Journal of Computational Physics 228 (2009) 361–386

Contents lists available at ScienceDirect

Journal of Computational Physics

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

An improved algorithm for *in situ* adaptive tabulation

Liuyan Lu *, Stephen B. Pope

Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA

ARTICLE INFO

Article history:
Received 14 July 2008
Accepted 12 September 2008
Available online 26 September 2008

PACS:
07.05.Mh
46.15.−x
47.11.−j

Keywords:
ISAT
Function approximation
Tabulation

ABSTRACT

In situ adaptive tabulation (ISAT) is a proven storage/retrieval method which efficiently provides accurate approximations to high-dimensional functions which are computationally expensive to evaluate. Previous applications of ISAT to computations of turbulent combustion have resulted in speed-ups of up to a thousand. In this paper, improvements to the original ISAT algorithm are described and demonstrated using two test problems. The principal improvements are in the table-searching strategies and the addition of an error checking and correction algorithm. Compared to an earlier version of ISAT, reductions in CPU time and storage requirements by factors of 2 and 5, respectively, are observed for the most challenging, 54-dimensional test problem.

1. Introduction

In computer simulations in different disciplines and applications, it is often required to evaluate a function (denoted by $\mathbf{f}(\mathbf{x})$) very many times, for different values of the input, $\mathbf{x}$. For example, if time-evolving partial differential equations are being solved by a finite-difference method, then some function $\mathbf{f}$ may need to be evaluated for each grid node on each time step based on the current values of the dependent variables at the nodes. If the function $\mathbf{f}$ is computationally expensive to evaluate, then these evaluations can dominate the overall computation cost, and it is natural to seek less expensive ways to evaluate $\mathbf{f}$, or to obtain an acceptably accurate approximation.

The above problem of the efficient approximation of a function $\mathbf{f}(\mathbf{x})$ is extremely general and can be quite varied depending on: the dimensionality, n_x, of the input $\mathbf{x}$; the dimensionality, n_f, of the output, $\mathbf{f}$; the distribution of the values of $\mathbf{x}$; the smoothness of the function $\mathbf{f}$; the number of evaluations required; the accuracy required; and the available computer memory. For low-dimensional problems (e.g. $n_x \leqslant 3$), often an effective strategy is to use structured tabulation. In a pre-processing stage, values of $\mathbf{f}(\mathbf{x})$ are tabulated; and then, during the simulation, approximate values of $\mathbf{f}$ are obtained by interpolation in the table. Both the work and the storage necessary to construct the table increase exponentially with n_x, which is why this approach is restricted to low-dimensional problems.

Since its introduction in 1997 [1] the *in situ* adaptive tabulation (ISAT) algorithm has proved very effective for a class of higher-dimensional problems (e.g. $n_x \leqslant 50$). The original application of ISAT was in the particle method used to solve the probability density function (PDF) equations for turbulent combustion [2]. In that case, $\mathbf{x}$ consists of the thermochemical state of a particle at the beginning of the time step (of duration Δt); and $\mathbf{f}$ represents the composition at the end of the step

* Corresponding author. Current address: GM R&D Center, Warren, MI 48090, USA.
E-mail address: ll267@cornell.edu (L. Lu).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.09.015

(resulting from adiabatic, isobaric reaction). Evaluating $\mathbf{f}(\mathbf{x})$ involves solving a stiff set of n_f ordinary differential equations (ODEs) for a time Δt. Typically this requires of order 10^4 μs of CPU time, whereas ISAT yields an accurate approximation in of order 10 μs, resulting in a thousandfold speed-up. Based on these timings, if $\mathbf{f}(\mathbf{x})$ needs to be evaluated for each of 10^6 particles on each of 10^3 time steps, then integrating the ODEs requires over 100 days of CPU time, whereas the same task is accomplished by ISAT in under 3 hours.

The ISAT algorithm continues to be heavily used for combustion problems both in the authors' group (e.g. [3–6]) and by others (e.g. [7–9]); and the algorithm has been incorporated into the ANSYS/FLUENT software [3,10]. ISAT has also been used in chemical engineering [11–13], in solid mechanics [14], control [15] and the study of thin films and surface reaction [16,17]. Several variants of ISAT have been proposed and investigated [18–24]. Other methods that have been used to address the same problem include: artificial neural networks [25–27]; high-dimensional model representation [28]; PRISM [29]; orthogonal polynomials [30]; and kriging [31].

The purpose of this paper is to describe and demonstrate several new additions to the original ISAT algorithm, which significantly enhance its performance. The next section provides an overview of the ISAT algorithm, and then the new components are described in Sections 3–5. In Section 6, quantitative testing of the algorithm is performed to demonstrate the efficacy of the new methods and to compare the computational performance with a previous implementation of ISAT. (Further tests are reported in [32,33].)

The ISAT algorithm makes extensive use of ellipsoids and hyper-ellipsoids (which, for simplicity, are referred to as ellipsoids, regardless of the dimension of the space). All of the algorithms used in ISAT involving ellipsoids are described in [34].

2. Overview of the ISAT algorithm

The purpose of ISAT is to tabulate a function $\mathbf{f}(\mathbf{x})$, where $\mathbf{x}$ and $\mathbf{f}$ are of dimension n_x and n_f, respectively. Given a *query*, $\mathbf{x}^q$, ISAT return $\mathbf{f}^a(\mathbf{x}^q)$ – an approximation to $\mathbf{f}(\mathbf{x}^q)$. It is assumed that $\mathbf{f}$ and $\mathbf{x}$ are appropriately scaled, with variations of order unity, so that the two-norm

$$\varepsilon = \|\mathbf{f}^a(\mathbf{x}^q) - \mathbf{f}(\mathbf{x}^q)\|, \tag{1}$$

is an appropriate measure of the approximation error. (There is no difficulty in using a more general definition of the error.)

An essential aspect of ISAT is that the table is built up, not in a pre-processing stage, but *in situ* (or "on line") as the simulation is being performed. At the beginning of the simulation the table is empty; and the table entries are added as needed based on queries, $\mathbf{x}^q$, generated by the simulation. In this way, only the region of the space which is accessed during the simulation is tabulated. In many multi-dimensional applications (e.g. combustion), this accessed region may be a tiny fraction of the whole space, and tabulating it is feasible, whereas tabulating the whole space is infeasible.

Table entries are referred to as *leaves* (since in the original algorithm they are leaves of a binary tree). The nth leaf includes: its location $\mathbf{x}^{(n)}$; the function value $\mathbf{f}^{(n)} = \mathbf{f}(\mathbf{x}^{(n)})$; and the $n_f \times n_x$ gradient matrix $\mathbf{A}^{(n)}$, which has components

$$A_{ij} = \frac{\partial f_i}{\partial x_j}. \tag{2}$$

This matrix is used to construct the *linear approximation* employed in ISAT. Given a query, $\mathbf{x}^q$, the linear approximation to $\mathbf{f}(\mathbf{x}^q)$ based on the nth leaf is defined as

$$\mathbf{f}^{l,n}(\mathbf{x}^q) = \mathbf{f}^{(n)} + \mathbf{A}^{(n)}(\mathbf{x}^q - \mathbf{x}^{(n)}); \tag{3}$$

and the error in this approximation is

$$\varepsilon^{(n)}(\mathbf{x}^q) = \|\mathbf{f}^{l,n}(\mathbf{x}^q) - \mathbf{f}(\mathbf{x}^q)\|. \tag{4}$$

Given a small error tolerance, ε_{tol}, ISAT returns approximations to $\mathbf{f}(\mathbf{x})$ with errors, ε, which are (with reasonable probability) less than ε_{tol}. An important concept, therefore, is the *region of accuracy* (ROA) of a leaf. For the nth leaf, the ROA is defined to be the connected region containing $\mathbf{x}^{(n)}$, within which the error $\varepsilon^{(n)}(\mathbf{x})$ in the linear approximation is less than the specified tolerance, ε_{tol}. In ISAT, the ROA is approximated by an ellipsoid, called the ellipsoid of accuracy (EOA). The EOA is initialized conservatively, and it may subsequently be "grown" (or otherwise modified) as additional information about the ROA is generated.

Briefly stated, the basic operations performed by ISAT on a query, $\mathbf{x}^q$, are the following:

(1) **Retrieve attempt**. A search is performed to identify any EOA which covers the query point. If such an EOA is found, the linear approximation to $\mathbf{f}(\mathbf{x}^q)$ based on that leaf is returned.
(2) **Grow attempt**. If the retrieve attempt is unsuccessful, then $\mathbf{f}(\mathbf{x}^q)$ is directly evaluated. Some number of leaves that in some sense are close to $\mathbf{x}^q$ are selected for *grow attempts*. For each of these selected leaves, the error ε in the linear approximation to $\mathbf{f}(\mathbf{x}^q)$ is evaluated, and if it is less than ε_{tol} then the leaf's EOA is *grown* to cover $\mathbf{x}^q$. If at least one grow attempt is successful, then $\mathbf{f}(\mathbf{x}^q)$ is returned.
(3) **Add**. If no grow attempt is successful, then a new leaf (at $\mathbf{x}^q$) is added to the table.

Several of the new contributions described in this paper pertain to the implementation of these processes, specifically:

(1) Searching for an EOA which covers $\mathbf{x}^q$.
(2) Searching for suitable leaves as candidates for growing.
(3) Modifying the leaf on a "grow".

In addition, a procedure for error checking and correction (ECC) has been incorporated. These new contributions are described in the subsequent sections.

3. Retrieve search

Given a query $\mathbf{x}^q$, the objective of the retrieve search is to identify any EOA which covers $\mathbf{x}^q$. There may be zero, one or more such EOAs. If at least one EOA covers $\mathbf{x}^q$, then the search process is deemed to be *complete* if it is guaranteed to identify one such EOA: otherwise it is *incomplete*. If the search identifies a covering EOA it is deemed to be successful. Any search strategy inevitably involves testing to determine whether or not a particular EOA covers the query point $\mathbf{x}^q$. This testing is described in Section 3.1. Then, in Section 3.2, we describe a strategy to reduce the associated computation cost which is based on projecting the problem onto a lower-dimensional affine space.

