction fractional steps. The particles are arranged in pairs: particles 1 and 2, 3 and 4, $\cdots$, $N-1$ and N are partners." 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 the 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 chemical kinetics.

At the discrete times $k\Delta t$, with τ_{res} being the specified residence time, 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, a specified number N_{sub} of substeps of duration $\Delta t_{sub} = \Delta t/N_{sub}$ is taken for reaction and mixing. During each substep of duration Δt_{sub}, the composition evolves by one mixing step of $\Delta t_{sub}/2$, followed by one reaction step of Δt_{sub}, and then by another mixing step of $\Delta t_{sub}/2$.

The values of the parameters used in the test calculation are given in Table 1. There are three inflowing streams: air (79% N_2, 21% O_2) at 300 K; methane at 300 K; and a pilot stream consisting of an equilibrium, stoichiometric fuel/air mixture at 2600 K. The mass flow rates of these streams are in the ratio 0.85:0.1:0.05. Initially (t=0), all particle compositions are set to be the pilot-stream composition. The pressure is atmospheric throughout. Two sets of chemical mechanisms are used for describing the combustion of methane/air. One is a skeletal mechanism consisting of 16 species (the same as that used in [34]), and the other is the more complex GRI3.0 mechanism which involves 53 species[35].

3 Overview of the ISAT algorithm

The *in situ* adaptive tabulation algorithm (ISAT) introduced by Pope [5] is a storage and retrieval methodology. 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. This section outlines the essential ideas of the original ISAT

Table 1

Specified parameters in the PaSR tests

Number of particles	N	100
Time step	Δt	0.1 ms
No. of sub-steps	N_{sub}	3
Substep	Δt_{sub}	0.033 ms
Residence time	τ_{res}	10 ms
Mixing time scale	τ_{mix}	1 ms
Pairing time scale	τ_{pair}	1 ms

algorithm [5]. The detailed description of the ISAT algorithm is given in [5].

Consider the application of ISAT for combustion chemistry computation in PDF calculations of the combustion process in an isobaric PaSR. At time t, the thermochemical composition of the ith particle is represented by the $n_\phi = n_s + 1$ variables $\boldsymbol{\phi}^i(t)$. 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 $\mathbf{R}(\boldsymbol{\phi}^0) \equiv \boldsymbol{\phi}(t_0 + \Delta t_{sub})$, which is the solution to Eq. 4 after a time-step Δt_{sub} from the initial condition $\boldsymbol{\phi}^0 = \boldsymbol{\phi}(t_0)$ at time t_0. Here for simplicity, Δt_{sub} 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 sub-step, $\boldsymbol{\phi}^0$, and $\mathbf{f}$ is the particle composition after the sub-step, i.e., the reaction mapping

$\mathbf{R}(\boldsymbol{\phi}^0) = \boldsymbol{\phi}(t_0 + \Delta t_{sub})$. Thus n_x and n_f are both vectors of length $n_s + 1$.

ISAT uses the ODE solver DDASAC 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.,

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

where $\mathbf{B}$ is a scaling matrix [5].

In addition to the tabulation point $\mathbf{x}$, the reaction mapping $\mathbf{f}$, and the mapping gradient 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 composition $\mathbf{x}^q$, 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 evaluate the corresponding function $\mathbf{f}(\mathbf{x}^q)$.

- Retrieve. If the query point falls within the ellipsoid of accuracy (EOA) of $\mathbf{x}$, the 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 direct function evaluation is performed to determine $\mathbf{f}(\mathbf{x}^q)$ which is 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 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.
- Direct evaluation (DE). If, however, the computed error is larger than ε_{tol} and the table is full, then the reaction mapping is returned without further action. This outcome is called a direct evaluation.

When ISAT is employed for combustion chemistry calculations, the computational efficiency is achieved through obtaining the reaction mapping using retrieves whenever possible, which is typically several orders of magnitude computationally cheaper than the direct function evaluation. 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.

3.1 Error tolerance and incurred local error

As described above, given a query point $\mathbf{x}^q$ falling within an existing ellipsoid of accuracy (EOA), the algorithm uses a piecewise linear approximation to approximate $\mathbf{f}(\mathbf{x}^q)$. The (scaled) difference between the exact function value, $\mathbf{f}(\mathbf{x}^q)$, and the linearized approximation, $\mathbf{f}^l(\mathbf{x}^q)$, is known as the

"retrieve error", or the "incurred local error". The incurred local error is a measure of the numerical accuracy achieved by ISAT. Effective control of the incurred local error is essential to ensure that the chemistry calculation evaluated by ISAT is numerically accurate (for sufficiently small values of the ISAT error tolerance, ε_{tol}).

As described above, in the ISAT algorithm, if a query point $\mathbf{x}^q$ does not fall within an existing EOA, but has error ε less than ε_{tol}, then the corresponding EOA is grown. Since the ROA may be non-convex, growing may cause the EOA to include inaccurate regions. As reported in [27], the growth of the EOAs may incur large local error (greater than ε_{tol}) with small probability.

The error incurred on the many retrieves has a distribution. The local incurred error ε for a large number of retrieves can be characterized by the cumulative distribution function (CDF) of the local error

$$F(y) \equiv Prob\{\varepsilon < y\}, \tag{7}$$

where y is the sample-space variable corresponding to ε. The CDF $F(y)$ provides a complete characterization of the local error. From $F(y)$, various statistics can be extracted such as the error statistics ε_α defined as

$$\alpha = F(\varepsilon_\alpha) = Prob\{\varepsilon < \varepsilon_\alpha\}, \text{for } 0 < \alpha < 1, \tag{8}$$

i.e. with probability α, the error is less than ε_α. Thus, by definition, for 90% of the queries, ε is less than the 90th percentile error $\varepsilon_{0.9}$. In the following, we study the statistics of $F(y)$ normalized using $\varepsilon_{ref} \equiv \varepsilon_{0.9}$. Normalization by ε_{ref} allows a comparison of the shapes of the CDFs obtained with different values of ε_{tol} and different implementations of ISAT.

3.2 Effect of the allowed storage on ISAT performance

For a given calculation of a reactive flow with a given ISAT error tolerance, ε_{tol}, and a given ISAT implementation, the allowed storage for ISAT may have a significant effect on the fractions of different events (e.g., retrieve, direct evaluation) and therefore on the ISAT performance. Let $p_R(Q, A)$, $p_G(Q, A)$, $p_A(Q, A)$ and $p_D(Q, A)$ denote the fractions of the events retrieve, grow, add and direction evaluation in a calculation resulting in Q queries with the allowed ISAT storage A. (For a given ISAT implementation, the allowed ISAT storage A can be specified in terms of either the megabyte allowed or the number of table entries allowed, and these two are equivalent. Here we take A to be the number of table entries allowed.) We have

$$p_R(Q, A) + p_G(Q, A) + p_A(Q, A) + p_D(Q, A) = 1. \tag{9}$$

In a large-scale calculation resulting a large number of queries, the grow and add events are likely only during the table building period, which in general accounts for only a small fraction of the whole simulation; and in the retrieving phase, essentially all queries are resolved either by retrieves or by direct evaluations. Hence for a very long run, we will have $p_R + p_D \approx 1$, and the average CPU time for a query, t_Q, can be well approximated as

$$\begin{aligned} t_Q &= t_R p_R(Q, A) + t_D p_D(Q, A) \\ &= t_R + p_D(Q, A)(t_D - t_R), \end{aligned} \tag{10}$$

where t_R is the average CPU time to perform a retrieve, t_D is the average CPU time to perform a direct function evaluation, and the second step in Eq. 10 follows from the observation $p_R \approx 1 - p_D$. Typically, the retrieve time t_R is several orders of magnitude smaller than the direct evaluation time t_D. According to Eq. 10, the computational performance of ISAT depends highly on the fraction of direct evaluations $p_D(Q, A)$, which is a function of the allowed storage. The ideal computational

performance that ISAT can achieve is $t_Q = t_R$, where essentially all the queries are resolved by retrieves and the contribution of direction evaluation in Eq. 10 is negligible.

Figure 1 shows the average CPU time per query, the fraction of direct evaluations, and the number of tabulated table entries used against the number of table entries allowed from the PaSR calculations with the same specified value of ε_{tol}. The key observation is that the fraction of the computationally-expensive direct evaluations decreases monotonically with the number of allowed table entries A, until it reaches zero for the largest value of A used. Consequently, in the region where A is relatively small, the average query time decreases substantially as A increases. In the region where A is large (here, $A \geq 30,000$), the average query time reaches a plateau and the allowed storage does not have a significant effect on ISAT performance. This is because, in this region, the allowed storage is sufficiently large: almost all of the queries are resolved by retrieves and the contribution of direction evaluations to the query time is negligible.

With a given ISAT implementation, for calculations of a particular reactive flow (with a specified incurred error and a specified number of queries), we can define the critical number of ISAT table entries, A^*, implicitly by

$$p_D(Q, A^*) = \frac{t_R}{t_D - t_R}. \tag{11}$$

Given that $p_D(Q, A)$ 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

$$\frac{t_Q}{t_R} = 1 + \frac{p_D(Q, A)}{p_D(Q, A^*)}. \tag{12}$$

Evidently the *storage ratio* $s \equiv A/A^*$ determines the effectiveness of ISAT. For $s \geq 1$, ISAT is very effective and $t_Q/t_R \leq 2$, i.e., within a factor of 2 of the ideal performance. For $s < 1$, the

time spent on direct evaluations is significant. For the calculation shown in Fig. 1, the value of t_R is about $40\mu s$, the values of t_D is about $5 \times 10^3 \mu s$, and the critical number of table entries,A^*, is about 2.8×10^4. As may be seen from Fig. 1, if the number of allowed table entries is less than the critical value, the calculation incurs a significant fraction of computationally expensive direct evaluations, which causes a dramatic deterioration in the computational performance. Conversely, for $A > A^*$, t_Q depends weakly on A.

It is worth mentioning that for calculations of a particular reactive flow (with a specified incurred error and a specified number of queries), the critical number of ISAT table entries, A^*, depends on the implementation of ISAT. An ISAT implementation with a smaller value of A^*, i.e., reduced storage requirement, is obviously advantageous. It can significantly alleviate the storage constraint for ISAT when applied to challenging reactive flows where large storage for ISAT is desirable. For a given table storage, an implementation with a smaller value of A^* results in a lower fraction of direct evaluations and thus achieves higher computational efficiency.

4 Recent augmentations to *in situ* adaptive tabulation

While the success of the previous ISAT implementation, ISAT4, has been well demonstrated in reactive flow calculations, there are opportunities for further improvement such as improved control of local errors and more efficient and complete searching. The extensive growth of the EOAs in ISAT4 may incur large local errors which are far greater than the user-specified error tolerance ε_{tol}. ISAT4 does not guarantee a complete search in the sense that if a query point is inside the EOA of some tabulated entry, this relevant tabulated entry is not guaranteed to be found. Consequently, unnecessary expensive direct function evaluation may be invoked to evaluate the query point instead of using the cheap linear approximation. Another potential consequence of this incomplete search

is duplicate tabulation. Unnecessary table entries are added to cover the same region of the composition space. Hence incomplete search may significantly affect both the computational efficiency and storage requirement.

To overcome the shortcomings in ISAT4, new augmentations to ISAT are made in the new implementation of ISAT (denoted as ISAT5) by Pope [28], e.g., caches, affine space, ellipsoid of inaccuracy, and error checking and correction. As shown in Section 6, these new augmentations significantly improve the ISAT performance in accuracy, efficiency and storage requirement. A brief overview of these major augmentations is given below. Full details are in [28].

- **Caches**. In ISAT5, two list-based data structures are introduced to store the most frequently used (MFU) tabulated entries, and the most recently used (MRU) tabulated entries. During the retrieve attempt, ISAT scans these two lists to check if the query can be retrieved from a specified number of the initial entries in the lists. If it can, the query is resolved and a linear approximation to $\mathbf{f}$ is returned.

  The idea of the MFU cache is motivated by the observation that a large fraction of queries are resolved by only a small number of table entries (or leaves). Figure 2 shows the fraction of retrieves per leaf (number of retrieves performed on a leaf normalized by the total number of retrieves in the simulation) for the fifty most frequently used leaves in the simulation. As may be seen, many of the most-used leaves are tabulated at the very beginning of the simulation as indicated by the small leaf index (or leaf ID). The leaf index records the tabulation sequence of the table entries. For this particular calculation, the most frequently used table entry resolves more than 10% of the total retrieves in the calculation. The 50 most frequently used leaves resolve about 60% of the total retrieves in the simulation, while the other 1950 leaves together contribute about 40%. Hence high computational efficiency is expected to be achieved by storing these few frequently used table entries in more accessible caches and trying to retrieve from them

first.

The idea of the MFU cache is motivated by the observation that in some applications there are repeated queries with the same or similar values of $\mathbf{x}$ (e.g., a homogeneous region in a flow).

- **Affine space**. In ISAT5, an *affine space* of dimension n_a ($1 \leq n_a \leq n_x$) is introduced and determined based on the tabulation points $\mathbf{x}$. By construction, the tabulation points are close to the affine space. This affine space is used so that searches in the ISAT table can be performed more efficiently in this n_a-dimensional space rather than in the full n_x-dimensional space.
- **Ellipsoid of inaccuracy**. For a given leaf, the ellipsoid of inaccuracy (EOI) is a hyper-ellipsoid centered at the tabulated point $\mathbf{x}$, which is initialized to cover the corresponding ellipsoid of accuracy (EOA). The query compositions outside the EOI are deemed to be inaccurate for linear approximations to the function $\mathbf{f}$. An important role of the EOIs is to identify for which EOAs a grow attempt should be performed. At the same time, the EOIs can prevent the extensive growth of the EOAs of the tabulated points, and thus achieve better control of local errors as shown in Section 6. The EOI implementation can be briefly described as follows. For a given query point $\mathbf{x}^q$, after an unsuccessful retrieve attempt, ISAT tests to see if $\mathbf{x}^q$ falls within the EOI of some tabulated point $\mathbf{x}$. If $\mathbf{x}^q$ is outside the EOI of $\mathbf{x}$, the error ε in the linear approximation is assumed to be larger than the error tolerance ε_{tol}, and ISAT does not retrieve from, nor tries to grow, the EOA of $\mathbf{x}$. On the other hand, if the query point $\mathbf{x}^q$ is inside the EOI but outside the EOA, then the error ε is measured. If the error ε is larger than the error tolerance ε_{tol}, then the existing EOI is shrunk to exclude the query point; otherwise the EOA is grown to include the query point, as illustrated in Fig. 3.
- **Error checking and correction**. The error checking and correction (ECC) is first introduced by Panda [36] particularly to improve the local error control. In the ECC method, on randomly selected queries (resulting in successful retrieves), $\mathbf{x}^q$, the exact value $\mathbf{f}(\mathbf{x}^q)$ is evaluated (in addition to obtaining the approximate value $\mathbf{f}^l$ from ISAT using Eq. 5) so that the error ε is

measured (using Eq. 6). If ε is less than ε_{tol}, the query is deemed to be accurate and no further action is taken. However if ε exceeds ε_{tol}, the EOA used to obtain $\mathbf{f}^l$ is shrunk to exclude $\mathbf{x}^q$.