In subsequent subsections we describe the four search strategies implemented, which are denoted BT, MRU, MFU and EBT. Each has a different computational cost and a different probability of success. In Section 3.3 we consider the optimal use of a set of such methods, which leads to the concept of "effective cost."

3.1. Ellipsoid covering test

In the current implementation of ISAT, an ellipsoid E is represented by its center $\mathbf{c}$ (an n_x-vector) and an $n_x \times n_x$ lower-triangular Cholesky matrix $\mathbf{L}$, so that E is defined by

$$E \equiv \{\mathbf{x} \mid \|\mathbf{L}^T(\mathbf{x}-\mathbf{c})\| \leqslant 1\}. \tag{5}$$

Note that with $\mathbf{L} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ being the singular value decomposition (SVD) of $\mathbf{L}$, then the columns of the orthogonal matrix $\mathbf{U}$ give the directions of the principal axes of E, and the length of the principal semi-axes are given by σ_i^{-1}, where $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_{n_x}$ are the singular values (i.e., the components of the diagonal matrix $\boldsymbol{\Sigma}$). The radius of the largest ball covered by E is $r_{\text{in}} = 1/\sigma_1$; and the radius of the smallest ball covering E is $r_{\text{out}} = 1/\sigma_{n_x}$. In ISAT, the information stored characterizing the EOA of a leaf is: $\mathbf{c}, \mathbf{L}, r_{\text{in}}$ and r_{out} (where, for the nth leaf, $\mathbf{c} = \mathbf{x}^{(n)}$).

Given a point $\mathbf{x}$, the direct way to determine whether it is covered by an ellipsoid is to evaluate $\|\mathbf{L}^T(\mathbf{x}-\mathbf{c})\|$ (see Eq. (5)), which requires of order n_x^2 operations. For some cases, however, the question can be decided (in $\mathcal{O}(n_x)$ operations) by first evaluating $r \equiv \|\mathbf{x}-\mathbf{c}\|$ and comparing it to r_{in} and r_{out}. The direct test has to be performed only if r lies between r_{in} and r_{out}.

3.2. Affine space

The cost of the searching and testing may be reduced by introducing an appropriate n_a-dimensional affine space (for $1 \leqslant n_a < n_x$). Let $\overline{\mathbf{x}} = P(\mathbf{x})$ and $\overline{E} = P(E)$ denote the orthogonal projections of $\mathbf{x}$ and E onto the affine space. Of order (at most) n_a^2 operations are needed to determine if $\overline{E}$ covers $\overline{\mathbf{x}}$; and if it does not, then it follows that E does not cover $\mathbf{x}$. On the other hand, if $\overline{E}$ covers $\overline{\mathbf{x}}$, it does not follow that E covers $\mathbf{x}$ and so the test in the full-dimensional space (requiring $\mathcal{O}(n_x)$ or $\mathcal{O}(n_x^2)$ operations) must then be performed.

Over a large number N of test points, $\mathbf{x}$, let p denote the fraction of tests in which $\overline{E}$ covers $\overline{\mathbf{x}}$. Then the number of operations required by the above two-step procedure scales approximately as $N(n_a^2 + pn_x^2)$, compared to Nn_x^2 for the direct method. Thus, the relative cost is $(n_a/n_x)^2 + p$. As n_a increases from unity, the first contribution $(n_a/n_x)^2$ obviously increases, but the second decreases (for a sensible choice of the affine space), since the number of "false positives" decreases (i.e., tests in which $\overline{E}$ covers $\overline{\mathbf{x}}$, but E does not cover $\mathbf{x}$). As may be expected, therefore, for the test cases investigated, there is an optimal value of n_a (usually between 2 and 6) which minimizes the cost of searching.

In two tests reported in Section 6.3 with $n_x = 17$ and 54, the speed-ups in retrieving achieved by using the affine space are approximately 3 and 7.

In ISAT the affine space is formed, and periodically re-formed, based on a principal component analysis of the set of leaf locations $\{\mathbf{x}^{(n)}\}$. The affine space is spanned by the first n_a principal directions; and the value of n_a is specified to be as small as possible such that the r.m.s. deviation of $\{\mathbf{x}^{(n)}\}$ from the affine space is less than 10% of the r.m.s. variation in the first principal direction.

The EOAs projected onto the affine space are called PEOAs. Their geometries (i.e., the projections of $\mathbf{c}$ and $\mathbf{L}$ onto the affine space and the corresponding values of r_{in} and r_{out}) are stored for each leaf, and are re-calculated whenever the affine space is redefined.

3.3. Efficacy of multiple search attempts

As mentioned above, four search strategies, denoted BT, MRU, MFU and EBT (described below) are used successively in ISAT. The first three perform incomplete searches, whereas EBT performs a complete search. The search concludes when an EOA is found (by any of the methods) to cover the query point, or when the EBT search completes.

In this subsection, we abstract and analyze the following general question raised by the above use of multiple independent strategies. We consider three independent methods A, B and C which successfully perform a task with strictly positive probabilities p_A, p_B and 1, respectively, i.e., method C is guaranteed to succeed. The computational costs (e.g. CPU time) of the three methods are T_A, T_B, and T_C. The question is: which methods should be used, and in which order, so as to minimize the expected cost of successfully performing the task?

We denote by ABC the strategy of first using method A; if this is unsuccessful then using method B; and if this unsuccessful then using method C, etc. And we denote by T_{ABC} the expected cost of this strategy. Elementary probability theory yields

$$T_{ABC} = T_A + (1 - p_A)[T_B + (1 - p_B)T_C]. \tag{6}$$

It is clear that method C has to be used last, and hence the possible strategies to be considered are AC, BC, ABC and BAC. We define T_A/p_A to be the "effective cost" of method A. This is the expected cost per success of the method. And we define the "relative effective cost" of method A by

$$R_A \equiv \frac{T_A}{p_A T_C}. \tag{7}$$

This is the expected cost per success relative to the sure method, C.

Considering the strategy AC, we have

$$T_{AC} = T_A + (1 - p_A)T_C, \tag{8}$$

which can be manipulated to yield

$$T_{AC}/T_C = 1 - p_A(1 - R_A). \tag{9}$$

We conclude, therefore, that AC is less costly than C (i.e., $T_{AC} < T_C$) if and only if R_A is less than unity.

The relative merits of strategies ABC and AC are revealed by the relation

$$(T_{ABC} - T_{AC})/T_C = (1 - p_A)p_B(R_B - 1). \tag{10}$$

Thus, it is beneficial to add B to AC provided that R_B is also less than unity.

Comparing ABC with BAC we obtain

$$(T_{ABC} - T_{BAC})/T_C = p_A p_B(R_A - R_B), \tag{11}$$

showing that it is beneficial to invoke first whichever of A and B has the smaller effective cost.

From these examples and by induction we draw the following general conclusion. Given a set of independent methods, the least expected cost is achieved by using all methods that have a relative effective cost, R, less than unity, and to invoke them in order of increasing R.

(In this analysis it is assumed that methods A and B are *independent*, in the sense that the probability of success of B is independent of that of A. If this is not the case, then in Eq. (6) the unconditional probability p_B needs to be replaced by the conditional probability $p_{B|\bar{A}}$ of the success of B, given that A failed; and the result given in Eq. (11) is modified. Note that, if method B is simply a repetition of method A, then $p_{B|\bar{A}}$ is zero.)

3.4. Search methods

Four search methods are used in succession, with a different data structure used for each. (How these data structures are built is described in Section 5.) The first method uses a binary tree (BT) and is essentially the same as in the original implementation of ISAT. The nodes of the BT consist of cutting planes (in the n_x-dimensional space). Given a query, $\mathbf{x}^q$, the tree is traversed, until a leaf is reached: this is called the "primary leaf" for the given query. The EOA of the primary leaf is then tested to determine whether it covers the query point. If it does, then the entire search terminates successfully. If it does not, then the BT search is unsuccessful.

After an unsuccessful BT search, the projection of the query point onto the affine space is evaluated for use in the subsequent three search methods.

Two separate linked lists are maintained on the leaves. In the MRU list, the leaves are in the order in which they have most *recently* been used; and in the MFU list, the leaves are in the order in which they have most *frequently* been used. Here, "used" means used to retrieve from the table. Thus, the head of the MRU list is the leaf used on the last retrieve; and the head of the MFU list is the leaf used most often in previous retrieves.

A specified number, n_{MRU}, of leaves at the head of the MRU list are tested in succession to determine whether their EOA covers the query point (using the two-stage test based on the affine space). Then, similarly, a specified number, n_{MFU}, of the

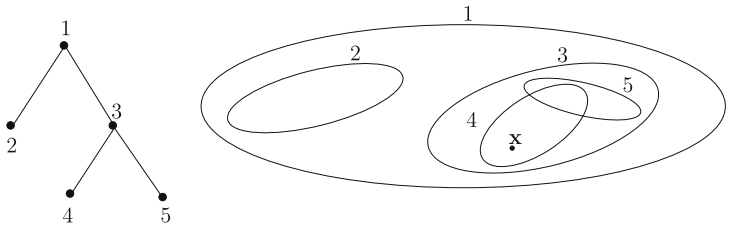


Fig. 1. Sketch of a very simple ellipsoidal binary tree (EBT) consisting of three leaves (2, 4 and 5) and two nodes (1 and 3). The point **x** is covered by the ellipsoid of leaf 4, which in turn is covered by the ellipsoids of its antecedent nodes (3 and 1).

leaves in the MFU list are tested (skipping any that were previously tested in the MRU scan). The search terminates successfully if and when an EOA covering the query point is found.

The likelihood of success of the MRU and MFU searches inevitably depends on the nature of the application. The MRU search is effective if, for example, there is substantial probability of successive queries being identical, or differing by a small amount such that the two query points are covered by the same EOA. The MFU search is effective if the use distribution of the leaves is highly non-uniform. In a representative test case (reported in [32,33]) with 2000 leaves, the single most frequently used leaf accounts for 10% of all retrieves, and the 50 most frequently used leaves account for 60% of the retrieves. In such circumstances the MFU method is certainly advantageous. While the optimal values of n_{MRU} and n_{MFU} are dependent on the application, the specifications $n_{\mathrm{MRU}} = 5$ and $n_{\mathrm{MFU}} = 30$ are found to yield good performance over the range of test cases investigated.

The data structure used in the final, complete search is an "ellipsoidal binary tree" (EBT) (the leaves of which correspond to the leaves of the BT). There is an ellipsoid associated with each node and each leaf of the EBT. For a leaf, this is the projected ellipsoid of accuracy (PEOA). For a node, as illustrated in Fig. 1, it is an ellipsoid which covers the ellipsoids of both of its children. If the projected query point is not covered by the ellipsoid of a node, it follows that the query point is not covered by any EOA in the sub-tree defined by the node: such a sub-tree is "infeasible" and can be eliminated from the search.

As depicted in Fig. 2, each node has a right child, a left child (with an ellipsoid associated with each), and a cutting plane. The normal distance from the cutting plane is denoted by s, with s being positive and negative, respectively, at the centers of the right and left ellipsoids. The maximum and minimum values of s for points in the right ellipsoid are denoted by $s_{r,\max}$ and $s_{r,\min}$; and for the left ellipsoid $s_{l,\max}$ and $s_{l,\min}$ are similarly defined. If the two ellipsoids do not intersect (as in Fig. 2a), then the cutting plane is taken to be a separating hyperplane, so that $s_{r,\min}$ is positive and $s_{l,\max}$ is negative. Conversely, if the two ellipsoids intersect (as in Fig. 2b), then there is overlap between the feasible ranges $[s_{l,\min}, s_{l,\max}]$ and $[s_{r,\min}, s_{r,\max}]$.

The EBT search starts at the primary leaf and proceeds by the following actions. At a leaf, test for the EOA covering the query point (using the two-stage PEOA/EOA test). If the EOA covers the query point, the search terminates successfully. Otherwise, move up the tree to the leaf's parent node. At a node, if both sub-trees incident on the node have already been tested, move up to the node's parent node, unless the node is the root, in which case the search terminates unsuccessfully. Otherwise, if the node's ellipsoid does not cover the query point move to the node's parent node; otherwise move to the right child (if $s > 0$ and the right sub-tree has not already been tested), or to the left child (if $s \leqslant 0$ and the left sub-tree has not already been tested, where s is the normal distance of the query point from the cutting plane).

Note that the testing at the nodes is performed in the affine space and hence requires at most of order n_a^2 operations. Further a screening test for s being in the range $[s_{r,\min}, s_{r,\max}]$ is performed in of order n_a operations.

As described above, the EBT search is complete: it is guaranteed to find an EOA covering the query point if one exists. For very large tables, such a complete search can be expensive, and its cost can be greater than directly evaluating the function $\mathbf{f}(\mathbf{x}^q)$. It is found to be beneficial overall to limit the extent of the EBT search so that on any query the CPU time used is less than a specified fraction, β_R of the average CPU time required to evaluate $\mathbf{f}(\mathbf{x}^q)$: the default value is $\beta_R = 0.5$.

4. EOA growing

If the retrieve attempt fails, then some leaves are selected as candidates for having their EOAs grown. We describe here the "grow search" performed to identify candidate leaves, and the subsequent EOA growth (and other modifications to the leaf). The treatment used is heuristic, and is the result of a good deal of experimentation with alternative implementations and variants. The fundamental reason for the need for heuristics is that the region of accuracy (ROA) is not in all circumstances well approximated by an ellipsoid. This point is now elaborated upon.

4.1. Region of accuracy

The fundamental issue involved is adequately illustrated by the simple case of a scalar function ($n_f = 1$) in two dimensions ($n_x = 2$). To leading order, the error in the linear approximation (Eq. (3)) to $\mathbf{f}(\mathbf{x}^q)$ based on the leaf at $\mathbf{x}^{(n)}$ is:

$$f^{l,n}(\mathbf{x}^q) - f(\mathbf{x}^q) = \frac{1}{2} H_{ij} r_i r_j, \tag{12}$$

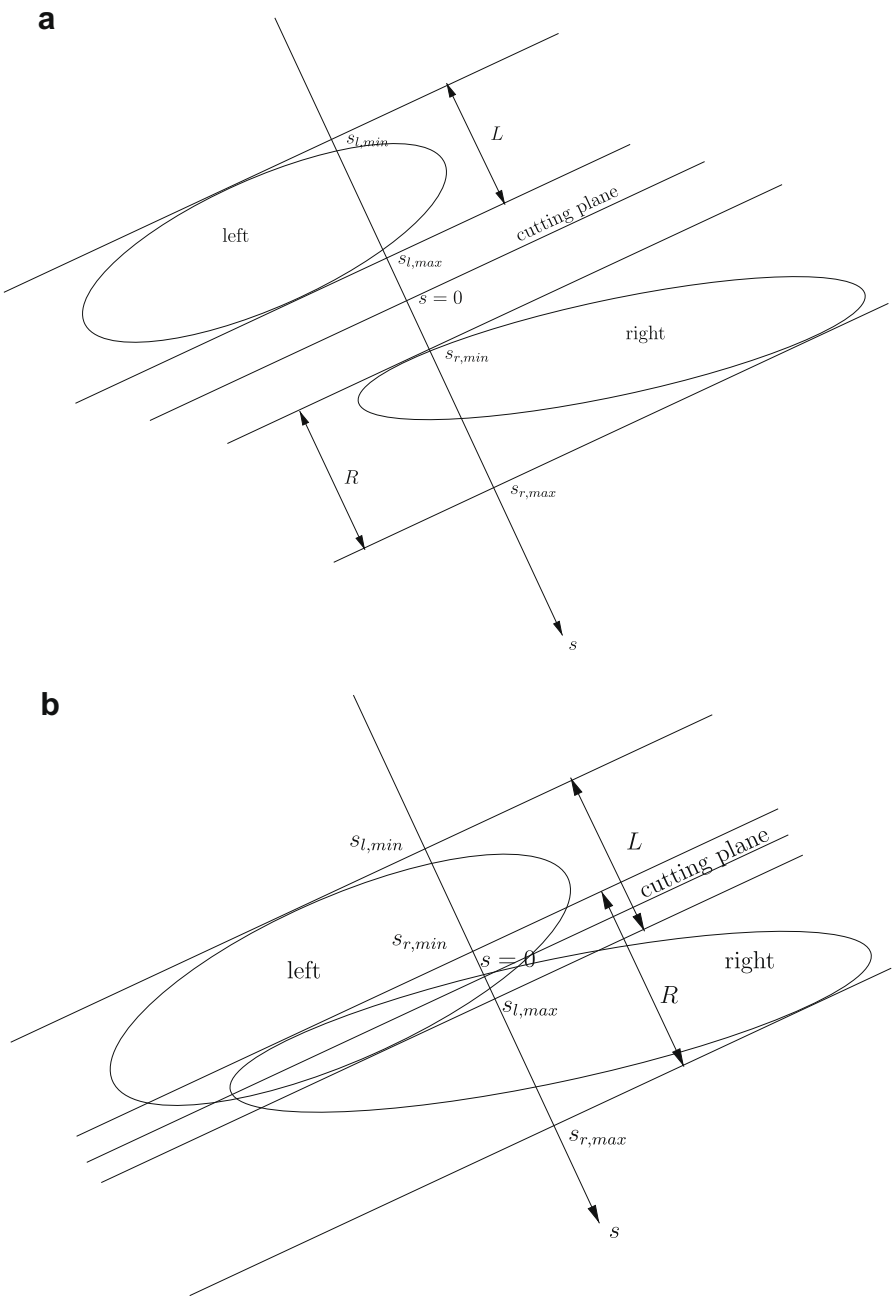


Fig. 2. Sketch of the cutting plane and the ellipsoids of the left and right children of a node of an EBT. The coordinate s measures distance normal to the cutting plane, and R and L show the ranges of s for points in the right and left ellipsoids, respectively. (a) Case in which the ellipsoids do not intersect and (b) case in which the ellipsoids intersect, and hence R and L overlap.

where $\mathbf{r}$ is the separation vector ($\mathbf{r} \equiv \mathbf{x} - \mathbf{x}^{(n)}$), and $\mathbf{H}$ is the symmetric Hessian matrix

$$H_{ij} \equiv \frac{\partial^2 f}{\partial x_i \partial x_j}. \tag{13}$$

Thus, the region of accuracy (ROA), R is (to leading order) given by

$$R = \left\{ \mathbf{r} \mid \frac{1}{2} \mid H_{ij} r_i r_j \mid < \varepsilon_{\text{tol}} \right\}. \tag{14}$$

The geometry of the ROA, R, is determined by the eigenvalues (λ_1 and λ_2) and the eigenvectors of $\mathbf{H}$. (Because $\mathbf{H}$ is symmetric, the eigenvalues are real, and the eigenvectors are orthogonal.) With $\widetilde{\mathbf{r}}$ being the components of $\mathbf{r}$ in the eigenvector basis, the ROA, Eq. (14), can be re-expressed as

$$R = \{\widetilde{\mathbf{r}} \mid\mid \lambda_1 \widetilde{r}_1^2 + \lambda_2 \widetilde{r}_2^2 \mid < 2\varepsilon_{\text{tol}}\}. \tag{15}$$

If both eigenvectors are of the same sign, then R is indeed an ellipse (with principal semi-axes $\sqrt{2\varepsilon_{\text{tol}}/\mid \lambda_{1,2} \mid}$). But if the eigenvalues have opposite signs, the ROA is hyperbolic (i.e., the bounding lines $\varepsilon(\mathbf{r}) = \varepsilon_{\text{tol}}$ are hyperbolae). In particular, we observe that along the lines $\widetilde{r}_2 = \pm\widetilde{r}_1\sqrt{\mid \lambda_1/\lambda_2 \mid}$, the error is zero.