With the above augmentations, a table entry in ISAT5 contains the information of $\mathbf{x}$, $\mathbf{f}$, the mapping gradient $\mathbf{A}$, the ellipsoid of accuracy (EOA), the ellipsoid of inaccuracy (EOI), the projections of EOAs (denoted as PEOAs) and EOIs (denoted as PEOIs) onto the affine space. In ISAT5, the EOAs and EOIs are stored in binary trees (BTs), and the PEOAs and PEOIs are stored as sets of ellipsoids in ellipsoidal binary trees (EBTs). Each leaf of an EBT stores an individual ellipsoid, and each node of an EBT stores a bounding ellipsoid which covers those of its children. For a given query $\mathbf{x}^q$, the ISAT5 implementation consists of the following five stages to obtain an approximation to $\mathbf{f}(\mathbf{x}^q)$. If $\mathbf{x}^q$ is resolved in any stage, an approximation to $\mathbf{f}(\mathbf{x}^q)$ is returned and the subsequent stages are not performed.

(1) *Primary retrieve*. This is performed through the following three subsequent procedures.
- The binary tree of EOAs is traversed to identify the *primary leaf*. The EOA of the primary leaf is tested to determine whether or not it contains the query $\mathbf{x}^q$.
- The first n_{mru} entries of the most recently used (MRU) cache are tested for an EOA containing $\mathbf{x}^q$, where n_{mru} is a specified integer.
- The first n_{mfu} entries of the most frequently used (MFU) cache is tested for an EOA containing $\mathbf{x}^q$, where n_{mfu} is a specified integer.

If $\mathbf{x}^q$ is in any identified EOA, the query is resolved and a linear approximation to $\mathbf{f}$ is returned.

(2) *Secondary retrieve*. The ellipsoidal binary tree (EBT) of the PEOAs (the projection of EOAs onto the affine space) is traversed in an attempt to find an EOA containing $\mathbf{x}^q$. If $\mathbf{x}^q$ is in any identified EOA, the query is resolved.

(3) *Grow*. A direct function evaluation is performed to obtain $\mathbf{f}(\mathbf{x}^q)$. The EBT of PEOIs (the projection of EOIs onto the affine space) is traversed and if $\mathbf{x}^q$ is in an EOI, then the EOA or

EOI is modified depending on the measured error obtained from Eq. 6. If the measure error is less than ε_{tol}, the corresponding EOA is grown to include the query point $\mathbf{x}^q$. Otherwise, the EOI is shrunk to exclude $\mathbf{x}^q$.

(4) *Add*. If in the preceding stages, $\mathbf{x}^q$ is not in any EOI (i.e., grow fails), and if the table is not full, a new entry associated with $\mathbf{x}^q$ is added.

(5) *Direct evaluation*. Otherwise if the table is full, $\mathbf{f}(\mathbf{x}^q)$ is returned without further action.

After each stage the data structures (caches, BT, EBTs) are updated as necessary.

5 Parametric study of the key parameters in ISAT5

In this section, we perform an investigation of some key parameters and operations in the ISAT5 implementation, particularly, the parameters controlling the affine space, the ellipsoids of inaccuracy, the retrieve and grow attempts, the pair covering, and the frequency of error checking and correction.

To investigate the effect of a single parameter (or operation), calculations are performed by varying the values (or settings) of the single parameter (or operation) considered, while all the other parameters and operations are set to their default settings. The relevant default settings in ISAT5 are given in Table 2.

5.1 Effect of the dimensionality of the affine space

As mentioned, an affine space of dimension n_a ($1 \leq n_a \leq n_x$) is used in ISAT5 so that searching can be performed more efficiently in this n_a-dimensional space rather than in the full n_x-dimensional space. The dimensionality n_a of the affine space is a key parameter, which may affect

Table 2

Default settings of some key parameters and operations in ISAT5

Parameter or operation	default setting
n_a	determined automatically
searching BT of EOA	invoked
searching MRU cache	invoked
searching MFU cache	invoked
searching the EBT of the PEOAs	invoked
ret_{frac}	0.5
$grow_{frac}$	2.0
$pair_{cover}$	2
$degen_r$	10000
$mode_{eoi}$	1
eoi_{lim}	10.0
ad_{theta}	0.9
$mode_{con}$	4
ζ	0.1

the computational performance.

Figure 4 shows the effect of n_a on the ISAT performance for the PaSR calculations with the skeletal and GRI3.0 mechanisms. In these calculations, the amount of searching per query for retrieve attempts is limited by the specified CPU time, which is taken to be the default setting, i.e., half of the average direct evaluation time (as implied by $ret_{frac} = 0.5$); the amount of searching per query for grow attempts is limited by the specified CPU time, which is taken to be the default setting, i.e., twice the average direct evaluation time (as implied by $grow_{frac} = 2.0$). Hence, for calculations with a sufficiently large table, the search in retrieve attempts and grow attempts may not be complete.

As may be seen from the figure, performing searches in a lower dimensional affine space is generally more efficient than in a higher dimensional affine space. With the skeletal mechanism, the following two calculations, the case with $\varepsilon_{tol} = 4 \times 10^{-4}$, $n_a = 2$ and the case with $\varepsilon_{tol} = 5 \times 10^{-4}$, $n_a = 16$ incur similar local error, but a speed-up factor of about 2.5 is observed in going from $n_a = 16$ to $n_a = 2$. Also searching in a lower dimensional affine space saves a significant amount of storage. For the above two cases, the case with $n_a = 2$ uses only half of the number of entries compared to the case with $n_a = 16$. In the current implementation of ISAT5, the dimensionality of the affine space is automatically determined based on the tabulated composition points. The resulting dimensionality of the affine space is generally less than five for all the calculations performed so far. As may be seen from Fig. 4, the automatically determined n_a yields performance not too far from optimal.

To further understand the dependence of adds on the dimensionality of the affine space, in addition to the above results where the amounts of time attempting retrieving and growing are limited, Fig. 5 shows the results where the amount of searching for retrieve attempts and grow attempts per query is unlimited so that in the calculations the search for retrieve and grow attempts is complete.

Similar to the observations on Fig. 4, the results of Fig. 5 show that performing searches in a lower dimensional affine space is generally more efficient than that in a higher dimensional affine space and the automatically determined n_a yields the performance not too far from optimal. Figure 5 also shows that with unlimited grow and retrieve attempts, the number of adds are independent of the dimensionality of the affine space as expected. Hence the increase in the number of adds with the dimensionality of the affine space in Fig. 4 is due to the incomplete retrieve and grow attempts in those calculations.

Another interesting observation is that (with both limited or unlimited grow and retrieve attempts) the incurred error (among the different calculations with the same specified error tolerance) depends on the dimensionality of the affine space. As revealed in Fig. 5d), this is due to the different amounts of ECC performed among the calculations. The ECC used is ECC-Q, which is based on the query time. The calculations with high dimensional affine spaces have large query times, and therefore large amounts of ECC, and consequently they incur smaller local errors. (For calculations without ECC (not shown), we do not observe the dependence of the incurred local errors on the dimensionality of the affine space.)

5.2 Effect of the operations and parameters controlling retrieve attempts

As mentioned above, ISAT achieves computational efficiency by calculating the reaction mapping using cheap retrieves instead of expensive direct evaluations. In ISAT5, retrieve attempts consist of primary retrieves and secondary retrieves. The primary retrieve may invoke the operations of traversing the binary tree (BT) of EOAs, searching the most recently used (MRU) list, and searching the most frequently used (MFU) list. In the secondary retrieve (SR), the EBT of the PEOAs is traversed trying to identify an EOA containing the query. Moreover, 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 ret_{frac} of the average CPU time per direct function evaluation.

In the current ISAT implementation, each of the operations in a retrieve attempt (abbreviated to BT, MRU, MFU, and SR) can be turned off. Figure 6 shows the effect of different retrieve operations in ISAT5. The study is conducted by performing a series of calculations in each of which one of the individual retrieve operations is turned off. Also shown are the results obtained using the default ISAT settings, where all the retrieval operations are performed (with $ret_{frac} = 0.5$ for secondary retrieve). The results clearly illustrate the crucial importance of the secondary retrieve. Without secondary retrieving, a significant computational penalty is incurred because a large fraction of the queries have to be resolved by performing direct function evaluations. For example, with the skeletal mechanism, the case with $\varepsilon_{tol} = 5 \times 10^{-4}$, "no SR" and the case with $\varepsilon_{tol} = 4 \times 10^{-4}$, "def" have similar incurred error, but the latter takes less time by a factor of 10. In contrast, each of the operations in primary retrieve, i.e., traversing the BT of the EOAs, searching the MRU, searching the MFU, does not show decisive importance. The default setting yields close to optimal performance.

To further investigate the effect of secondary retrieving on ISAT performance, Fig. 7 shows the effect of the parameter ret_{frac} (controlling the allowed CPU time in secondary retrieving). For the calculations performed, as shown in Fig. 7d), with the skeletal mechanism, the SR searches are complete except for $ret_{frac} = 0.1$; with the GRI3.0 mechanism, the SR searches are complete except for $ret_{frac} \leq 0.01$. Hence over the range from 0.1 to 0.9, the value of parameter ret_{frac} does not have a significant effect on either the computational efficiency or the storage requirement. (One would not expect to see differences except statistical with $ret_{frac} > 0.1$ due to the complete SR searches.) As illustrated in the calculations with the GRI3.0 mechanism, significant computation penalty is incurred for a small value of ret_{frac} (e.g., 0.001), which results in a significant fraction

of incomplete searches and consequently a large fraction of grow events (not shown). Using the default value of the parameter $ret_{frac} = 0.5$, the performance is not too far from optimal (within 10%).

5.3 Effect of the parameter controlling EOA growth

In the ISAT algorithm, the ellipsoid of accuracy (EOA) associated with each tabulated entry is initialized conservatively, and growing the EOAs is one of the key ingredients to effectively increase the percentage of retrieves and therefore improve the computational efficiency.

As mentioned in Section 4, for a query point, $\mathbf{x}^q$, if the retrieve operation fails, then the EBT of the PEOIs is traversed during which a set of PEOIs and EOIs are tested to check whether or not they contain the query point. If the query point is in an EOI, then the corresponding EOA or EOI is "modified" depending on the measured error. To prevent excessive searches and modification which might incur a performance penalty, the amount of growing is controlled so that the CPU time spent on it (in a given query) is less than a specified fraction $grow_{frac}$ of the average CPU time per direct function evaluation.

Figure 8 illustrates the effect of the parameter $grow_{frac}$ on the performance of ISAT. As may be seen, increasing $grow_{frac}$ from 0.5 to 1.0 results in larger errors, fewer adds, and decreased CPU time per query. The effect of varying $grow_{frac}$ between 1.0 and 4.0 is, in general, smaller, and less regular. For the calculations performed, relatively large values of $grow_{frac}$ are clearly advantageous in terms of both computational efficiency and storage requirement. Using the default value of the parameter $grow_{frac} = 2.0$, the performance is not too far from optimal (within 25%).

5.4 Effect of the parameter controlling pair covering

As mentioned, in ISAT5, the projection of the EOAs and EOIs onto the affine space, i.e., PEOAs and PEOIs, are stored as sets of ellipsoids on which ellipsoidal binary trees (EBTs) are defined. Each leaf of an EBT stores an individual ellipsoid (PEOA or PEOI), and each node of an EBT stores a covering ellipsoid which covers its pair of children. In ISAT5, the EBTs of PEOAs and PEOIs are searched to determine whether a given query is covered by one or more of the leaf ellipsoids. If a PEOA does not cover a query point, then neither does the EOA: but the converse is not necessarily true.

The algorithm used to construct the covering ellipsoid on each node is significant and may affect ISAT performance. Accordingly, we investigate the four different pair covering strategies implemented in ISAT5, namely, Mode 1 (spheroid algorithm with no shrinking); Mode 2 (covariance algorithm); Mode 3 (iterative algorithm); and Mode 4 (spheroid algorithm with shrinking). (See [28] for details.)

Figure 9 illustrates the relative performance of the different pair covering strategies. It is primarily a matter of searching efficiency. A "good" covering ellipsoid excludes more of the space and hence avoids unnecessary negative tests, and so more EOA's and EOI's can be found in a given time. This leads to more growing and hence slightly larger $\varepsilon_{0.9}$, but shorter retrieve and query times. As may be seen from the figure, Mode 2, the default setting in ISAT5, is clearly advantageous over the other three strategies in terms of both the computational efficiency and storage requirement.

5.5 Effect of the parameters controlling ellipsoids of inaccuracy

As mentioned in Section 4, ellipsoids of inaccuracy (EOI) are introduced to identify for which EOAs a grow attempt should be performed. At the same time, the EOIs may prevent the extensive growth of the EOAs of tabulated points, and thus achieve better control of local errors. In the current implementation of ISAT5, there are several important parameters that control the initialization and subsequent shrinking of EOIs. For example the parameter eoi_{lim} controls the initial sizes of the EOIs; the parameter $mode_{eoi}$ determines the method used in shrinking EOIs; ad_{theta} controls how the EOI initialization limits the covering of existing tabulation points; and $degen_r$ controls the degeneration of leaves (i.e., there is no EOI associated with the leaf and the EOA cannot be grown further). A negative value of $degen_r$ indicates that degeneration for the leaf will not be performed; for a positive value of $degen_r$, the leaf degenerates after $degen_r$ retrieves. The settings of these parameters may affect the ISAT performance. The default values for these parameters in the current ISAT5 implementation are $mode_{eoi} = 1$, $eoi_{lim} = 10.0$, $degen_r = -1$ and $ad_{theta} = 0.9$.

Figures 10 and 11 show the effect of different parameters in the EOI algorithm on ISAT performance for the PaSR calculations with the skeletal and GRI3.0 mechanisms, respectively. As may be seen, for the most part, the performance of ISAT is not sensitive to these parameters. Smaller values of eoi_{lim} yield relatively better ISAT performance, especially when small incurred errors are required; larger values of ad_{theta} are slightly advantageous over smaller ones; for the existing methods for shrinking EOIs, $mode_{eoi} = 0, 1$ or 2 is advantageous compared to $mode_{eoi} = 3$, which corresponds to no EOI shrinking; for the calculations performed, the parameter $degen_r$ does not have a significant effect on the ISAT performance. Using the default values of these parameters, the performance of ISAT5 is not too far from optimal.

5.6 Effect of the parameter controlling conflicts between EOAs and EOIs

As mentioned in Section 4, the ellipsoid of inaccuracy (EOI) is a hyper-ellipsoid centered at a tabulated point $\mathbf{x}$, and it is initialized to cover the corresponding ellipsoid of accuracy (EOA). However as the ISAT table develops with the simulation, EOIs may shrink to exclude the inaccurate region and EOAs may grow. Thus there is a possibility that at some later stage, part of some EOAs go outside of the corresponding EOIs. This is called a *conflict* between the EOA and EOI. The strategy used to deal with the situation is significant and may affect ISAT performance. Accordingly we investigate the four different strategies implemented in ISAT, which are controlled by the parameter $mode_{con}$, namely: $mode_{con} = 1$ (restore original EOA, degenerate leaf); $mode_{con} = 2$ (shrink EOA to be covered by EOI, degenerate leaf); $mode_{con} = 3$ (shrink EOA to be covered by EOI, do not degenerate leaf); and $mode_{con} = 4$ (do nothing, i.e., conflict not resolved). As may be seen, the first three strategies are more conservative compared to the last one in terms of controlling the sizes of EOAs.

Figure 12 illustrates the relative performance of different strategies taken to resolve the conflict between EOAs and EOIs. As may be seen from the figure, $mode_{con} = 3$ and $mode_{con} = 4$ are advantageous over $mode_{con} = 1$ and $mode_{con} = 2$, since they require relatively less computational time and storage requirement. The current default setting for $mode_{con}$ is 4, which gives reasonable performance.

5.7 Effect of the frequency of error checking and correction

As mentioned in Section 4, in the error checking and correction method, on randomly selected queries resulting in a successful retrieve, error checking and correction is performed. This process

not only eliminates the error on the query in question, but also prevents (or at least decreases the chances of) inaccurate approximations on subsequent queries with similar values of $\mathbf{x}^q$. This is an expensive process due to the extra direct evaluation required, and in ISAT5 so far we have explored two different methods for controlling the frequency of ECC.