Fig. 3 illustrates why a hyperbolic ROA causes problems for ISAT. As shown, the EOA is initially entirely within the ROA. However, when the EOA is grown based on the point $\mathbf{p}$ (at which the error is zero) it then includes significant regions where the error exceeds the tolerance. Hence a subsequent retrieve from a point such as $\mathbf{q}$ can result in an error ε substantially larger than ε_{tol}.

(Even for an elliptical ROA, the EOA can be grown to include regions outside the ROA. However, the extent of these regions and the problems they cause are much less than for hyperbolic ROAs.)

For the scalar, two-dimensional, hyperbolic case considered, the ROA is unbounded. In the vector case ($n_f > 1$), the ROA is unbounded only if the Hessian based on each component of $\mathbf{f}$ contains both positive and negative eigenvalues, and if there is an alignment between the hyperplanes on which the error in each component is zero. It may be, therefore, that in applications with moderate or large values of n_f, the ROAs encountered may be less pathological than the hyperbola in the scalar case.

4.2. Ellipsoid of inaccuracy (EOI)

In addition to an EOA (and PEOA), each leaf has an "ellipsoid of inaccuracy" (EOI) and its projection (PEOI) onto the affine space. In a quite approximate way, the EOA and EOI are considered as lower and upper bounds on the region of accuracy. A point, $\mathbf{x}$, covered by the EOA is deemed accurate (with respect to the linear approximation at $\mathbf{x}$ based on the leaf properties), whereas a point not covered by the EOI is deemed inaccurate. The EOI, which is concentric with the EOA, is initialized to be quite large (as described in Section 5.2) and is modified (as described later in this section) as further information about the leaf's region of accuracy is obtained.

The normal picture of the EOA and EOI is sketched in Fig. 4a: points covered by the EOA are deemed accurate, those not covered by the EOI are deemed inaccurate, and the accuracy of other points (covered by the EOI but not by the EOA) is deemed to be unknown. As mentioned, only to a limited extent do the EOA and EOI accurately bound the region of accuracy, and hence their names should not be take too literally. In particular, the case sketched in Fig. 4b can arise in which the EOI does not completely cover the EOA. This case is said to result in a "conflict" because, in a literal interpretation, the region covered by the EOA but not by the EOI is deemed to be both accurate and inaccurate, which is clearly contradictory.

4.3. Grow search

Based on a query $\mathbf{x}^q$, a leaf's EOA can be grown only if the linear approximation (at $\mathbf{x}^q$ based on the leaf properties) is accurate. The EOIs are used to determine leaves which are good candidates for growing. If an EOI covers $\mathbf{x}^q$, then the linear

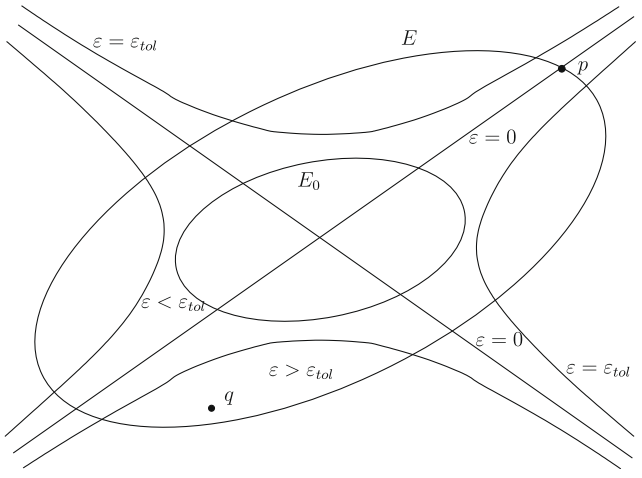


Fig. 3. Sketch of EOA growing for the case of a hyperbolic region of accuracy (ROA). The initial EOA, E_0, is entirely within the ROA, which is bounded by the hyperbolae on which $\varepsilon = \varepsilon_{\text{tol}}$. The EOA, E, which results from growing E_0 based on the accurate point $\mathbf{p}$, includes inaccurate regions, such as the point $\mathbf{q}$.

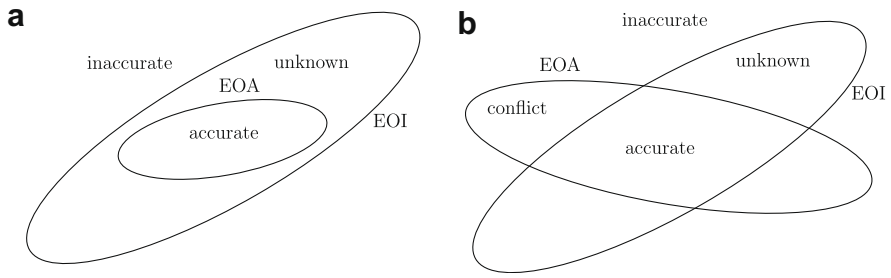


Fig. 4. Sketch of the EOA and EOI of a leaf for (a) the normal case in which the EOI covers the EOA and (b) the case of a conflict in which part of the EOA is not covered by the EOI.

approximation *may* be accurate, and the leaf is deemed to be a candidate for growing. On the other hand, if the EOI does not cover $\mathbf{x}^q$, then the linear approximation is estimated to be inaccurate, and the leaf is not a candidate for growing.

In order to identify candidate leaves, a "grow search" is performed to identify some or all of the leaves whose EOI covers $\mathbf{x}^q$. This task (for growing based on EOIs) is directly analogous to the retrieve search (for retrieving based on EOAs); and one of the same techniques is used, namely an EBT search.

A second ellipsoidal binary tree (EBT) is formed, in this case with the leaves corresponding to PEOIs (the EOIs projected onto the affine space). The searching of the EBT is performed in exactly the same way as in the EBT retrieve search. In this case the amount of searching is limited so that, on any query, the total amount of CPU time spent on grow attempts does not exceed a specified factor β_G of the average CPU time require to evaluate $\mathbf{f}(\mathbf{x})$: the default value is $\beta_G = 2$.

4.4. EOA growth and EOI modification

For each candidate leaf, the error ε in the linear approximation to $\mathbf{f}(\mathbf{x}^q)$ is measured and compared to the error tolerance, ε_{tol}. In the accurate case ($\varepsilon < \varepsilon_{\text{tol}}$) the EOA is grown; otherwise the EOA is not changed. As in the original ISAT algorithm, the grown EOA is uniquely defined as the ellipsoid of minimum content, with the same center, which covers both the original EOA and the query point $\mathbf{x}^q$.

The procedure to modify the EOI is more involved. Fig. 5 shows the ray from the center of the leaf $\mathbf{c}$ through the query point $\mathbf{x}^q$. We denote by s the distance along this ray (measure from $\mathbf{c}$), with s_q denoting the distance to $\mathbf{x}^q$. There is an unknown point $\mathbf{x}^a$, a distance s_a along the ray, at which the ray intersects the boundary of the region of accuracy (and hence $\varepsilon = \varepsilon_{\text{tol}}$ there). The EOI is modified based on a point $\mathbf{x}^m$, a distance s_m along the ray, where $\mathbf{x}^m$ is (as now explained) a conservative estimate of $\mathbf{x}^a$.

A Taylor-series analysis inevitably shows that the error $\varepsilon(s)$ along the ray varies as s^2 (to leading order). The error is known at $\mathbf{x}^q$ and hence $\varepsilon(s)$ can be estimated as

$$\varepsilon(s) \approx \varepsilon(s_q)\left(\frac{s}{s_q}\right)^2, \tag{16}$$

and hence s_a can be estimated (based on $\varepsilon(s_a) = \varepsilon_{\text{tol}}$) as

$$\frac{s_a}{s_q} \approx \left(\frac{\varepsilon_{\text{tol}}}{\varepsilon(s_q)}\right)^{\frac{1}{2}}. \tag{17}$$

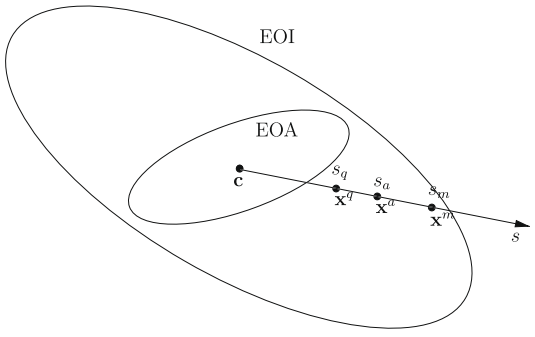


Fig. 5. Sketch showing the distance s from the center $\mathbf{c}$ of the EOI along the ray through the query point $\mathbf{x}^q$. In this case $\varepsilon(s_q)$ is less than ε_{tol}; $\mathbf{x}^a$ is the (unknown) point at which the error equals ε_{tol}; and $\mathbf{x}^m$ is the conservative estimate of $\mathbf{x}^a$ used to shrink the EOI.

In the accurate case ($\varepsilon(s_q) < \varepsilon_{\text{tol}}$), s_a is greater than s_q. In place of Eq. (17) the more conservative estimate

$$\frac{s_m}{s_q} = \min\left(\frac{\varepsilon_{\text{tol}}}{\varepsilon(s_q)}, 4\right), \quad \text{for } \varepsilon(s_q) < \varepsilon_{\text{tol}}, \tag{18}$$

is used. On the other hand, in the inaccurate case, s_a is less than s_q, and the more conservative estimate used is

$$\frac{s_m}{s_q} = \max\left(\left[\frac{\varepsilon_{\text{tol}}}{\varepsilon(s_q)}\right]^{\frac{1}{4}}, \frac{1}{4}\right), \quad \text{for } \varepsilon(s_q) \geqslant \varepsilon_{\text{tol}}. \tag{19}$$

These specifications of s_m define the point $\mathbf{x}^m$ which is then used to modify the EOI. If $\mathbf{x}^m$ is not covered by the EOI, then no modification is made. Otherwise, the EOI is "shrunk" using the "conservative algorithm" described in [34].

After all candidate leaves have been considered for modification, the EBT defined on the PEOAs is updated (if one or more EOA has been grown); and similarly the EBT defined on the PEOIs is updated (if one or more EOI has been shrunk).

4.5. Incurred error and ECC

In the testing of ISAT, the error ε incurred on a retrieve can be measured (by directly evaluating the function $\mathbf{f}(\mathbf{x}^q)$ based on the query $\mathbf{x}^q$). We then define the cumulative distribution function (CDF) $F(y)$ to be the fraction of retrieves on which the incurred error is less than y; and then its complement $\overline{F}(y) \equiv 1 - F(y)$ is the fraction of retrieves on which ε is greater than or equal to y.

Fig. 6 shows a plot of the complement of the CDF $\overline{F}(y)$ for a representative test case. As may be seen, the distribution does not have a sharp cut-off at $y = \varepsilon_{\text{tol}}$; but instead, larger errors occur with rapidly decreasing frequency. We take the 90% error $\varepsilon_{0.9}$ (such that $F(\varepsilon_{0.9}) = 0.9$) as a representative measure of the incurred error. The larger errors observed in Fig. 6 occur because some EOAs are inadvertently grown to include inaccurate regions. During the development of the present ISAT algorithm, very many variants of the algorithm were implemented and tested, and the influence of various parameters were explored. It is found that the shape of the error distribution, i.e., $F(y)$ plotted against $y/\varepsilon_{0.9}$, is remarkably insensitive to the details of the implementation of ISAT (see Fig. 19 in Section 6.8). However, these details can have a significant effect on the value of $\varepsilon_{0.9}/\varepsilon_{\text{tol}}$, on the table size, and on the CPU time required. It is essential to appreciate that the relative merits of different implementations must be examined for a fixed value of the incurred error $\varepsilon_{0.9}$, not for a fixed value of the error tolerance.

The technique of error checking and correction (ECC), suggested by B. Panda (private communication) and now described, is found to be particularly effective in reducing the incurred error $\varepsilon_{0.9}/\varepsilon_{\text{tol}}$ at a small computational cost. The idea of ECC is to check the incurred error occasionally, and if it exceeds the error tolerance, to shrink the EOA responsible for the inaccurate retrieve. Measuring the incurred error is an expensive process, since it requires the direct evaluation of the function $\mathbf{f}(\mathbf{x})$. In ISAT, retrieves are selected at random (by a Poisson process) for ECC treatment, with a probability such that, on average, the frequency of ECC events is controlled as described below. If the retrieve error is found to exceed the error tolerance ε_{tol}, then the EOA is shrunk to become the maximum content concentric ellipsoid, covered by the original EOA, which does not cover the query point.

Two variants of ECC (denoted ECC-Q and ECC-G) are examined. In ECC-Q, the ECC frequency is controlled so that the CPU time spent on ECC is a specified fraction ζ_Q of the total (with $\zeta_Q = 0.1$). In the ECC-G the ECC frequency is controlled so that the number of ECC events is a specified fraction ζ_G of the number of grow events (with $\zeta_G = 0.1$).

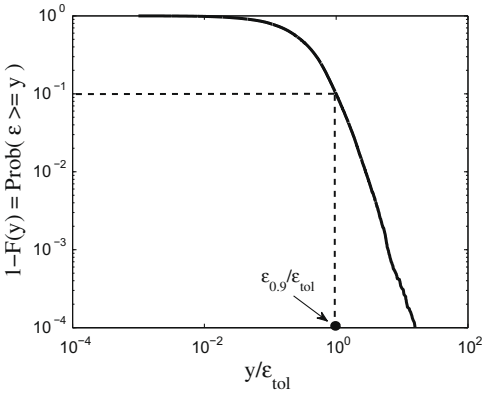


Fig. 6. Complement of the CDF of the local error $\overline{F}(y) = \text{Prob}\{\varepsilon \geqslant y\}$, showing the 90% error, $\varepsilon_{0.9}$. From the PaSR/GRI test with $\varepsilon_{\text{tol}} = 2^{-10}$ for ISAT5-Q. (See Section 6.8 for details.)

5. Adding

On a given query, if the retrieve and grow attempts fail, and if the table is not full, then a new entry is added to the table. (The treatment used when the table is full is discussed at the end of this section.) The primary operations involved in an "add" are the initialization of the new leaf (EOA, EOI, etc.) and its insertion into the various data structures. These operations are now outlined.

5.1. Initialization of the EOA

Let n be the index of the leaf to be added. Then its location $\mathbf{x}^{(n)}$ is that of the query, $\mathbf{x}^q$, and this is also the center of the EOA and EOI. The function value $\mathbf{f}^{(n)}$ is known from the grow attempt, and the gradient matrix $\mathbf{A}^{(n)}$ is evaluated.

The initial EOA is specified based on the region of accuracy of the *constant approximation* $\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}^{(n)})$. Let $\mathbf{r} \equiv \mathbf{x} - \mathbf{x}^{(n)}$ denote distance from the tabulation point. Then, to leading order, the error $\varepsilon_c(\mathbf{r})$ in the constant approximation is

$$\varepsilon_c = \|\mathbf{A}^{(n)}\mathbf{r}\|, \tag{20}$$

or equivalently

$$\varepsilon_c^2 = \mathbf{r}^T\mathbf{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{r}, \tag{21}$$

where $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \mathbf{A}^{(n)}$ is the SVD of $\mathbf{A}^{(n)}$.

The region of accuracy of the constant approximation is given by

$$R_c = \{\mathbf{r} \mid \varepsilon_c(\mathbf{r}) \leqslant \varepsilon_{\text{tol}}\}. \tag{22}$$

For the case in which there are n_x strictly positive singular values, σ_i, Eq. (21) shows that, to leading order, R_c is an ellipsoid: the principal semi-axes have lengths $1/\sigma_i$, and directions given by the columns of $\mathbf{V}$. If a singular value is zero (which is inevitably the case if n_f is less than n_x), then according to Eq. (21) the region of accuracy is unbounded in the corresponding direction.

Based on these observations, the EOA is initialized as an ellipsoid with principal semi-axes in the directions of the columns of $\mathbf{V}$, and of length

$$\lambda_i \equiv \min\left(1/\sigma_i, \alpha_{EOA}\varepsilon_{\text{tol}}^{\frac{1}{2}}\right), \tag{23}$$

with $\alpha_{EOA} = 0.1$ Note that the neglected quadratic term in the expansion for ε_c (Eq. (20)) adds a contribution of order $|\mathbf{r}|^2$, and the specification Eq. (23) ensures $|\mathbf{r}|^2 \leqslant \alpha^2\varepsilon_{\text{tol}}$ for all points in the initial EOA.

This specification of the initial EOA is very conservative in that it aims to limit the error in the constant approximation, whereas the more accurate linear approximation is used in retrieving. Tests of ISAT with growing suppressed confirm that the retrieving error from EOAs as initialized is extremely small – certainly less than ε_{tol}.

5.2. Initialization of the EOI

An upper bound on the length of the semi-axes of the EOI is specified as $r_{\max} = \alpha_{EOI}\varepsilon_{\text{tol}}^{\frac{1}{2}}$, with $\alpha_{EOI} = 10$. If the components of $\partial^2 f_i/\partial x_j \partial x_k$ are of order unity, then (to leading order) the error in the linear approximation is of order $|\mathbf{r}|^2$, and hence is of order $\alpha_{EOI}^2\varepsilon_{\text{tol}}$ on the boundary of the ball of radius $r_{\max}$ centered at $\mathbf{x}_{(n)}$.

Since the retrieve attempt failed, it is known that the linear approximations to $\mathbf{f}(\mathbf{x}^{(n)})$ based on all other (tested) leaves $m (m \neq n)$ are inaccurate. It is reasonable to suppose (especially for small ε_{tol}) that similarly the linear approximations to $\mathbf{f}(\mathbf{x}^{(m)})$ based on the leaf n (for all $m \neq n$) are inaccurate, and hence these other tabulated points are outside the region of accuracy of the nth leaf. Accordingly, the EOI is initialized by an approximation to the ellipsoid of maximum content, centered at $\mathbf{x}^{(n)}$, covered by the ball of radius $r_{\max}$, and which does not cover any other tabulation point.

This initialization of the EOI is accomplished as follows. A search on the binary tree of leaves is performed to identify (up to 1,000) tabulations points $\mathbf{x}^{(m)}$ that are covered by the ball of radius $r_{\max}$ centered at $\mathbf{x}^{(m)}$. This search is based on the observation that, at a node of the binary tree, if the ball is entirely on one side of the cutting plane, then the sub-tree lying on the opposite side of the cutting plane can be eliminated from the search. If more than 100 points are found, then only the 100 points closest to $\mathbf{x}^{(n)}$ are retained. The algorithm described in [34] is then used to construct the EOI as an ellipsoid centered at $\mathbf{x}^{(n)}$, which covers none of the retained points, and with principal semi-axes no larger than $r_{\max}$. (This algorithm involves a parameter θ, which is set to $\theta = 0.9$.)