In the first method, denoted as ECC-Q, the frequency of ECC is controlled so that the CPU time consumed in ECC is less than a specified fraction ζ of the total. Figure 13 shows results demonstrating the performance of ISAT with different amounts of allowed time in ECC-Q. The results come from a series of PaSR calculations with the skeletal mechanism, each calculation resulting in 1.0×10^8 queries. There are 18 separate tests corresponding to all combinations of $\varepsilon_{tol} = 1 \times 10^{-4}$, 2×10^{-4}, 3×10^{-4} and $\zeta = 0$, 0.01, 0.03, 0.1, 0.3, 0.5. For $\varepsilon_{tol} = 3 \times 10^{-4}$ and $\zeta = 0$, the incurred error $\varepsilon_{0.9}$ is a little less than 10^{-3}. Increasing ζ from 0 to 0.01 results in more than halving $\varepsilon_{0.9}$ with a negligible CPU penalty. As ζ is increased, $\varepsilon_{0.9}$ decreases, but the CPU time increases. Notice that the optimal performance corresponds to the envelop of the different curves. The optimal value of ζ is around 0.1. With this value, the CPU time required to achieve a given value of $\varepsilon_{0.9}$ is about halved (compared to using $\zeta = 0$). Figure 14 demonstrates the effect of ECC on adds, i.e., the number of table entries required for ISAT. The important observation is that the use of ECC can significantly reduce the number of table entries required for a given incurred error. As may be seen from the figure, about the same error is incurred for the following three cases $\varepsilon_{tol} = 1 \times 10^{-4}$ with $\zeta = 0$, $\varepsilon_{tol} = 2 \times 10^{-4}$ with $\zeta = 0.03$, and $\varepsilon_{tol} = 3 \times 10^{-4}$ with $\zeta = 0.3$; however the storage required is in the ratio 4:2:1.

The same test case is performed for the more complicated mechanism, GRI3.0, which involves 53 species. Figure 15 shows results demonstrating the performance of ISAT with ECC-Q. There are in total 24 separate tests corresponding to all combinations of $\varepsilon_{tol} = 5 \times 10^{-4}$, 6×10^{-4}, 7×10^{-4}, 8×10^{-4} and $\zeta = 0$, 0.01, 0.03, 0.1, 0.3, 0.5. For $\varepsilon = 8 \times 10^{-4}$ and $\zeta = 0$, the incurred error $\varepsilon_{0.9}$ is

a little less than 1.6×10^{-3}. Increasing ζ from 0 to 0.03 to 0.1 results in reducing $\varepsilon_{0.9}$ by 30% and 45%, respectively, with an acceptable CPU penalty. As ζ is increased, $\varepsilon_{0.9}$ decreases, but the CPU time increases, which illustrates the computational penalty while achieving high accuracy. The optimal value of ζ is around 0.1. Figure 16 demonstrates the effect of ECC on adds for GRI3.0. As observed for the skeletal mechanism, the use of ECC can significantly reduce the number of table entries required for a given incurred error.

As shown above, ECC-Q (with the ECC frequency based on the query time) achieves the desirable computational performance for the cases where a sufficiently large storage is assigned for the ISAT table. However with a limited storage assigned for ISAT, some pathological cases are observed where due to the excessive shrinking of EOAs by ECC, an undesirable sharp increase in direct function evaluations is observed after the table is full, which severely deteriorates the computational performance.

To address the above challenge, a more advanced ECC method, denoted as ECC-G, is proposed in which the frequency of ECC is controlled so that the number of ECC events is less than a specified fraction ζ of the number of grow events. One thing worth mentioning is that for a long-run simulation, the fraction of grows (out of total queries) is in general negligible, and so therefore is the fraction of ECC events. Hence the ECC-G incurs a negligible additional computational cost.

Figure 17 shows ECC-G results demonstrating the performance of ISAT with different amounts of ECC allowed. The results come from a series of PaSR calculations with both the skeletal and GRI3.0 mechanisms, each calculation resulting in 1.2×10^8 queries. For the skeletal mechanism, there are 18 separate tests corresponding to all combinations of $\varepsilon_{tol} = 1 \times 10^{-4}$, 2×10^{-4}, 3×10^{-4} and $\zeta = 0$, 0.01, 0.03, 0.1, 0.3, 0.5. For the GRI3.0 mechanism, there are 24 separate tests corresponding to all combinations of $\varepsilon_{tol} = 3 \times 10^{-4}$, 4×10^{-4}, 5×10^{-4}, 6×10^{-4} and $\zeta = 0$, 0.01, 0.03, 0.1, 0.3, 0.5. As observed for the first method, the use of ECC can significantly reduce

both the average query CPU time and the number of table entries required for a given incurred error. For example, for the GRI3.0 mechanism, for $\varepsilon_{tol} = 3 \times 10^{-4}$ and $\zeta = 0$, the incurred error $\varepsilon_{0.9}$ is about 8×10^{-4}. Increasing ζ from 0 to 0.1 results in about halving $\varepsilon_{0.9}$ with a negligible CPU penalty. As ζ is increased, $\varepsilon_{0.9}$ decreases, but the CPU time increases. Notice that the optimal performance corresponds to the envelop of the different curves. The optimal value of ζ is around 0.1 to 0.3. With this value, the CPU time required to achieve a given value of $\varepsilon_{0.9}$ is about halved (compared to using $\zeta = 0$).

In short, the ECC augmentation to the ISAT algorithm produces a significant decrease both in the ratio of $\varepsilon_{0.9}/\varepsilon_{tol}$ and in the number of table entries required for a given incurred error. For the first method ECC-Q, where the frequency of ECC is controlled by the query time, the optimal value of ζ is about 0.1 and it incurs a modest computational cost ($\zeta\%$). For the second method ECC-G, where the frequency of ECC is controlled by the number of grow, the optimal value of ζ is between 0.1 to 0.3 and it incurs a negligible computational cost. With the exception of those reported in the preceding paragraph and shown in Fig. 17, the test results reported in this paper use ECC-Q with the frequency based on CPU time. However, the tests performed reveal that it is preferable to base the ECC frequency on the growing frequency instead, hence ECC-G is recommended for future applications of ISAT.

In summary, in ISAT5, using the default settings in Table 2, the performance is not too far from optimal.

6 Comparison of different implementations of ISAT

In this section, we investigate the relative performance of different implementations of ISAT, namely ISAT4 and ISAT5, in terms of local error control, computational efficiency and storage

requirement. The values of the parameters used in the different implementations are the default values, which, based on experience so far, are close to optimal.

6.1 Incurred local error

We investigate the local error incurred by different implementations of ISAT through an examination of the 90th percentile error, $\varepsilon_{0.9}$, and the cumulative distribution function (CDF) of the incurred error. To obtain the CDF without too large a computational penalty, during the PaSR calculations, we perform the accuracy test every 1000 retrieves. That is, every 1000 retrieves, not only is ISAT used to determine the linear approximation to the reaction mapping, $\mathbf{f}^l(\mathbf{x}^q)$, but also the exact mapping $\mathbf{f}(\mathbf{x}^q)$ is obtained by the computationally expensive direct integration so that the local error ε can be measured directly. Then the CDF of the local error is constructed based on the samples of ε. For each PaSR calculation performed, more than 10^5 samples of the local error ε are obtained.

Figure 18 shows the 90% error $\varepsilon_{0.9}$ against the user-specified error tolerance ε_{tol} for different implementations of ISAT. As may be seen from the figure, for a given implementation of ISAT and for a given test case, $\varepsilon_{0.9}$ varies monotonically with ε_{tol} (almost linearly for small values of ε_{tol}), but the ratio $\varepsilon_{0.9}/\varepsilon_{tol}$ can be affected by the particulars of the implementation. For a given user-specified error tolerance ε_{tol}, the incurred error $\varepsilon_{0.9}$ is significantly less with ISAT5 than with ISAT4. For the two ECC methods considered, they achieve comparable performance. In other words, the new algorithms (ellipsoid of inaccuracy (EOI) and error checking and correction (ECC)) in ISAT5 are very effective in controlling the incurred local error. Hence to achieve the same incurred error, one can specify a much larger error tolerance ε_{tol} in ISAT5 than in ISAT4. For example, for the same incurred error $\varepsilon_{0.9} = 10^{-3}$, ε_{tol} in ISAT5 with ECC can be about five times larger than in ISAT4 for both the skeletal and GRI3.0 mechanisms.

Figure 19 shows the CDF of incurred local error for different implementations of ISAT with three different values of error tolerance ε_{tol}. The quantity $1-F(y) = Prob\{\varepsilon \geq y\}$ is shown to focus on the larger errors. From this figure, we see that long tails exist for the plot $1-F(y)$ associated with all three implementations of ISAT. This clearly shows that none of the implementations exhibits excellent error control associated with a sharp cutoff in the shape of $1-F(y)$ at the specified error tolerance. (Notice that the CDF is plotted against the incurred error normalized by the 90% error $\varepsilon_{0.9}$. The observations on the CDF do not contradict the conclusion that for a given user-specified error tolerance ε_{tol}, the incurred error $\varepsilon_{0.9}$ is significantly less with ISAT5 than with ISAT4 as shown in Fig. 18.) Also Fig. 19 shows that ε_{tol} does not have a significant effect on the behavior in the tail of the curve $1-F(y)$ when plotted against y/ε_{ref} even though ISAT5 with ECC produces less variation in the tail among different error tolerances. The ECC shown here is ECC-Q (i.e., based on the query time), for ECC-G (based on the number of grows, not shown), similar observations are made.

We further examine the tail of the CFD of the incurred error by studying the ratios $\varepsilon_{0.99}/\varepsilon_{0.9}$ and $\varepsilon_{0.999}/\varepsilon_{0.9}$. Figure 20 shows these ratios against the 90% error from different implementations of ISAT for both the skeletal and the GRI3.0 mechanisms. As may be seen from the figure, for ISAT4, there is a large variation in these ratios. The mean over the range of the 90th percentile error considered is about 3.2 and 8 for $\varepsilon_{0.99}/\varepsilon_{0.9}$ and $\varepsilon_{0.999}/\varepsilon_{0.9}$, respectively. The new implementation ISAT5, particularly ISAT5 with ECCs, significantly reduces the fluctuations in these ratios. There is a slight improvement in the means of these ratios over the range of the 90th percentile error considered, which implies that the tail of the error distribution is narrower for ISAT5 than for ISAT4. In other words, ISAT5 (with or without ECC) incurs fewer large errors than ISAT4. (The quantity $\varepsilon_{0.999}/\varepsilon_{0.9}$ may contain some level of statistic error.)

6.2 Efficiency and storage requirement

One thing worth stressing is that in order to study the relative performance of different implementations of ISAT, it is essential to compare the computational efficiency and storage requirements at a fixed incurred error, not at a fixed error tolerance.

For the PaSR test, the average CPU time per query and the storage used by the ISAT table for different implementation of ISAT are shown in Figs. 21 and 22. As may be seen, for both the skeletal and GRI3.0 mechanisms, "error checking and correction" improves ISAT5 performance, substantially so for storage. Moreover ISAT5 with ECC-G (based on the number of of grows) outperforms ISAT 5 with ECC-Q (based on the query time) in terms of computational efficiency and storage.

With ECC-G, ISAT5 is computationally slightly more efficient than ISAT4, especially for more complicated mechanism. For the 16-species skeletal mechanism, at the incurred error $\varepsilon = 1 \times 10^{-3}$, the CPU time per query is about in the ratio 1.3:1 for ISAT4 and ISAT5 with ECC-G. For the 53-species GRI3.0 mechanism, at the incurred error $\varepsilon = 1 \times 10^{-3}$, the CPU time per query is about in the ratio 2:1 for ISAT4 and ISAT5 with ECC-G. (The incurred error $\varepsilon = 1 \times 10^{-3}$ is significant in the sense that this small incurred error is sufficient to guarantee a numerically accurate prediction of all species in computations of turbulent methane/air flames as shown in [27].)

Another important observation is that, the new implementation, ISAT5 with ECC, reaches the specified allowed storage at a much smaller incurred error. For example, for the 53-species GRI3.0 mechanism, ISAT4, ISAT5 without ECC, and ISAT5 with ECC-G reach the allowed storage at about $\varepsilon_{0.9} = 1 \times 10^{-3}$, 8×10^{-4}, and 4×10^{-4}, respectively. For a given incurred error, the new implementation (ISAT5 with ECC) significantly reduces the amount of storage used. For both

the 16-species skeletal mechanism and the 53-species GRI3.0 mechanism, at the incurred error $\varepsilon_{0.9} = 1 \times 10^{-3}$, the amount of storage used is about in the ratio 5:2.4:1 for ISAT4, ISAT5 without ECC, and ISAT5 with ECC. This has significant practical importance because ISAT5 with ECC can significantly alleviate the storage constraint on ISAT when applied to challenging reactive flows where large storage is required.

7 Conclusion

The *in situ* adaptive tabulation (ISAT) algorithm has been widely used to speed up the chemistry calculations in reactive flows. The new implementation of ISAT, denoted as ISAT5 [28], to succeed the previous version of implementation ISAT4, incorporates new algorithms, such as caches, affine space, ellipsoid of inaccuracy, and error checking and correction to improve the accuracy, efficiency, and storage requirement of the algorithm. In this study, we perform a parametric study of the key parameters in ISAT5 and investigate the relative performance of different implementations of ISAT via the PDF calculations of the combustion of a non-premixed methane/air mixture in a partially stirred reactor.

The study shows that the introduction of a low-dimensional affine space substantially improves the ISAT computation performance. Performing searches in a lower dimensional affine space is computationally more efficient than that in the full composition space. Also the study reveals the crucial importance of the secondary retrieving on the ISAT performance. It shows that significant computational penalty is incurred if secondary retrieving is not used.

The study also shows that the *error checking and correction* (ECC) augmentation (especially ECC-G) to the ISAT algorithm produces a significant decrease in the incurred tabulation error. For ISAT5 with ECC-G, for the test cases considered, the optimal number of ECC events is about 10% to

30% of the number of grows. With the optimal amount of ECC, to achieve a given incurred error, both the computational time and the number of table entries required are significantly reduced at a negligible computational cost. The ECC-G method will be employed as the default in future ISAT5 applications.

Moreover, the parametric study shows that using the default settings in ISAT5, specifically the default values of the parameters controlling the affine space, the ellipsoids of inaccuracy, the retrieve and grow attempts, the pair covering, and the frequency of error checking and correction, the performance is not too far from optimal.

The local error incurred by different implementations of ISAT is investigated by studying the 90% error, $\varepsilon_{0.9}$, and the corresponding cumulative distribution function (CDF). Even though they do not exhibit perfect error control associated with a sharp cutoff in the shape of CDF at the user-specified error tolerance, the ellipsoids of inaccuracy (EOI) and error checking and correction (ECC) in ISAT5 are very effective in decreasing the incurred local error. For a given user-specified error tolerance, ε_{tol}, ISAT5 with ECC significantly reduces the incurred error (e.g., by a factor of 2 to 5) compared to ISAT4. In other words, to achieve the same incurred error, one can specify a much larger value of ε_{tol} in ISAT5 than in ISAT4. For example, for the PaSR calculations considered, for the same incurred error $\varepsilon_{0.9} = 10^{-3}$, ε_{tol} in ISAT5 with ECC can be about five times larger than in ISAT4 for both the skeletal and GRI3.0 mechanisms.