Once the EOA and EOI have been initialized, all other leaf information to be stored is computed (e.g. the PEOA and PEOI).

5.3. Insertion in the data structures

Once all of the information pertaining to the added leaf has been computed, the leaf is inserted into each of the data structures.

The new leaf is inserted at the head of the MRU linked list and at the tail of the MFU linked list. These lists are updated in an obvious way whenever a leaf is used in a successful retrieve.

The new leaf is inserted into the binary tree (BT) by replacing the primary leaf by a node, whose two children are the new leaf and the primary leaf. The cutting plane at the node is set to be the perpendicular bisector of the line between the two leaves, in the linearly transformed space in which the EOA of the primary leaf is a ball. The BT data structure is unaffected by retrieving and growing: it changes only as a result of adding.

There are two ellipsoidal binary trees (EBTs), one defined on the PEOAs, the other on the PEOIs, and they both function in the same way. To insert a new leaf, the EBT is traversed (based on $\mathbf{x}^{(n)}$) to identify the new leaf's sibling. This leaf is then replaced by a node, whose two children are the new leaf and its sibling. In general, a node of an EBT includes a cutting plane and the related geometrical information discussed in Section 3.4 and shown in Fig. 2. The node also includes a "bounding ellipsoid," which covers the ellipsoids associated with both of its children. The bounding ellipsoid is formed using the "covariance algorithm" described in [34]. Whenever a PEOA or PEOI is added or modified (as a result of growing), all of its antecedent nodes are updated (i.e., the cutting planes and bounding ellipsoids of these nodes are re-formed).

5.4. Treatment of a full table

For challenging problems (especially for large n_x) the number of ISAT table entries can continually increase until the available memory is exhausted. Consequently, it is necessary to monitor and limit the table size, and to specify the action to be taken when a grow attempt fails and the table is full. The three actions considered are:

(a) Return the value of $\mathbf{f}(\mathbf{x}^q)$ (that is evaluated during the grow attempt) and do not alter the table.
(b) Delete the least recently used leaf (at the tail of MRU) and replace it by the new leaf.
(c) Delete the least frequently used leaf (at the tail of MFU) and replace it by the new leaf.

As with many aspects of ISAT, the best strategy depends on the nature of the problem. In particular, in this instance, it depends on whether or not the problem is statistically stationary, i.e., whether the distribution of queries $\mathbf{x}^q$ is fixed or evolving. For statistically stationary problems (a) appears optimal. In tests performed, it is generally found that the new leaf added in (b) or (c) provides no benefit compared to that deleted, but there is a substantial penalty in the cost of the subsequent growing of the added leaf. (It should be appreciated that the new, added leaf is likely to have a relatively small EOA (resulting from its initialization), whereas the deleted leaf is likely to be larger due to grows that it has experienced.)

For some non-stationary problems it is likely that (b) is optimal. In particular if the query distribution moves away from part of the tabulated region, and never returns, then the corresponding table entries can be deleted without penalty.

6. Performance of ISAT

Comprehensive testing of the ISAT algorithm has been performed for two test cases of a partially-stirred reactor (PaSR). The PaSR test cases are described in the first sub-section. Then test results are presented on the performance of ISAT, including the incurred error, CPU time per query, and number of table entries. Different variants of the ISAT implementation, and different values of various parameters are investigated, with the objectives of assessing the efficacy of different components of the algorithm and of determining near optimal values of the parameters. In the final sub-section, the performance of the new ISAT algorithm is compared to that of an earlier implementation.

6.1. The PaSR test case

The test case considered is a partially-stirred reactor (PaSR) involving the combustion of methane in air. This results in a time-dependent particle simulation which becomes statistically stationary. The function $\mathbf{f}(\mathbf{x})$, for which ISAT is employed, is the mapping between the thermochemical composition of a particle at the beginning of a time step ($\mathbf{x}$), and its value at the end of the time step ($\mathbf{f}$).

The PaSR has been used previously in many studies of combustion models and numerical algorithms [1,35–40]. The specific case considered is the non-premixed, adiabatic combustion of methane in air at atmospheric pressure. The pair-wise mixing model [1] is used, and this is specifically designed to provide a broad distribution of compositions ($\mathbf{x}$), and hence a good test case for ISAT. The computations performed in the PaSR test are similar to those for a single cell in a PDF or LES/FDF simulation of turbulent combustion, and hence the performance of ISAT in the PaSR test has a direct bearing on its performance in these applications.

The PaSR evolves in time, t, in discrete time steps of size Δt. At time t, the reactor consists of an even number N of particles, the nth of which has the thermochemical composition $\boldsymbol{\phi}^{(n)}(t)$. The n_ϕ components of this composition vector $\boldsymbol{\phi}$ are the specific moles of the n_s chemical species and the enthalpy (i.e., $n_\phi = n_s + 1$). At the beginning of each time step, events occur corresponding to *outflow*, *inflow* and *pairing* which cause the ensemble of particles to change discontinuously. Between these

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
Sub-step	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

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

$$\frac{\mathrm{d}\boldsymbol{\phi}^p}{\mathrm{d}t} = -(\boldsymbol{\phi}^p - \boldsymbol{\phi}^q)/\tau_{\text{mix}}, \tag{24}$$

$$\frac{\mathrm{d}\boldsymbol{\phi}^q}{\mathrm{d}t} = -(\boldsymbol{\phi}^q - \boldsymbol{\phi}^p)/\tau_{\text{mix}}, \tag{25}$$

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

$$\frac{\mathrm{d}\boldsymbol{\phi}^{(n)}}{\mathrm{d}t} = \mathbf{S}(\boldsymbol{\phi}^{(n)}), \tag{26}$$

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

At the discrete times $k\Delta t$, with τ_{res} being the specified residence time, outflow and inflow consist of selecting $\frac{1}{2}N\Delta t/\tau_{\text{res}}$ 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, $\frac{1}{2}N\Delta t/\tau_{\text{pair}}$ 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 sub-steps of duration $\Delta t_{\text{sub}} = \Delta t/N_{\text{sub}}$ are taken for reaction and mixing. During each sub-step of duration Δt_{sub}, the composition evolves by a mixing step (of size $\Delta t_{\text{sub}}/2$), followed by a reaction step (of size Δt_{sub}), and then by another mixing step (of size $\Delta t_{\text{sub}}/2$).

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 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. Initially ($t = 0$), all particle compositions are set to be the pilot-stream composition. The pressure is atmospheric throughout.

Two different chemical mechanisms are used to describe the combustion of methane, specifically to determine the net creation rates $\mathbf{S}$ of the species (in Eq. (26)). The first is the skeletal mechanism used in [41], which involves $n_s = 16$ species; the second is the more complex GRI 3.0 mechanism [42] which involves $n_s = 53$ species.

ISAT is used to perform each reaction sub-step for each particle. The input $\mathbf{x}$ is the scaled composition of the particle at the beginning of the sub-step ($x_\alpha = \phi_\alpha^{(n)}(t_0)/\phi_{\alpha,\text{ref}}$); and the output is the composition at the end of the sub-step $f_\alpha = \phi_\alpha^{(n)}(t_0 + \Delta t_{\text{sub}})/\phi_{\alpha,\text{ref}}$. Here $\phi_{\alpha,\text{ref}}$ is a reference value, chosen to represent the variation of ϕ_α, so that the variations in $\mathbf{x}$ and $\mathbf{f}$ are of order unity.

The direct evaluation of $\mathbf{f}(\mathbf{x})$ requires the integration of the stiff set of ordinary differential equations, Eq. (26). This is accomplished using the ODE solver DDASAC [43].

To perform the PaSR simulation for one residence time requires 30,000 ISAT queries (i.e., 300 sub-steps for 100 particles). The PaSR becomes statistically stationary after about three residence times, about 10^5 queries. Most of the tests reported below are for about 4000 residence times (1.2×10^8 queries), and hence are dominantly in the statistically stationary state.

6.2. Effect of the limit on table size

The difficulty of a tabulation problem, and hence the performance of ISAT, depends on many factors (e.g. n_x, n_f, ε_{tol}). Among these factors are the total number of queries Q and the maximum allowed table size A. (Here we measure A in table entries (i.e., leaves), but it can also be measured in megabytes.) In this sub-section we illustrate the effect of A on the performance of ISAT.

For a problem with Q queries and a limit of A on the allowed number of ISAT table entries, let $p_R(Q,A), p_G(Q,A), p_A(Q,A)$ and $p_D(Q,A)$ denote the fraction of queries resulting in retrieves, grows, adds and direct evaluations (which are required when the table is full if the retrieve and grow attempts fail). For the PaSR test case, since it is random, p_R, p_G, p_A and p_D can also be viewed as probabilities. These fractions (or probabilities) sum to unity, i.e., we have

$$p_R(Q,A) + p_G(Q,A) + p_A(Q,A) + p_D(Q,A) = 1. \tag{27}$$

In the hypothetical case of infinite storage, adding is always possible and hence no direct evaluations are required, i.e., $p_D(Q,\infty) = 0$.

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 have $p_R + p_D \approx 1$, and the average CPU time for a query, t_Q, can be well approximated as

$$\begin{aligned} t_Q &\approx t_R p_R(Q,A) + t_D p_D(Q,A) \\ &\approx t_R + p_D(Q,A)(t_D - t_R), \end{aligned} \tag{28}$$

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. (28) 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. (28), 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. (28) is negligible.