This study highlights the fact that comparison of the computational performance of different implementations of ISAT must take account of the level of tabulation error incurred, i.e., performance comparisons must be made at fixed incurred error. To achieve a fixed incurred error, ISAT5 is advantageous over ISAT4 both in the computational efficiency and in storage required. In the PDF calculations of the combustion of non-premixed methane/air in a PaSR, at the incurred error $\varepsilon_{0.9} = 1 \times 10^{-3}$, compared to ISAT4, ISAT5 with ECC-G speeds up the chemistry calculation

by 30% for the 16-species skeletal mechanism and 100% for the 53-species GRI3.0 mechanism. At the same time for both the skeletal and GRI3.0 mechanisms, the storage required (Mbytes) is about in the ratio 5:1 for ISAT4 and ISAT5 with ECC, respectively. This significantly alleviates the storage constraint on ISAT when applied to challenging reactive flows where large storage is required.

Acknowledgments

This work is supported by the National Science Foundation through grant CBET-0426787. 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 Foundation, and other leading public agencies, foundations, and corporations.

References

[1] Chen, J-Y., Kollmann, W. and Dibble, R.W., 1989, PDF modelling of turbulent nonpremixed methane jet flames. *Combustion Science and Technology*, **64**, 315-346.

[2] Turányi, T., 1994, Parameterization of reaction mechanisms using orthonormal polynomials. *Computers & Chemistry*, **18**, 45-54.

[3] Christo, F.C., Masri, A.R. and Nebot, E.M., 1996, Artificial neural network implementation of chemistry with PDF simulation of H_2/CO_2 flames. *Combustion and Flame*, **106**, 406-427.

[4] Blasco, J.A., Fueyo, N., Dopazo, C. and Ballester, J., 1998, Modelling the temporal evolution of a reduced combustion chemical system with an artificial neural network. *Combustion and Flame*, **113**, 38-52.

[5] Pope, S.B., 1997, Computationally efficient implementation of combustion chemistry using *in situ* adaptive tabulation. *Combustion Theory and Modelling*, **1**, 41-63.

[6] Tonse, S.R., Moriary, N.W., Brown, N.J. and Frenklach, M., 1999, PRISM: Piecewise Reusable Implementation of Solution Mapping. An economical strategy for chemical kinetics. *Israel Journal of Chemistry*, **39**, 97-106.

[7] Bell, J.B., Brown, N.J., Day, M.S., Frenklach, M., Grcar, J.F., Propp, R.M. and Tonse, S.R., 2000, Scaling and efficiency of PRISM in adaptive simulations of turbulent premixed flames. *Proceedings of the Combustion Institute*, **28**, 107-113.

[8] Rabitz, H. and Alis, Ö., 1999, General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, **25**,197-233.

[9] Pope, S.B., 1985, PDF methods for turbulent reactive flows. *Progress in Energy and Combustion Science*, **11**, 119-192.

[10] Xu, J. and Pope, S.B., 2000, PDF calculations of turbulent nonpremixed flames with local extinction. *Combustion and Flame*, **123**, 281-307.

[11] Tang, Q., Xu, J. and Pope, S.B., 2000, PDF calculations of local extinction and NO production in piloted-jet turbulent methane/air flames. *Proceedings of the Combustion Institute*, **28**, 133-139.

[12] Masri, A.R., Cao, R., Pope, S.B. and Goldin, G.M., 2004, PDF calculations of turbulent lifted flames of H_2/N_2 issuing into a vitiated co-flow. *Combustion Theory and Modelling*, **8**, 1-22.

[13] Liu, K., Pope, S.B. and Caughey, D.A., 2005, Calculations of bluff-body stabilized flames using a joint probability density function model with detailed chemistry. *Combustion and Flame*, **141**, 89-117.

[14] Cao, R. and Pope, S.B., 2005, The influence of chemical mechanisms on PDF calculations of nonpremixed piloted jet flames. *Combustion and Flame*, **143**, 450-470.

[15] Lu, L., Ren, Z., Raman, V., Pope, S.B. and Pitsch, H., 2004, LES/FDF/ISAT computations of turbulent flames. *Proceedings of CTR Summer Program 2004*, Center For Turbulence Research, 283-294.

[16] Lu, L., Ren, Z., Lantz, S.R., Raman, V., Pope, S.B. and Pitsch, H., 2005, Investigation of strategies for the parallel implementation of ISAT in LES/FDF/ISAT computations. 4th Joint Meeting of the U. S. Sections of the Combustion Institute, Drexel University, Philadelphia, PA, March 20-23.

[17] Singer, M.A. and Pope, S.B., 2004, Exploiting ISAT to solve the reaction-diffusion equation. *Combustion Theory and Modelling*, **8**, 361-383.

[18] Singer, M.A., Pope, S.B. and Najm, H.N., 2006, Modeling unsteady reacting flow with operator-splitting and ISAT. *Combustion and Flame*, **147**, 150-162.

[19] Gordon, R.L., Masri, A.R., Pope S.B. and Goldin, G.M., 2007, A numerical study of auto-ignition in turbulent lifted flames issuing into a vitiated co-flow. *Combustion Theory and Modelling*, **11**, 351-376.

[20] Wang, L.G. and Fox, R.O., 2003, Application of in situ adaptive tabulation to CFD simulation of nano-particle formation by reactive precipitation. *Chemical Engineering Science*, **58**, 4387-4401.

[21] Hedengren, J.D. and Edgar, T.F., 2005, In situ adaptive tabulation for real-time control. *Industrial and Engineering Chemistry Research*, **44**, 2716-2724.

[22] Varshney, A. and Armaou, A., 2005, Multiscale optimization using hybrid PDE/kMC process systems with application to thin film growth. *Chemical Engineering Science*, **60**, 6780-6794.

[23] Veljkovic, I., Plassmann, P.E. and Haworth, D.C., 2003, A scientific on-line database for efficient function approximation. Computational Science and its Applications - ICCSA 2003, *Lecture Notes in Computer Science*, **2667**, 643-653.

[24] Chen, J.-Y., 2004, Analysis of in situ adaptive tabulation performance for combustion chemistry and improvement with a modified search algorithm. *Combustion Science and Technology*, **176**, 1153-1169.

[25] Panda, B., Riedewald, M., Pope, S.B., Gehrke, J. and Chew, L.P., 2006, Indexing for function approximation. *Very Large Data Bases*, ACM 1-59593-385-9/06/09.

[26] Lu, L., Lantz, S.R., Ren, Z. and Pope, S.B., 2007, Computationally efficient implementation of combustion chemistry in parallel PDF calculations. *Combustion Theory and Modelling*, in preparation.

[27] Liu, B.J.D. and Pope, S.B., 2005, The performance of in situ adaptive tabulation in computations of turbulent flames. *Combustion, Theory Modelling*, **9** (4), 549-568.

[28] Pope, S.B., 2007, An improved algorithm for *in situ* adaptive tabulation. Cornell University, in preparation.

[29] Correa, S.M., 1993, Turbulence-chemistry interactions in the intermediate regime of premixed combustion. *Combustion and Flame*, **93**, 41-60.

[30] Correa, S.M. and Braaten, M.E., 1993, Parallel simulations of partially stirred methane combustion. *Combustion and Flame*, **94**, 469-486.

[31] Chen, J-Y., 1997, Stochastic modeling of partially stirred reactors. *Combustion Science and Technology*, **122**, 63-94.

[32] Yang, B. and Pope, S.B., 1998, An investigation of the accuracy of manifold methods and splitting schemes in the computational implementation of combustion chemistry. *Combustion and Flame*, **112**, 16-32.

[33] Ren, Z. and Pope, S.B., 2004, An investigation of the performance of turbulent mixing models. *Combustion and Flame*, **136**, 208-216.

[34] Saxena, V. and Pope, S.B., 1998, PDF calculations of major and minor species in a turbulent piloted jet flame. *Proceedings of the Combustion Institute*, **27**, 1081-1086.

[35] Smith, G.P., Golden, D.M., Frenklach, M., Moriarty, N.W., Eiteneer, B., Goldenberg, M., Bowman, C.T., Hanson, R.K., Song, S., Gardiner, Jr., W.C., Lissianski, V. and Qin, Z., http://www.me.berkeley.edu/gri_mech/.

[36] Panda, B., 2007, private communication.

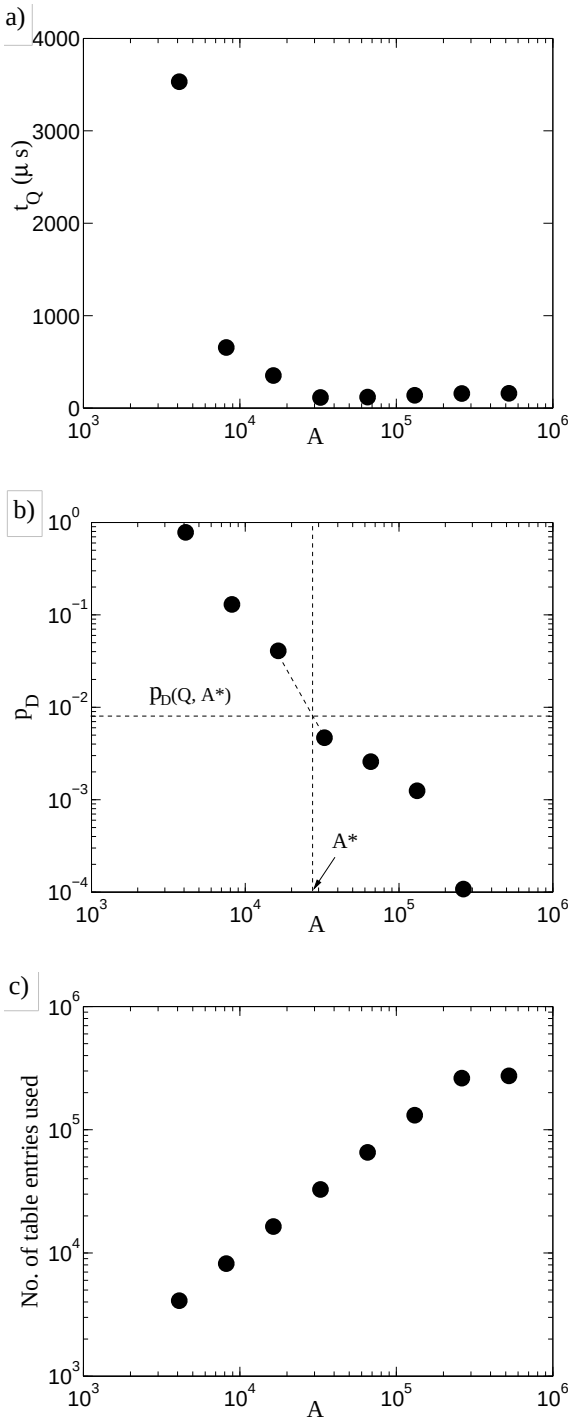


Fig. 1. For the PaSR test with the 16-species skeletal mechanism, figure showing $^{a)}$ the CPU time (μs) per query; $^{b)}$ fraction of direct evaluations; $^{c)}$ number of table entries used against the number of table entries allowed in ISAT5. The error tolerance is $\varepsilon_{tol} = 1 \times 10^{-4}$ and the calculation results in 1.5×10^8 queries.

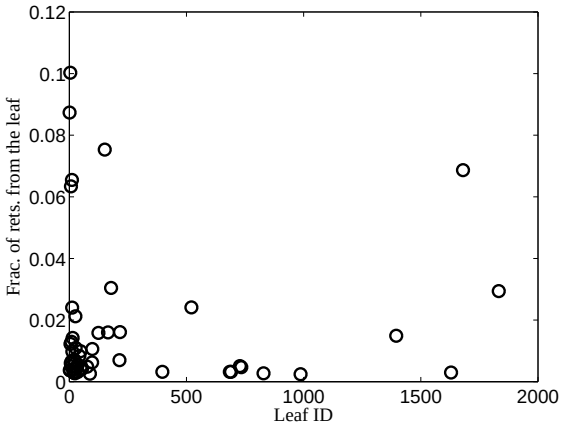


Fig. 2. For the PaSR test using the 16-species skeletal mechanism with error tolerance $\varepsilon_{tol} = 1 \times 10^{-3}$ and number of allowed table entries $A = 2 \times 10^3$, figure showing the fraction of retrieve from the fifty most frequently used leaves. The leaf ID (or leaf index) records the tabulation sequence of the table entries. The calculation results in 1.0×10^8 queries and the ISAT table reaches the storage limit.

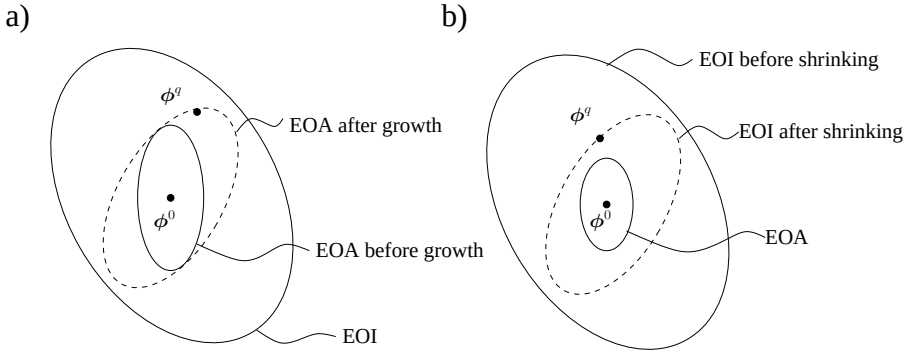


Fig. 3. $^{a)}$ Sketch showing the growth of an ellipsoid of accuracy to include the query point $\mathbf{x}^q$ at which the error ε is less than the error tolerance ε_{tol}; $^{b)}$ Sketch showing the shrinking of an ellipsoid of inaccuracy to exclude the query point $\mathbf{x}^q$ at which the error ε is greater than the error tolerance ε_{tol}.