For a particular PaSR test, Fig. 7 shows the average CPU time per query, and the fraction of direct evaluations against the number of table entries allowed. 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 (which is the only case in which the table is not full). 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, $A \geqslant 30,000$), 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, and for a given test case (with a specified error tolerance and a specified number of queries), we can define the critical number of ISAT table entries, A^*, implicitly by

$$p_D(Q,A^*) = \frac{t_R}{t_D - t_R}. \tag{29}$$

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

$$\frac{t_Q}{t_R} = 1 + \frac{p_D(Q,A)}{p_D(Q,A^*)}. \tag{30}$$

Evidently the *storage ratio* $s \equiv A/A^*$ determines the effectiveness of ISAT. For $s \geqslant 1$, ISAT is very effective and $t_Q/t_R \leqslant 2$, i.e., within a factor of 2 of the ideal performance. For $s < 1$, the time spent on direct evaluations is significant. For the calculation shown in Fig. 7, the value of t_R is about 40 μs, the value of t_D is about 5000 μs, and the critical number of table entries, A^*, is about 28,000. As may be seen from Fig. 7, 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

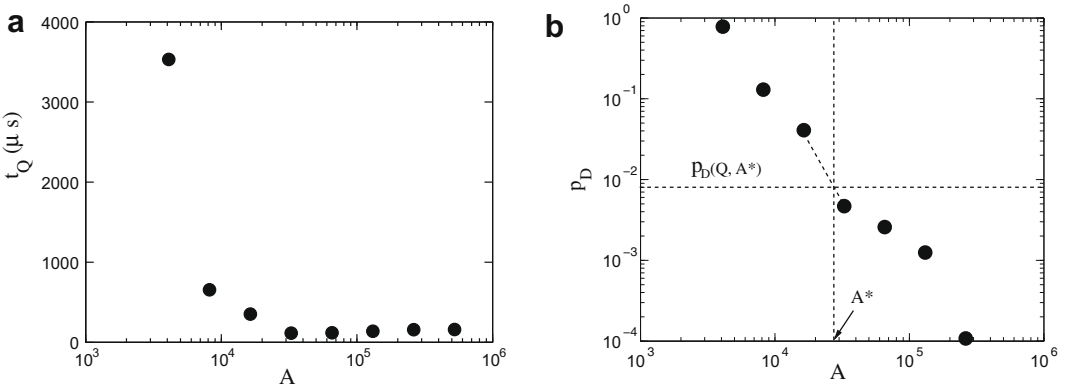


Fig. 7. For the PaSR test with the 16-species skeletal mechanism (a) the average CPU time μs per query; (b) the fraction of direct evaluations against the number of table entries allowed in ISAT. The error tolerance is $\varepsilon_{tol} = 1 \times 10^{-4}$ and the calculation results in 1.5×10^8 queries.

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 problems where large storage for ISAT is desirable. For a given table storage, an implementation with a smaller value of A^* results in a smaller fraction of direct evaluations and thus achieves higher computational efficiency.

6.3. Effect of the dimensionality of the affine space

As described in Section 3.2, an n_a-dimensional affine space is employed in searching EOAs (for retrieving) and EOIs (for growing). A series of PaSR tests are performed to examine the effect of the value of n_a on different aspects of ISAT's performance, and the results are shown in Fig. 8. These test are performed using both the skeletal and GRI mechanisms; using three values of the error tolerance ε_{tol}; and using different fixed values of n_a as well as the value determined by a principal com-

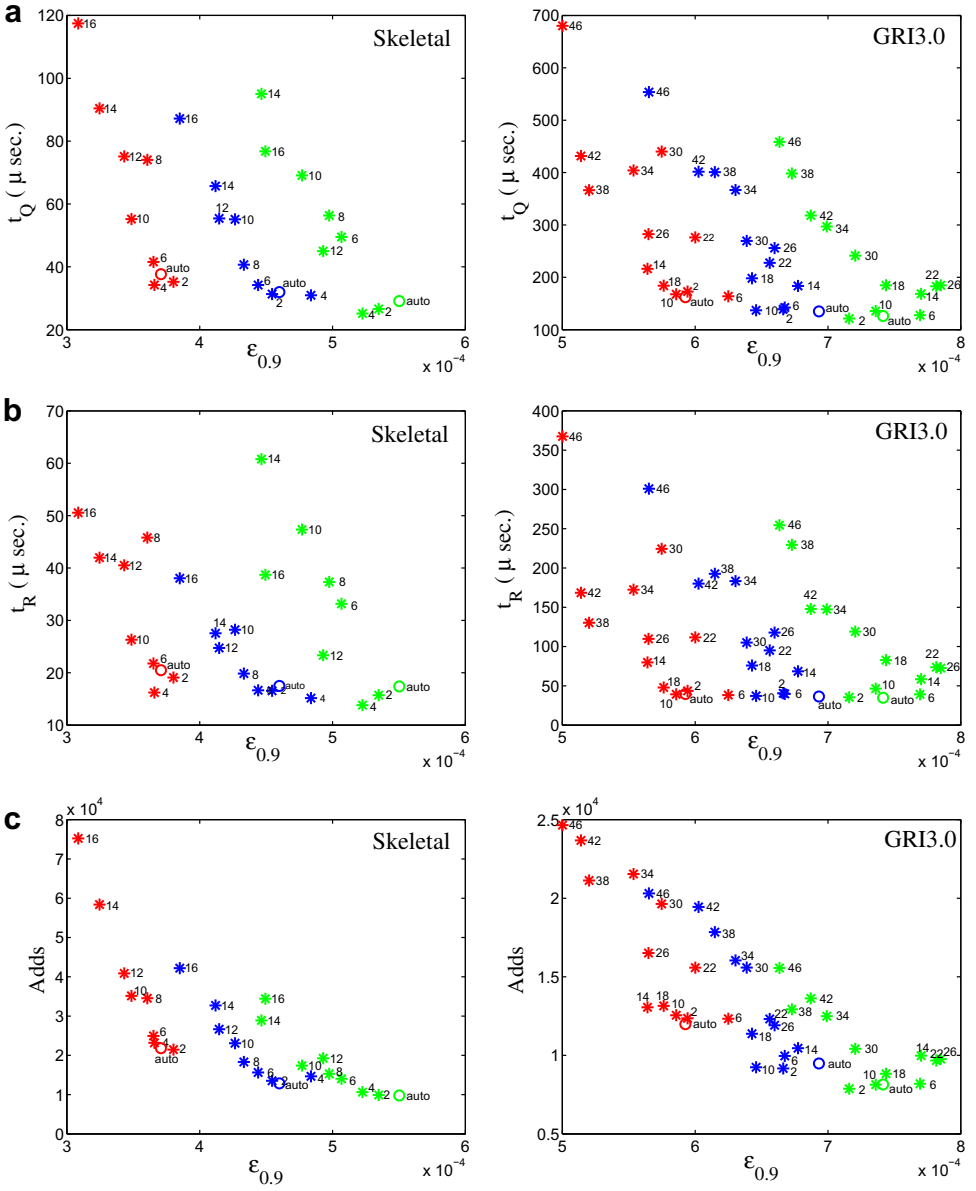


Fig. 8. For the PaSR test, the effect of the dimensionality of the affine space n_a on ISAT performance: (a) CPU time μs per query; (b) CPU time μs per retrieve, and (c) number of adds (or table entries) against the 90% error. Left column: skeletal mechanism with $\varepsilon_{\text{tol}} = 4 \times 10^{-4}$ (red), $\varepsilon_{\text{tol}} = 5 \times 10^{-4}$ (blue) and $\varepsilon_{\text{tol}} = 6 \times 10^{-4}$ (green); right column: GRI3.0 mechanism with $\varepsilon_{\text{tol}} = 5 \times 10^{-4}$ (red), $\varepsilon_{\text{tol}} = 6 \times 10^{-4}$ (blue) and $\varepsilon_{\text{tol}} = 7 \times 10^{-4}$ (green). Symbol $*$: implementation with fixed values of n_a (indicated by the numbers on the plots); $\circ$: 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

ponent analysis (as described in Section 3.2). Each test results 1.2×10^8 queries, and in no case in the ISAT table full. The quantities shown in Fig. 8 are: the average time for a query, t_Q, and the average time spent on retrieving, t_R; the number of entries added to the table, A; and the incurred error, $\varepsilon_{0.9}$.

The first point to appreciate from Fig. 8 is that the incurred error $\varepsilon_{0.9}$ depends significantly on the value of n_a as well as on ε_{tol}. For example, in the GRI test (right column of Fig. 8), the incurred error for $\varepsilon_{\text{tol}} = 5 \times 10^{-4}$ and $n_a = 6$, is greater than for the larger error tolerance $\varepsilon_{\text{tol}} = 6 \times 10^{-4}$ and $n_a = 46$. The emphasizes the need to compare performance at the same value of $\varepsilon_{0.9}$, not ε_{tol}.

It is clear from Fig. 8 that relatively small fixed values of n_a can lead to a three-fold decrease in t_Q and t_R (at fixed $\varepsilon_{0.9}$). For the skeletal mechanism ($n_x = n_f = 17$), the values $n_a = 2$ and $n_a = 4$ yield comparably good performance, whereas for the GRI mechanism ($n_x = n_f = 54$) good performance is observed for $n_a \leqslant 10$.