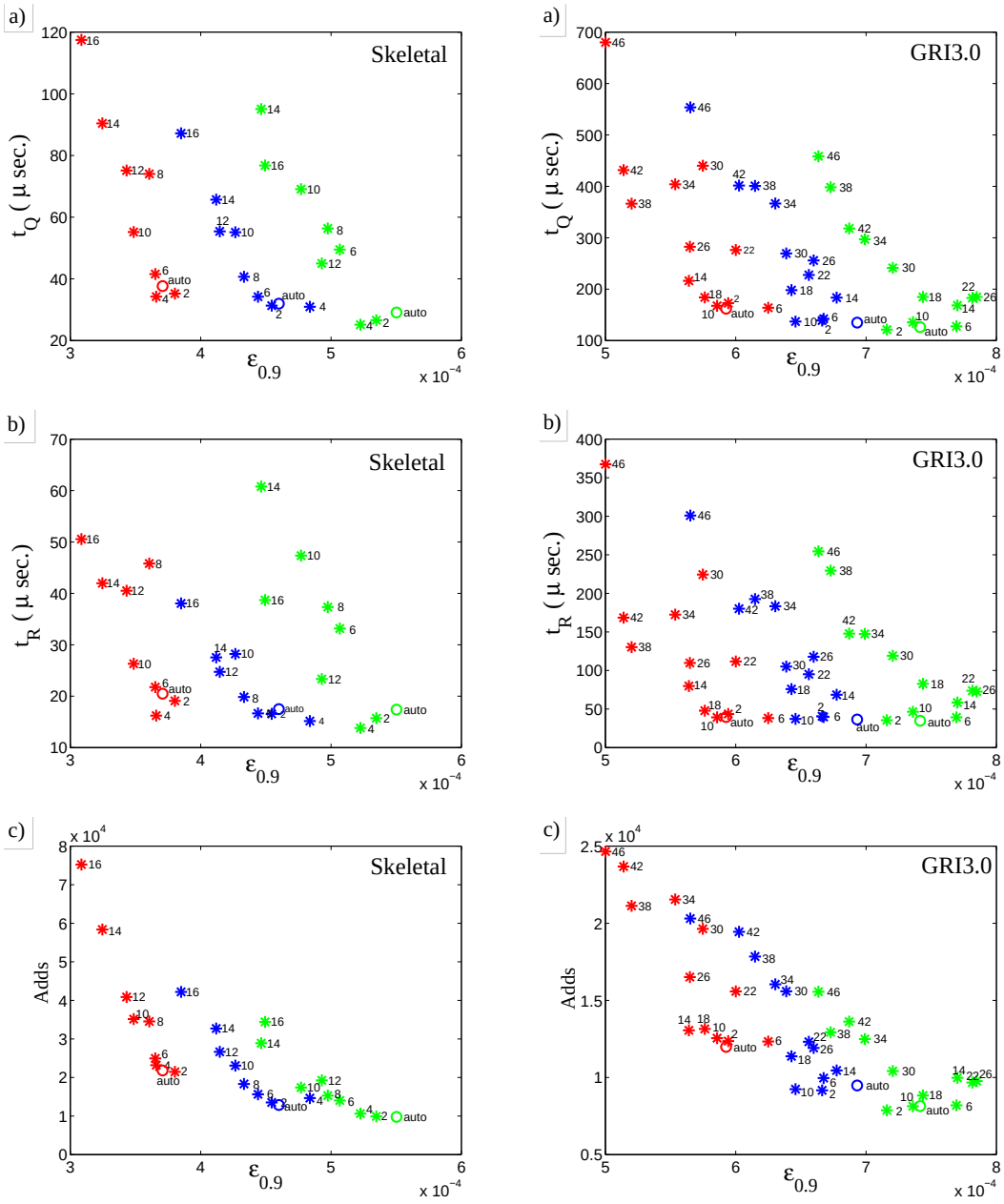


Fig. 4. For the PaSR test, figure showing the effect of the dimensionality of the affine space n_a on ISAT performance. Figures show $^{a)}$ CPU time (μs) per query; $^{b)}$ CPU time (μs) per retrieve, and $^{c)}$ number of adds (or table entries) against 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 4 \times 10^{-4}$ (red), $\varepsilon_{tol} = 5 \times 10^{-4}$ (blue) and $\varepsilon_{tol} = 6 \times 10^{-4}$ (green); right column: GRI3.0 mechanism with $\varepsilon_{tol} = 5 \times 10^{-4}$ (red), $\varepsilon_{tol} = 6 \times 10^{-4}$ (blue) and $\varepsilon_{tol} = 7 \times 10^{-4}$ (green). Symbol $*$: implementation with fixed values of n_a (indicated by the numbers on the plot); $\circ$: implementation with n_a automatically determined. In the calculations, the amount of attempted retrieving and growing is limited by the specified CPU time. The table is not full for all the calculations and each calculation results in 1.2×10^8 queries.

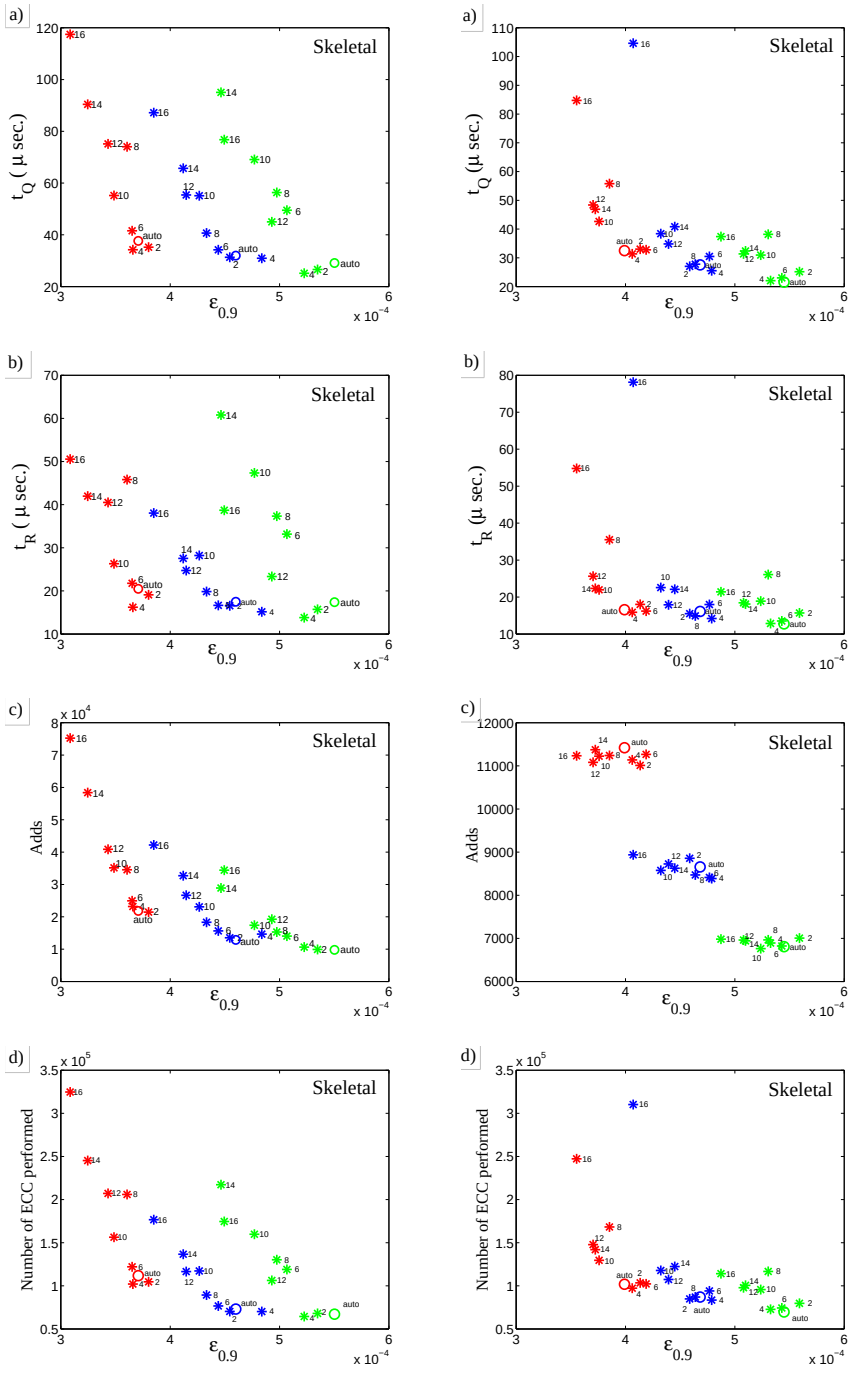


Fig. 5. For the PaSR test, figure showing the effect of the dimensionality of the affine space n_a on ISAT performance with the skeletal mechanism. Left column: retrieve and grow attempts limited by the specified CPU time; right column: unlimited retrieve and grow attempts, i.e., complete searches for retrieving and growing. Figures shows $^{a)}$ CPU time (μs) per query, $^{b)}$ CPU time (μs) per retrieve, $^{c)}$ number of adds (or tabulated entries), and $^{d)}$ number of ECC performed against 90% error. Red: $\varepsilon_{tol} = 4 \times 10^{-4}$; blue: $\varepsilon_{tol} = 5 \times 10^{-4}$; green: $\varepsilon_{tol} = 6 \times 10^{-4}$. Symbol $*$: implementation with fixed values of n_a; $\circ$: implementation with n_a automatically determined. The table is not full for all the calculations and each calculation results in 1.2×10^8 queries.

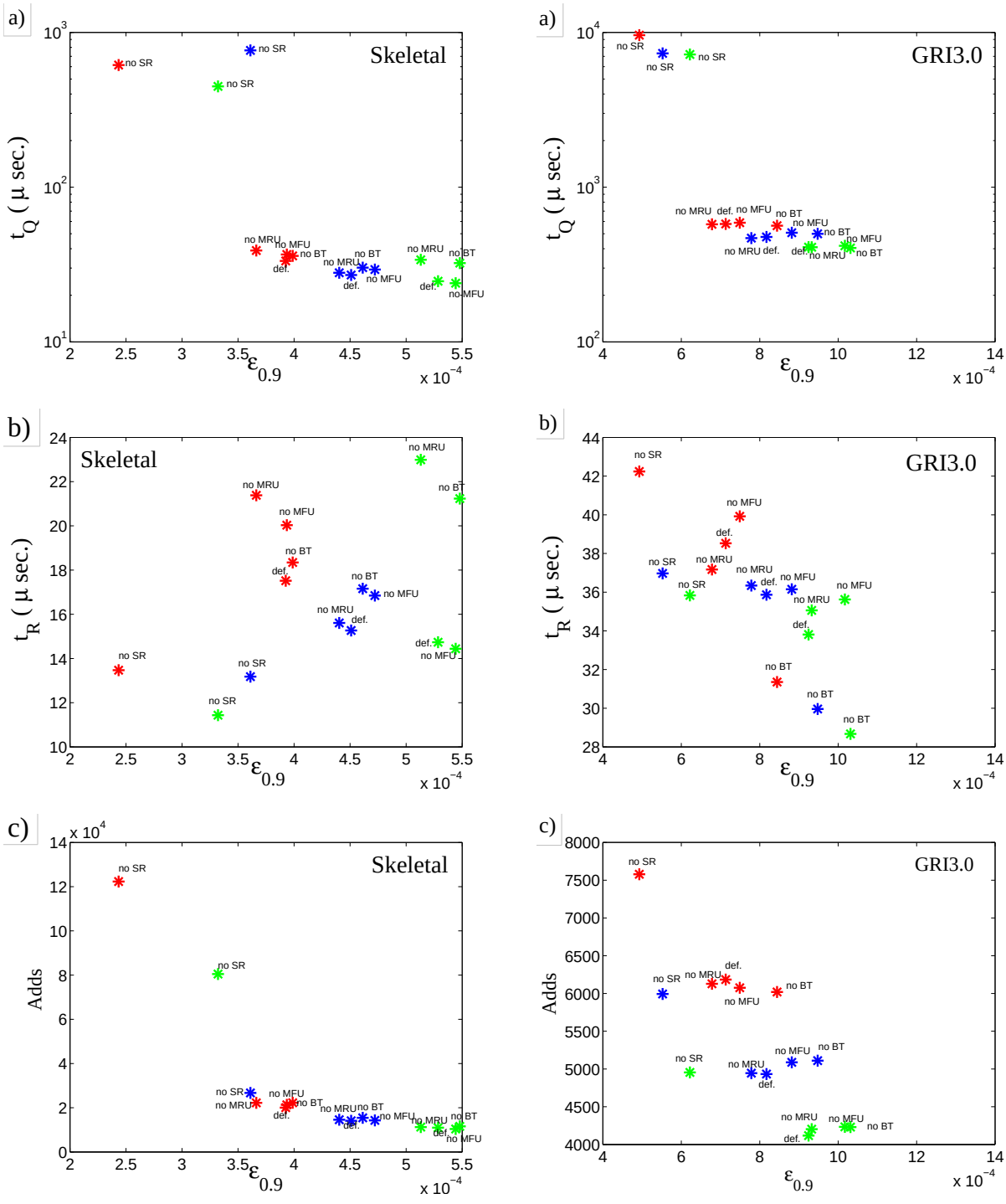


Fig. 6. For the PaSR test, figure showing the effect of different retrieve operations on ISAT performance. Figure shows $^{a)}$ CPU time (μs) per query, $^{b)}$ CPU time (μs) per retrieve, and $^{c)}$ number of adds against 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 4 \times 10^{-4}$ (red), $\varepsilon_{tol} = 5 \times 10^{-4}$ (blue), $\varepsilon_{tol} = 6 \times 10^{-4}$ (green), and each calculation results in 1.2×10^8 queries; right column: GRI3.0 mechanism with $\varepsilon_{tol} = 5 \times 10^{-4}$ (red), $\varepsilon_{tol} = 6 \times 10^{-4}$ (blue), $\varepsilon_{tol} = 7 \times 10^{-4}$ (green), and each calculation results in 1.2×10^7 queries. The number of table entries allowed is sufficiently large and the table is not full for all the calculations.

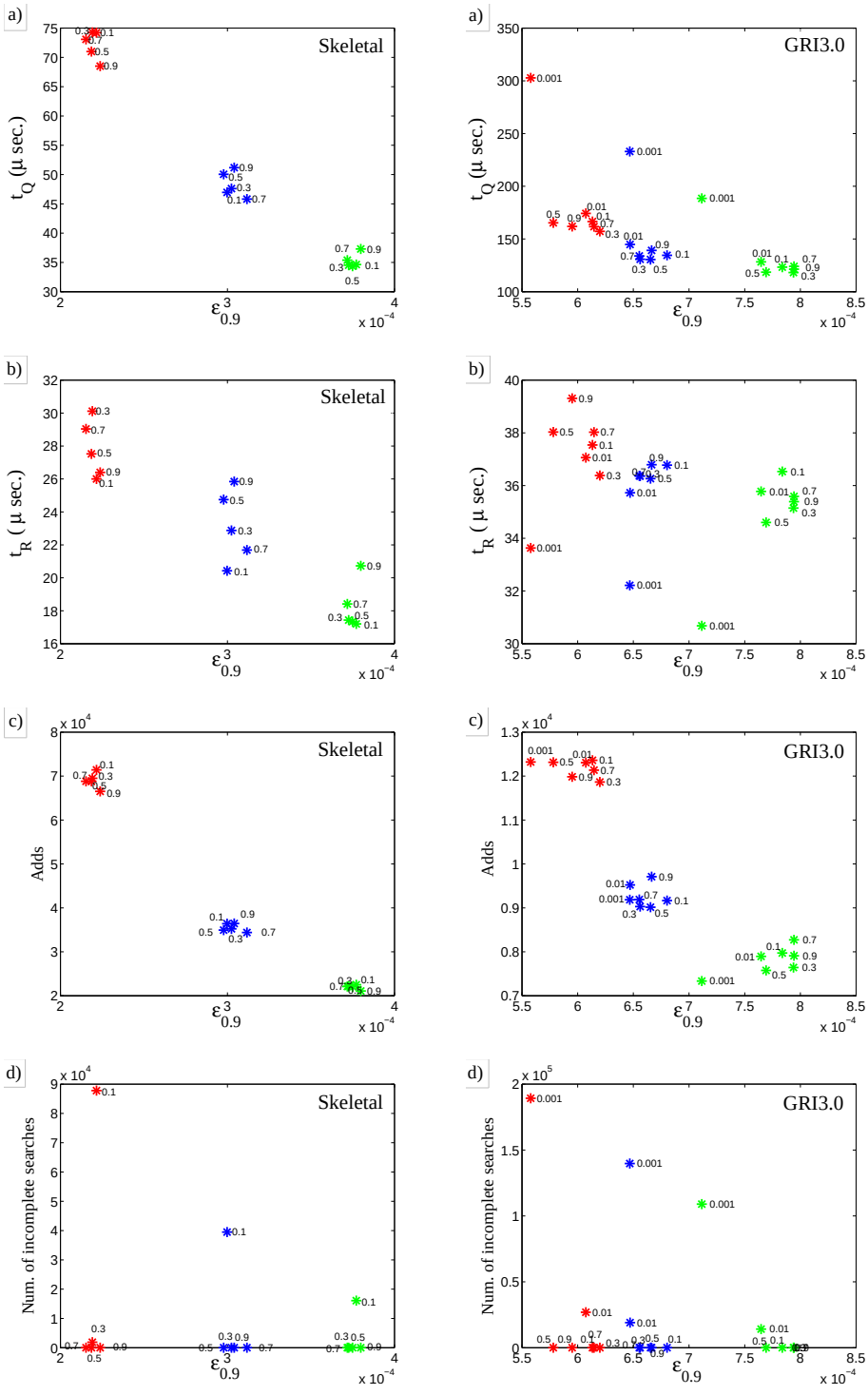


Fig. 7. For the PaSR test, figure showing the effect of parameter ret_{frac} on ISAT performance. Figure shows $^{a)}$ CPU time (μs) per query, $^{b)}$ CPU time (μs) per retrieve, and $^{c)}$ number of adds against 90% error. $^{d)}$ number of incomplete searches against 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 2 \times 10^{-4}$ (red), $\varepsilon_{tol} = 3 \times 10^{-4}$ (blue), $\varepsilon_{tol} = 4 \times 10^{-4}$ (green); right column: GRI3.0 mechanism with $\varepsilon_{tol} = 5 \times 10^{-4}$ (red), $\varepsilon_{tol} = 6 \times 10^{-4}$ (blue), $\varepsilon_{tol} = 7 \times 10^{-4}$ (green). The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each calculation results in 1.2×10^8 queries.

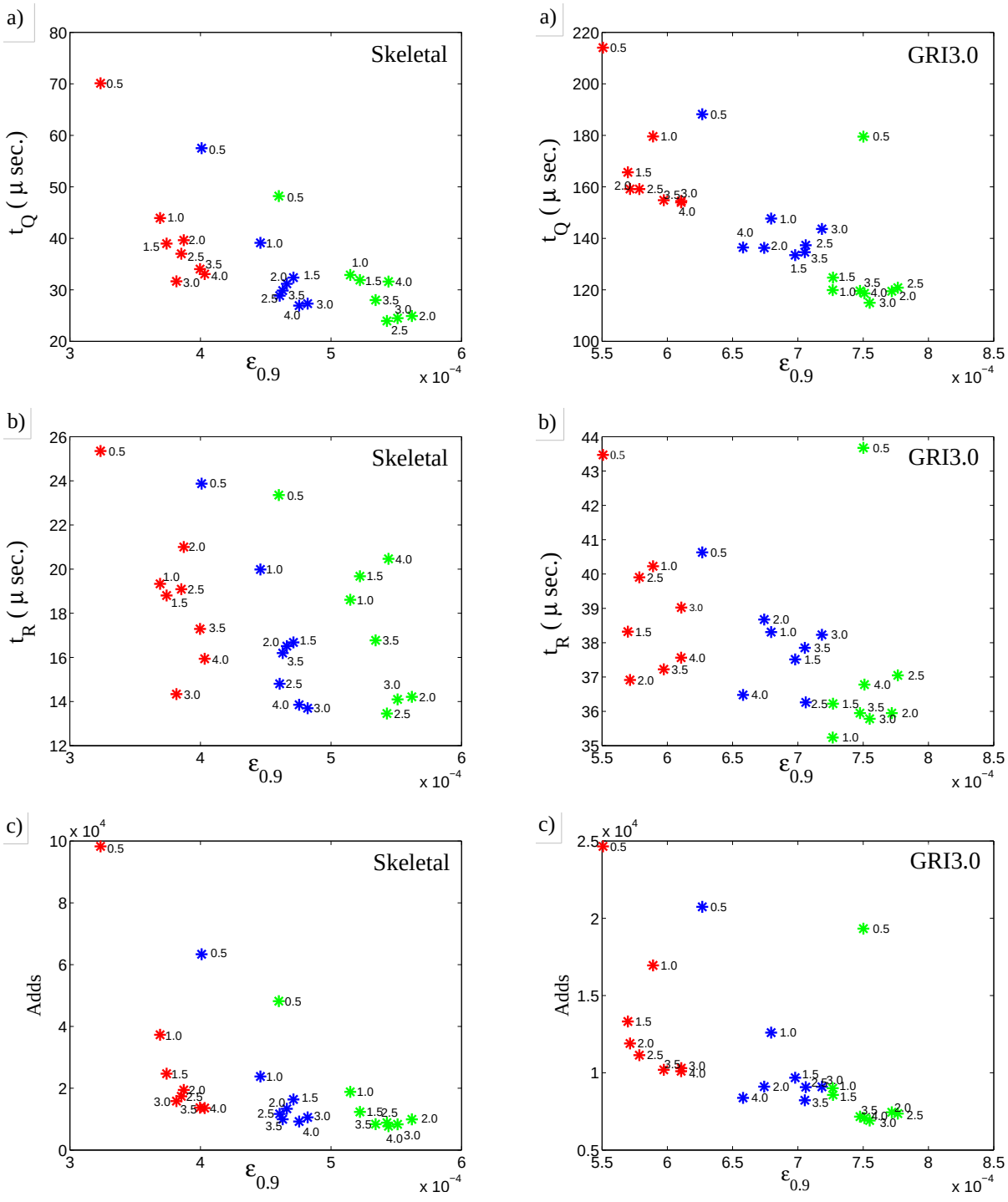


Fig. 8. For the PaSR test, figure showing the effect of the parameter $grow_{frac}$ on ISAT performance for different values of the ISAT error tolerance ε_{tol}. Figure shows $^{a)}$ CPU time (μs) per query $^{b)}$ CPU time (μs) per retrieve and $^{c)}$ number of adds against 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 4 \times 10^{-4}$ (red $*$), $\varepsilon_{tol} = 5 \times 10^{-4}$ (blue $*$), and $\varepsilon_{tol} = 6 \times 10^{-4}$ (green $*$); right column: GRI3.0 mechanism with $\varepsilon_{tol} = 5 \times 10^{-4}$ (red $*$), $\varepsilon_{tol} = 6 \times 10^{-4}$ (blue $*$), and $\varepsilon_{tol} = 7 \times 10^{-4}$ (green $*$). The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each calculation results in 1.2×10^8 queries.

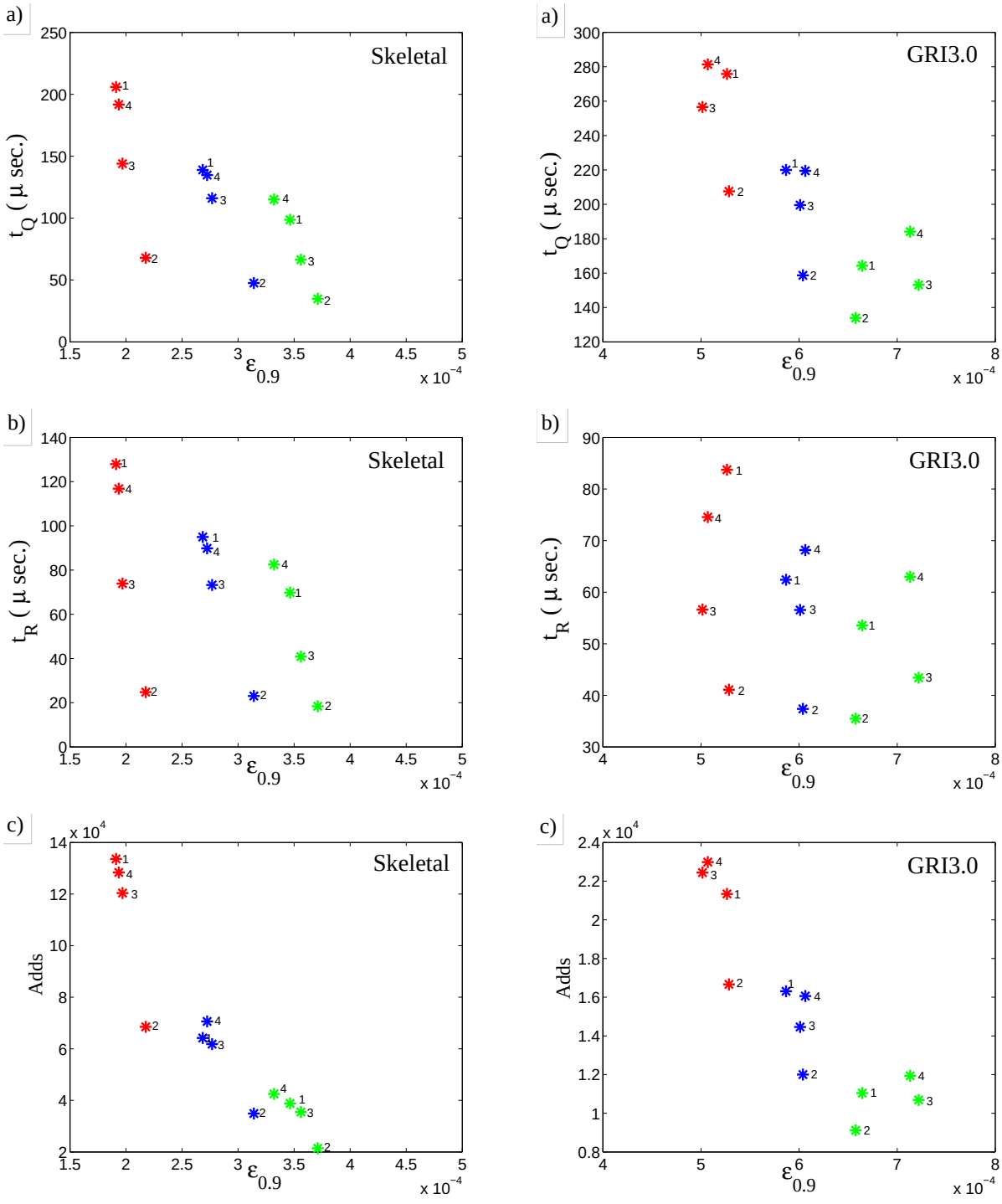


Fig. 9. For the PaSR test, figure showing the effect of different pair covering algorithms on ISAT performance. For different algorithms, figures show $^{a)}$ CPU time (μs) per query, $^{b)}$ CPU time (μs) per retrieve, and $^{c)}$ number of adds against 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 2 \times 10^{-4}$ (red $*$), $\varepsilon_{tol} = 3 \times 10^{-4}$ (blue $*$) and $\varepsilon_{tol} = 4 \times 10^{-4}$ (green $*$); right column: GRI3.0 mechanism with $\varepsilon_{tol} = 4 \times 10^{-4}$ (red $*$), $\varepsilon_{tol} = 5 \times 10^{-4}$ (blue $*$) and $\varepsilon_{tol} = 6 \times 10^{-4}$ (green $*$). The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each calculation results in 1.2×10^8 queries.

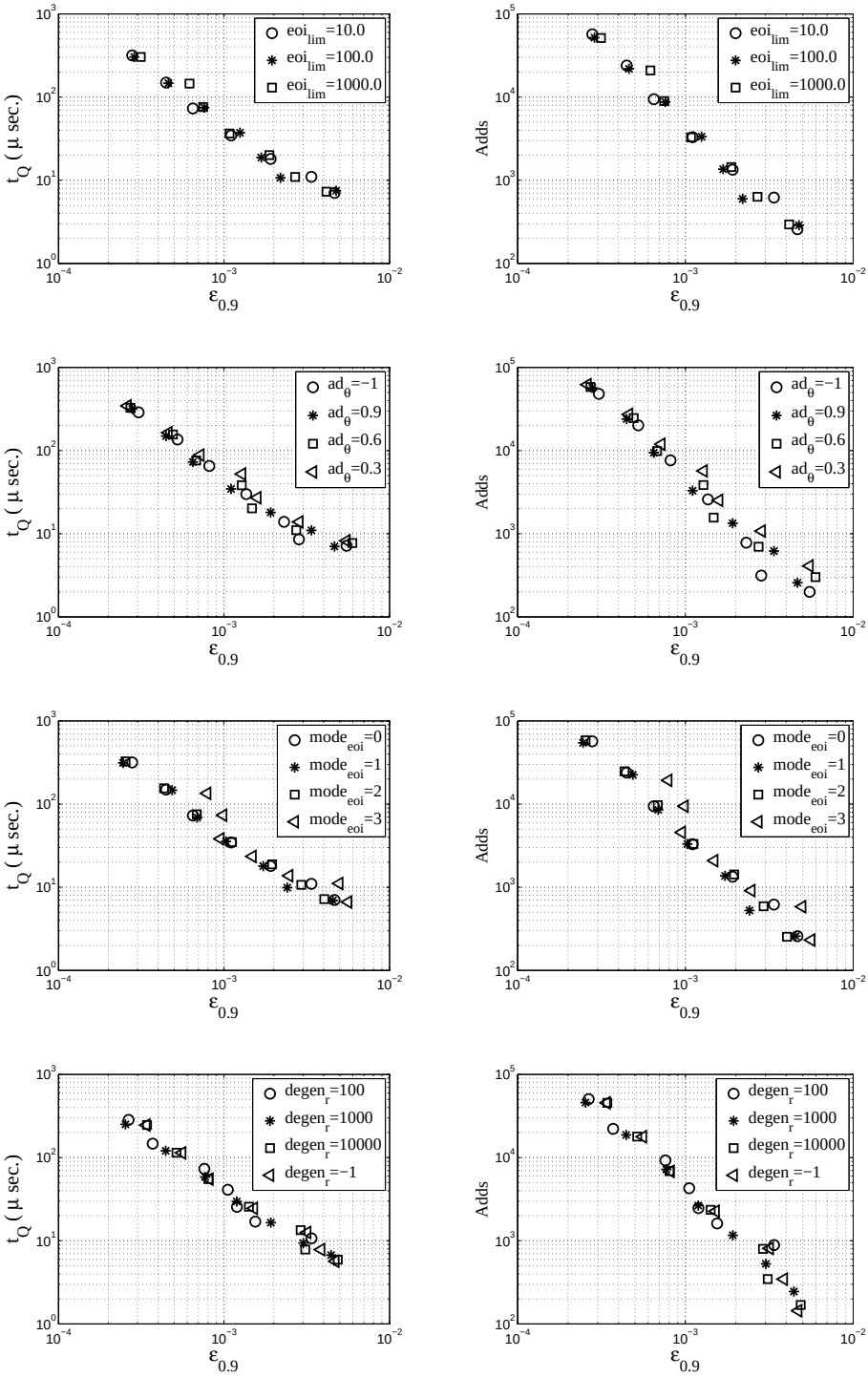


Fig. 10. For the PaSR test with the skeletal mechanism, figure showing the effect of different parameters in the EOI algorithm on ISAT performance. Left column, CPU time (μs) per query against 90% error; right column, number of adds against 90% error. The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each calculation results in 1.2×10^7 queries.

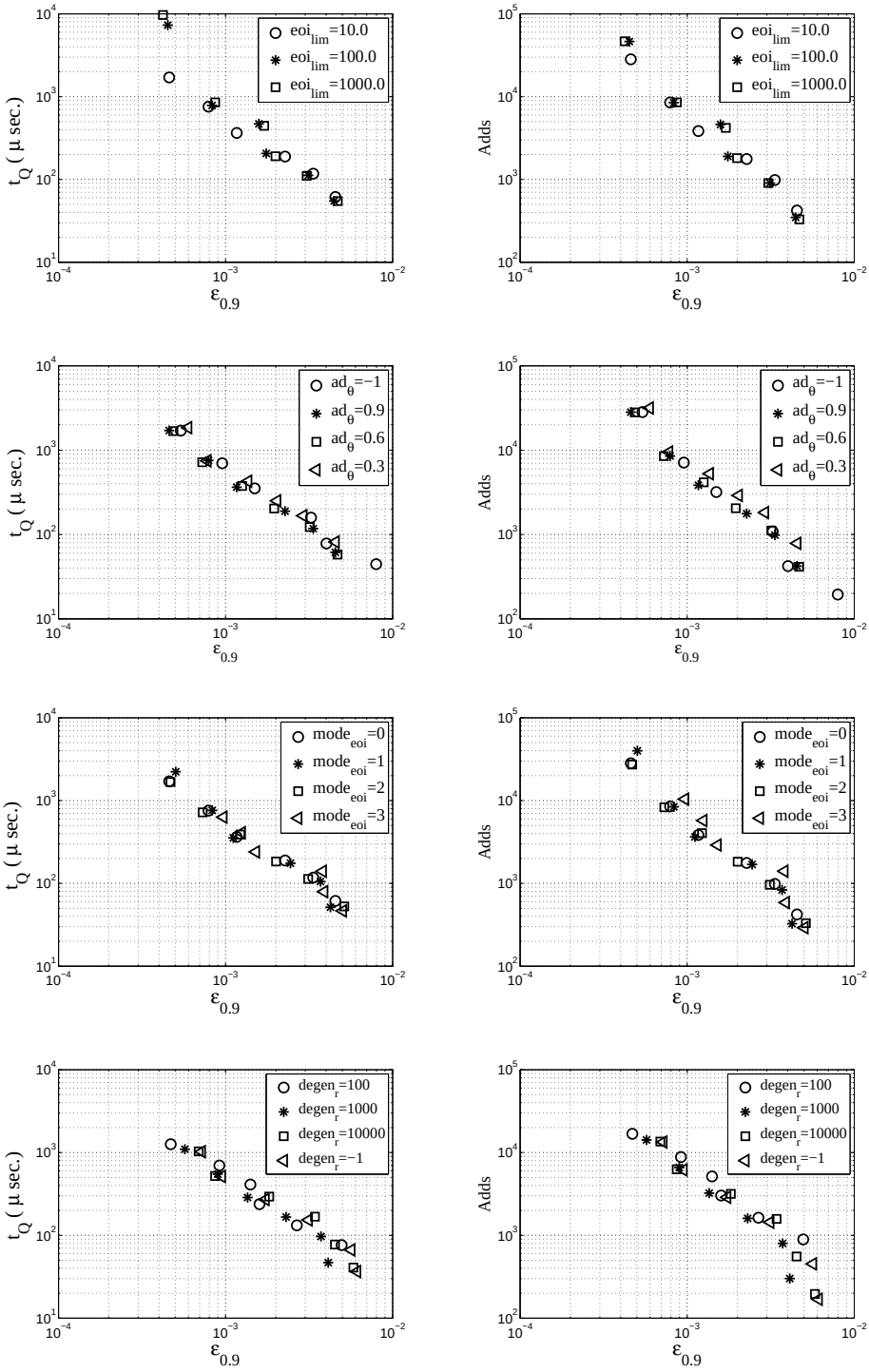


Fig. 11. For the PaSR test with the GRI3.0 mechanism, figure showing the effect of different parameters in the EOI algorithm on ISAT performance. Left column, CPU time (μs) per query against 90% error; right column, number of adds against 90% error. The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each calculation results in 1.2×10^7 queries.

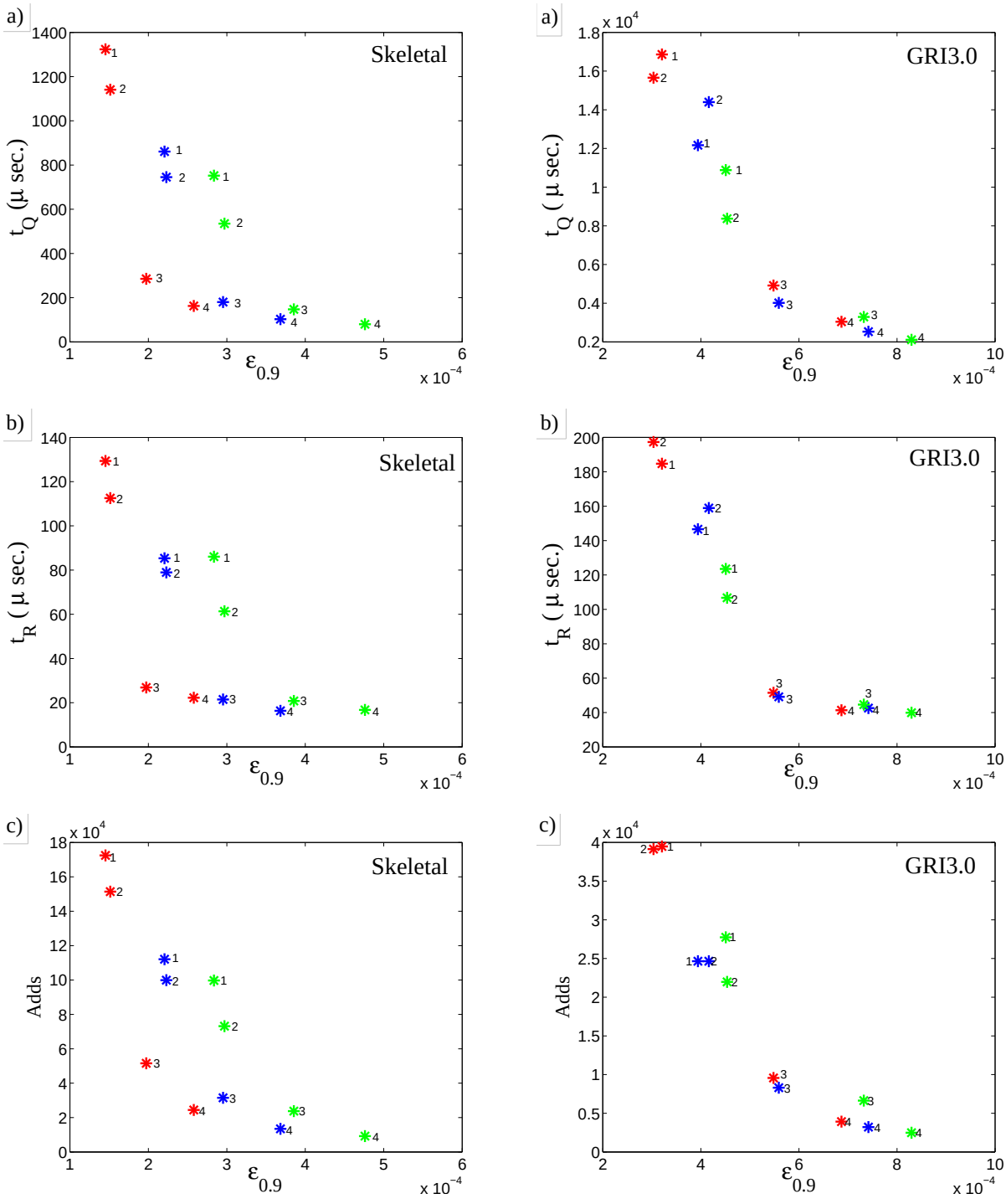


Fig. 12. For the PaSR test, figure showing the effect of different strategies ($mode_{con}$ = 1, 2, 3, or 4) taken to resolve the conflict between EOAs and EOIs. For different strategies, figure shows $^{a)}$ CPU time (μs) per query, $^{b)}$ CPU time (μs) per retrieve, and $^{c)}$ number of adds against 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 2 \times 10^{-4}$ (red $*$), $\varepsilon_{tol} = 3 \times 10^{-4}$ (blue $*$) and $\varepsilon_{tol} = 4 \times 10^{-4}$ (green $*$), and each calculation results in 3.0×10^7 queries; right column: GRI3.0 mechanism with $\varepsilon_{tol} = 4 \times 10^{-4}$ (red $*$), $\varepsilon_{tol} = 5 \times 10^{-4}$ (blue $*$) and $\varepsilon_{tol} = 6 \times 10^{-4}$ (green $*$). The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each calculation results in 1.2×10^6 queries.

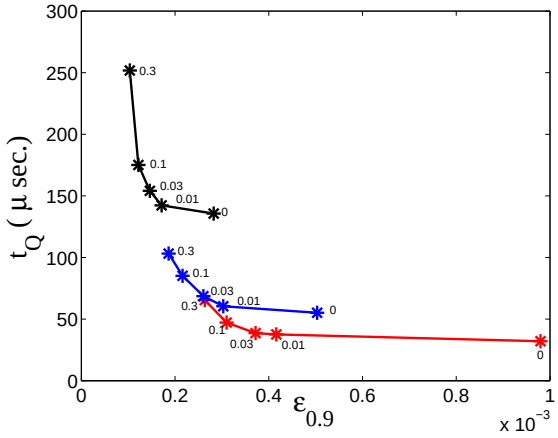


Fig. 13. For the PaSR test with the 16-species skeletal mechanism, figure showing the CPU time (μs) per query against 90% error of ISAT5 with ECC-Q. Error tolerance $\varepsilon_{tol} = 1 \times 10^{-4}$ (black $*$), 2×10^{-4} (blue $*$), 3×10^{-4} (red $*$); the numbers 0, 0.01, 0.03, 0.1, 0.3 indicate the value of ζ controlling the amount of ECC performed. The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each results in 1.0×10^8 queries.

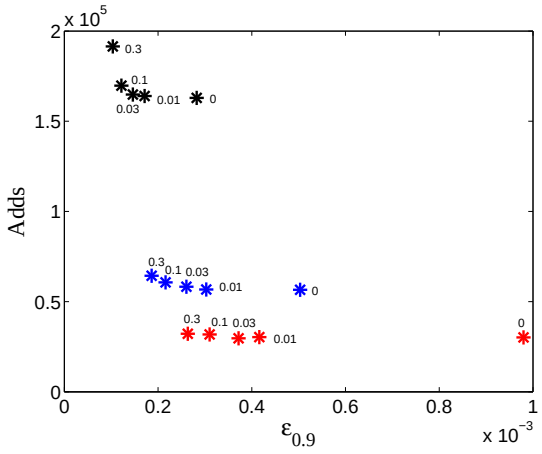


Fig. 14. For the PaSR test with the 16-species skeletal mechanism, figure showing number of adds against 90% error of ISAT with ECC-Q. Error tolerance $\varepsilon_{tol} = 1 \times 10^{-4}$ (black $*$), 2×10^{-4} (blue $*$), 3×10^{-4} (red $*$); the numbers 0, 0.01, 0.03, 0.1, 0.3 indicate the value of ζ controlling the amount of ECC performed. The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each calculation results in 1.0×10^8 queries.

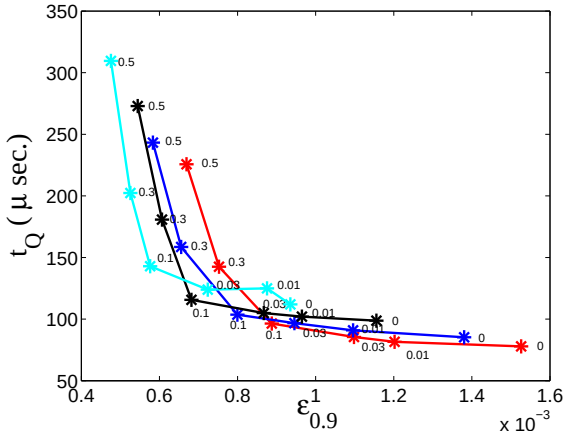


Fig. 15. For the PaSR test with the GRI3.0 mechanism, figure showing CPU time (μs) per query against 90% error of ISAT5 with ECC-Q. Error tolerance $\varepsilon_{tol} = 5 \times 10^{-4}$ (cyan $*$), 6×10^{-4} (black $*$), 7×10^{-4} (blue $*$), 8×10^{-4} (red $*$) ; the numbers 0, 0.01, 0.03, 0.1, 0.3, 0.5 indicate the value of ζ controlling the amount of ECC performed. The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each calculation results in 1.5×10^8 queries.

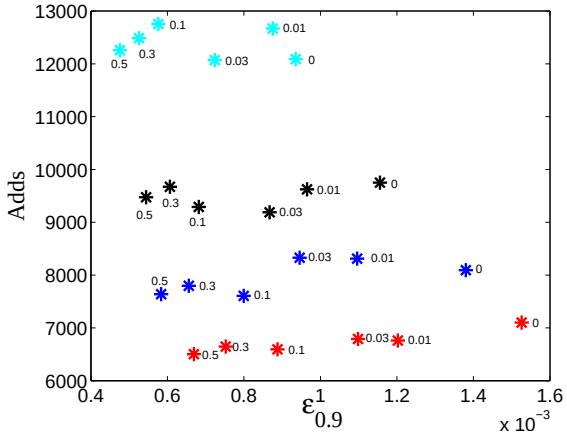
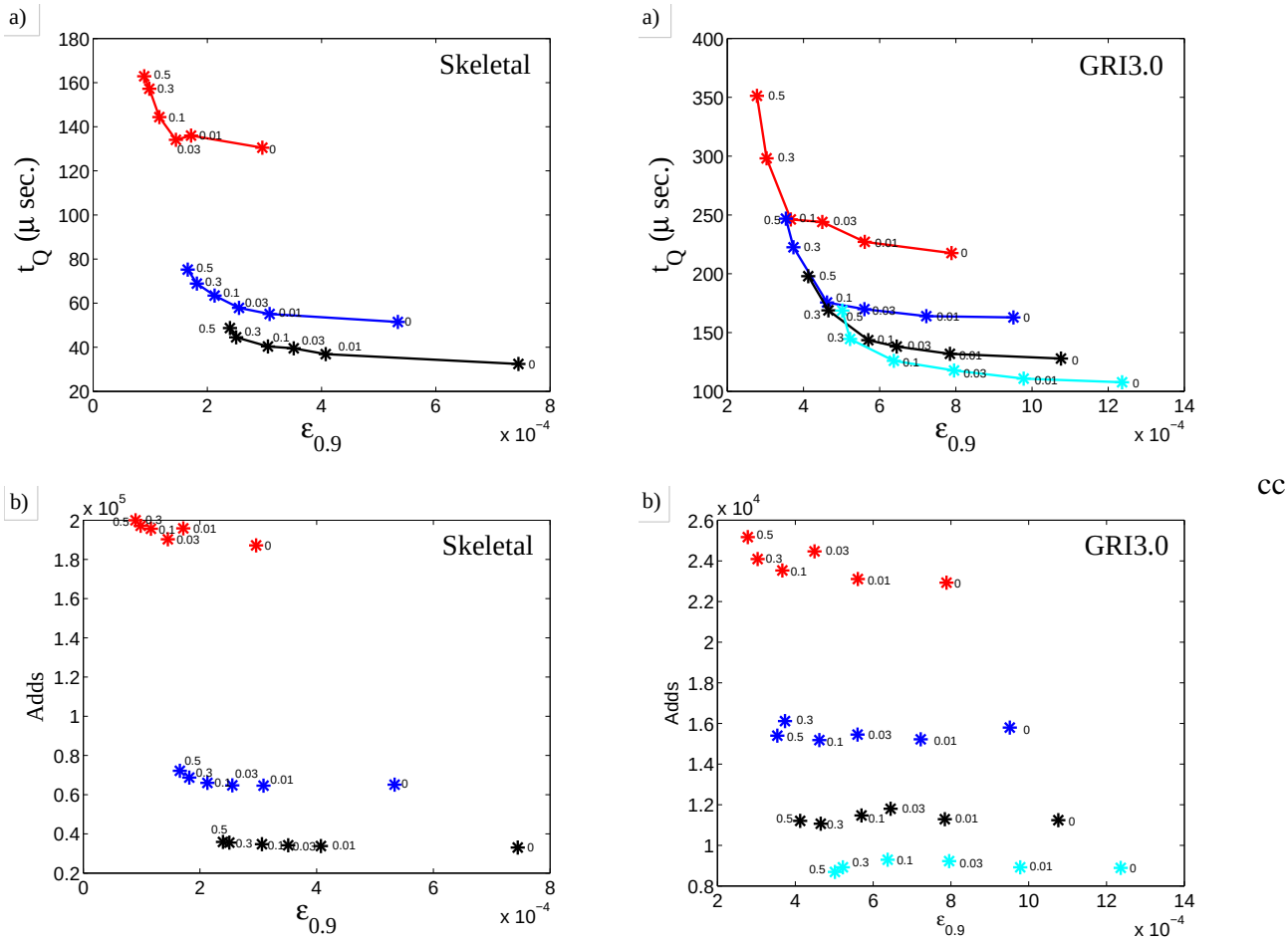


Fig. 16. For the PaSR test with the GRI3.0 mechanism, figure showing number of adds against 90% error of ISAT5 with ECC-Q. Error tolerance $\varepsilon_{tol} = 5 \times 10^{-4}$ (cyan $*$), 6×10^{-4} (black $*$), 7×10^{-4} (blue $*$), 8×10^{-4} (red $*$); the numbers 0, 0.01, 0.03, 0.1, 0.3, 0.5 indicate the value of ζ controlling the amount of ECC performed. The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each calculation results in 1.0×10^8 queries.



cc

Fig. 17. For the PaSR test, figure showing the effect of ECC-G on ISAT performance. Figures show $^{a)}$ CPU time (μs) per query, $^{b)}$ number of adds against 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 1 \times 10^{-4}$ (red $*$), 2×10^{-4} (blue $*$), 3×10^{-4} (black $*$); right column: GRI3.0 mechanism with $\varepsilon_{tol} = 3 \times 10^{-4}$ (red $*$), 4×10^{-4} (blue $*$), 5×10^{-4} (black $*$) and 6×10^{-4} (cyan $*$). The numbers 0, 0.01, 0.03, 0.1, 0.3 and 0.5 indicate the value of ζ controlling the amount of ECC performed. The number of table entries allowed is sufficiently large and the table is not full for all the calculations. Each results in 1.2×10^8 queries.

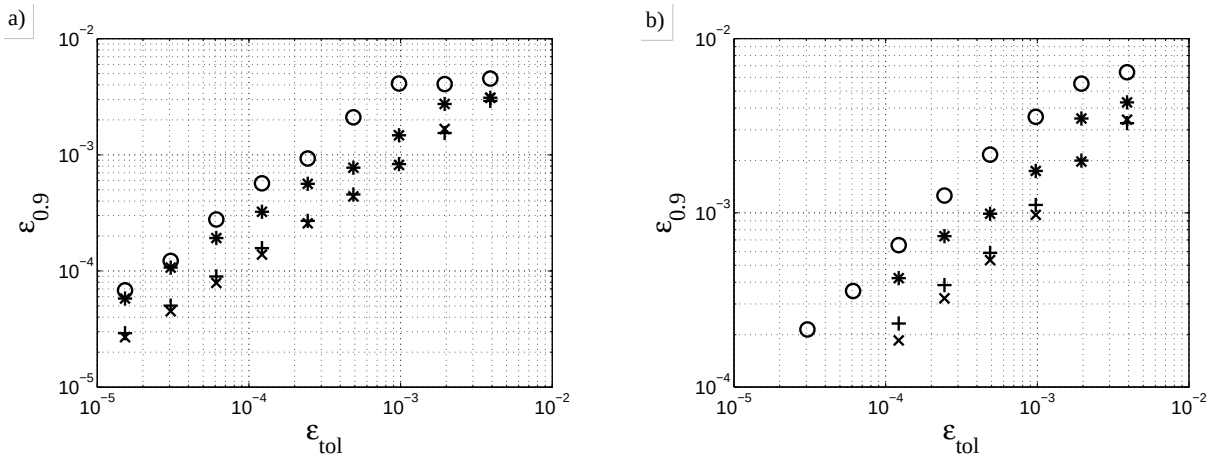


Fig. 18. The 90% error against the user-specified error tolerance from different implementations of ISAT for $^{a)}$ the skeletal mechanism and $^{b)}$ the GRI3.0 mechanism. Symbol $\circ$, ISAT4; $*$, ISAT5 without ECC; $+$, ISAT5 with ECC-Q ($\zeta = 0.1$, performed based on the average query time, t_Q); $\times$, ISAT5 with ECC-G ($\zeta = 0.1$, performed based on the number of grows). Each calculation results in 1.2×10^8 queries.

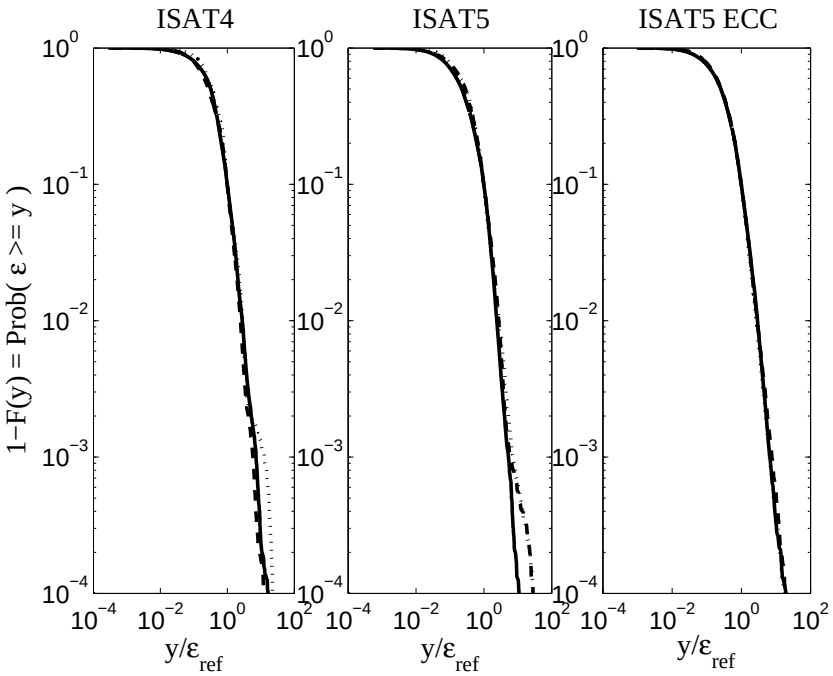


Fig. 19. CDF of local error against y/ε_{ref} with $\varepsilon_{ref} = \varepsilon_{0.9}$ from different implementations of ISAT for the GRI3.0 mechanism. Dotted line: $\varepsilon_{tol} = 2^{-8}$; dashed line: $\varepsilon_{tol} = 2^{-9}$; solid line: $\varepsilon_{tol} = 2^{-10}$. Each calculation results in 1.2×10^8 queries. The ECC shown is ECC-Q, which is based on the query time.

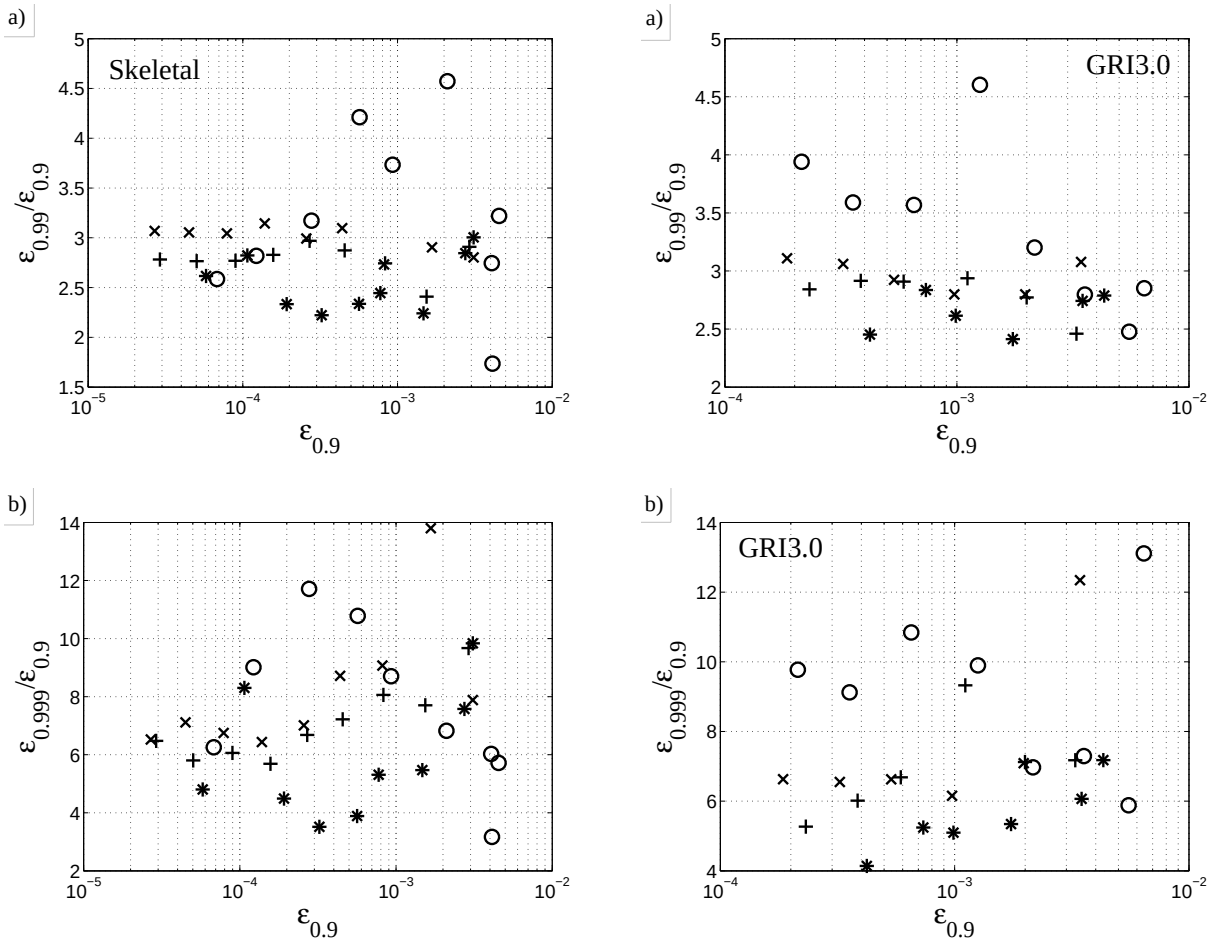


Fig. 20. For different implementations of ISAT, figure shows $^{a)}$ the ratio of $\varepsilon_{0.99}/\varepsilon_{0.9}$ and $^{b)}$ the ratio of $\varepsilon_{0.999}/\varepsilon_{0.9}$ against 90% error. Left column: skeletal mechanism; right column: GRI3.0 mechanism. Symbol $\circ$, ISAT4; $*$, ISAT5 without ECC; $+$, ISAT5 with ECC-Q ($\zeta = 0.1$); $\times$, ISAT5 with ECC-G ($\zeta = 0.1$). Each calculation results in 1.2×10^8 queries.

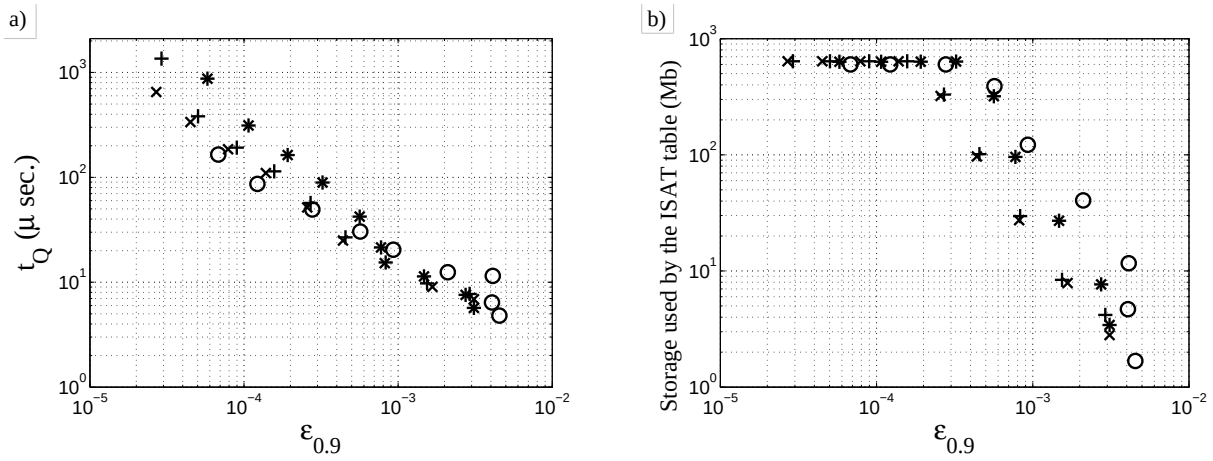


Fig. 21. For the PaSR test with the 16-species skeletal mechanism, figure showing of different ISAT implementations $^{a)}$ CPU time (μs) per query against 90% error; $^{b)}$ storage required (megabytes) against 90% error. Symbol $\circ$, ISAT4; $*$, ISAT5 without ECC; $+$, ISAT5 with ECC-Q ($\zeta = 0.1$); $\times$, ISAT5 with ECC-G ($\zeta = 0.1$). Each calculation results in 1.2×10^8 queries. The allowed storage of 600 megabytes was reached for some cases.

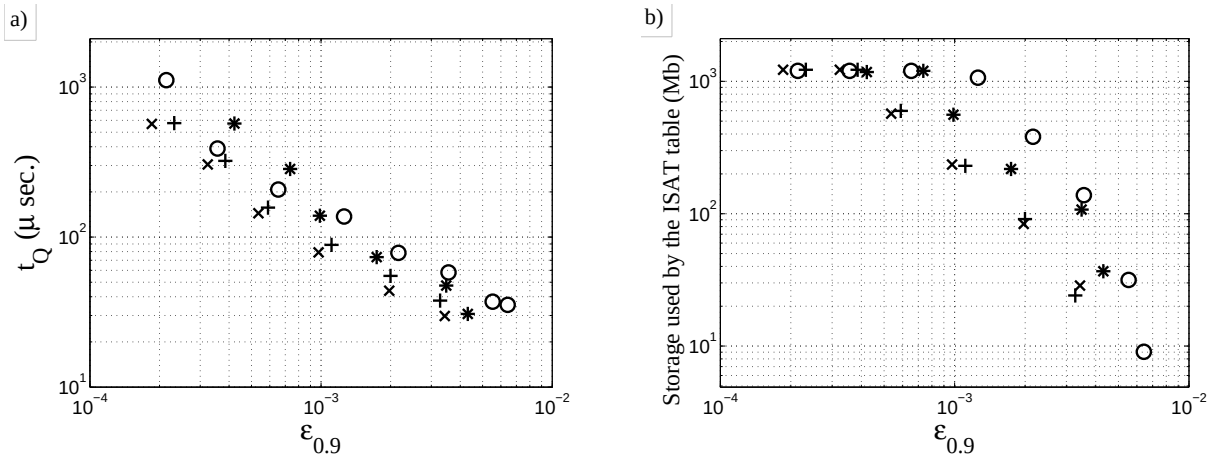


Fig. 22. For the PaSR test with the 53-species GRI3.0 mechanism, figure showing of different ISAT implementations $^{a)}$ CPU time (μs) per query against 90% error; $^{b)}$ storage required (megabytes) against 90% error. Symbol $\circ$, ISAT4; $*$, ISAT5 without ECC; $+$, ISAT5 with ECC-Q ($\zeta = 0.1$); $\times$, ISAT5 with ECC-G ($\zeta = 0.1$). Each calculation results in 1.2×10^8 queries. The allowed storage of 600 megabytes was reached for some cases.