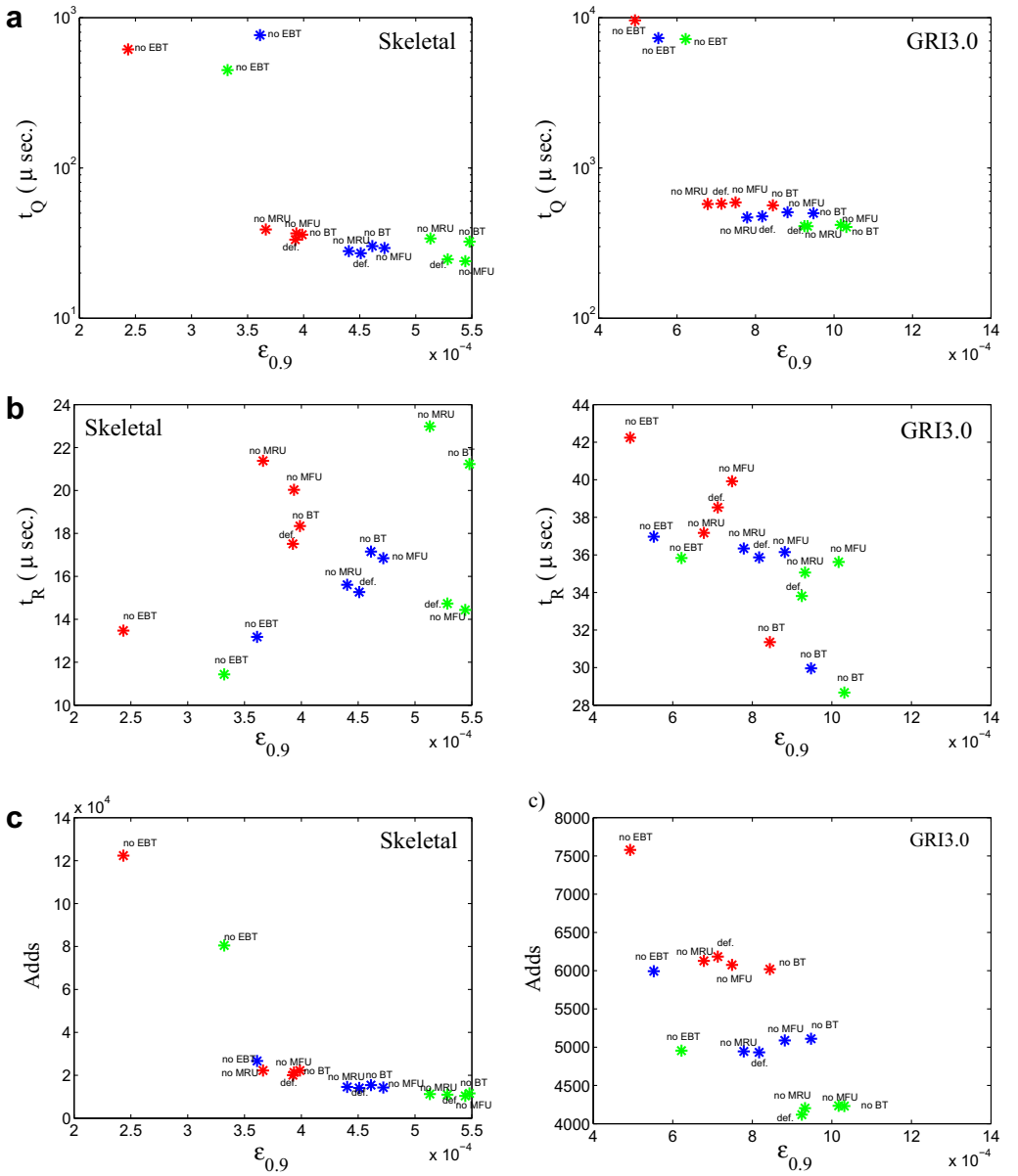


Fig. 9. For the PaSR test, the effect of different retrieve operations on ISAT performance: (a) CPU time μs per query, (b) CPU time μs per retrieve, and (c) number of adds against the 90% error. Left column: skeletal mechanism with $\varepsilon_{\text{tol}} = 4 \times 10^{-4}$ (red), $\varepsilon_{\text{tol}} = 5 \times 10^{-4}$ (blue), $\varepsilon_{\text{tol}} = 6 \times 10^{-4}$ (green), and each calculation results in 1.2×10^8 queries; right column: GRI3.0 mechanism with $\varepsilon_{\text{tol}} = 5 \times 10^{-4}$ (red), $\varepsilon_{\text{tol}} = 6 \times 10^{-4}$ (blue), $\varepsilon_{\text{tol}} = 7 \times 10^{-4}$ (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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

It may also be observed that the relatively small values of n_a lead to fewer table entries (for the same incurred error, $\varepsilon_{0.9}$). This is because the more efficient searching has a higher probability of success (given that the amount of searching allowed is limited by the CPU time expended). By using a small value of n_a, the table size can be reduced typically by a factor of two. This can have a yet more favorable impact on performance for cases in which the table becomes full.

In nearly all of these test cases, the automatic determination of the affine space using principal component analysis yields a value of n_a less than 5. As may be seen from Fig. 8, this automatically-determined value is close to optimal with respect to all quantities examined (t_Q, t_R and A).

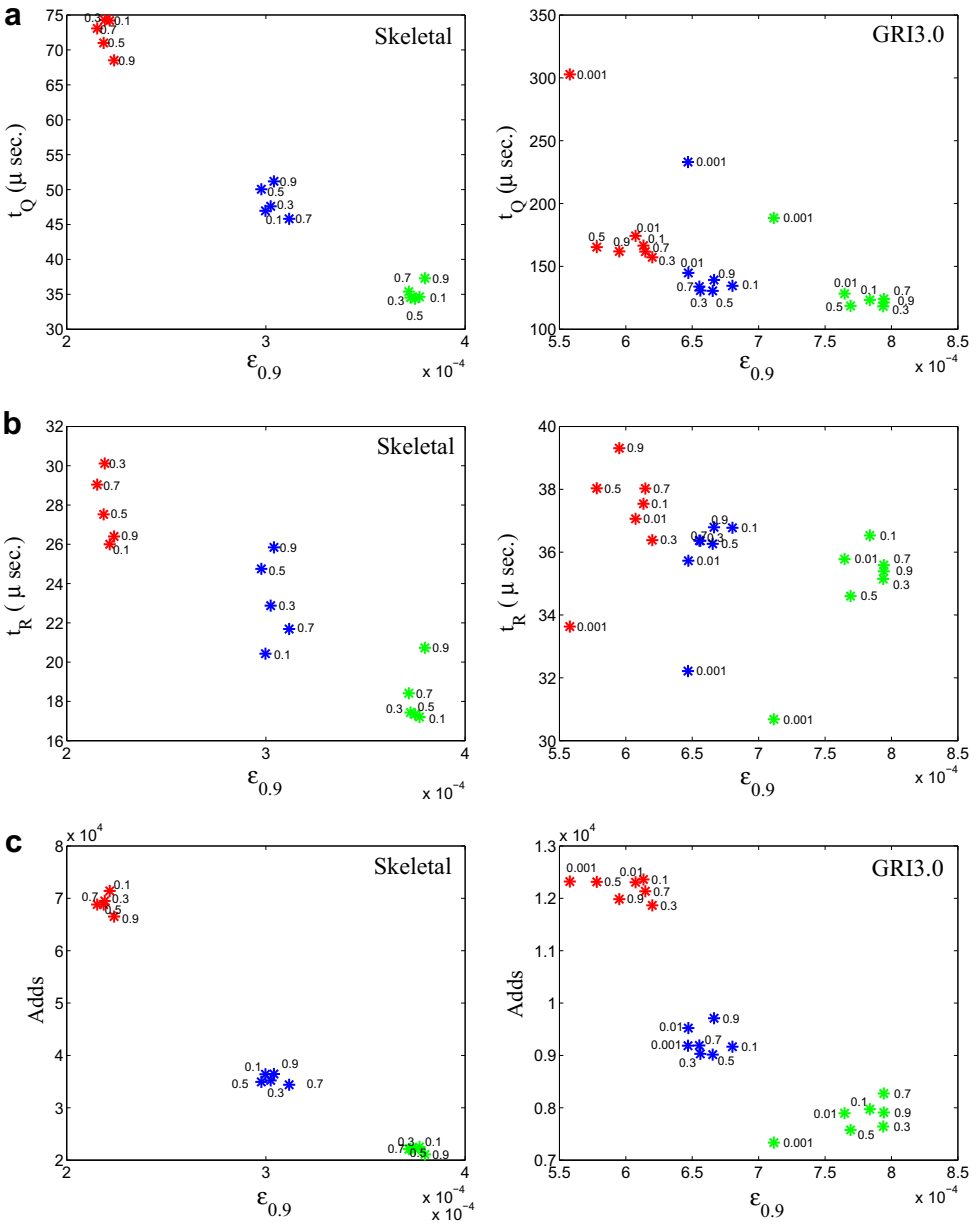
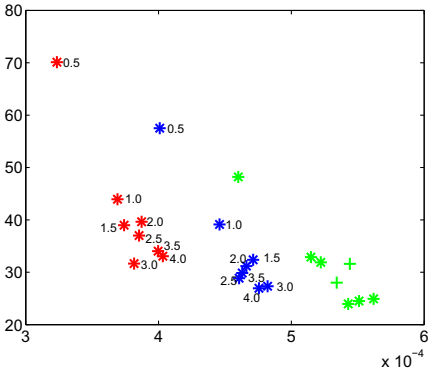


Fig. 10. For the PaSR test, the effect of parameter β_R on ISAT performance: (a) CPU time μs per query and (b) CPU time μs per retrieve, and (c) number of adds against the 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 2 \times 10^{-4}$ (red), $\varepsilon_{tol} = 3 \times 10^{-4}$ (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 numbers next to the symbols show the value of β_R. 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

6.4. Effect of operations and parameters controlling retrieve attempts

As described in Section 3.4, a succession of methods – BT, MRU, MFU, EBT – are used to attempt to retrieve. In Fig. 9 we investigate the effectiveness of these methods by suppressing them one at a time. Thus, for example, "no BT" indicates the case in which BT is suppressed, whereas "def" denotes the default case in which all methods are used.

The most striking observation from Fig. 9 is the beneficial effect of EBT: without it (indicated by "no EBT"), query times increase by an order of magnitude (albeit with a slightly reduced incurred error). The effects of other methods are much smaller and somewhat varied. Nevertheless, the default (of using all methods) is optimal (or near optimal) with respect to query time and table size in all cases.



The amount of EBT searching on each query is limited so that the CPU time consumed is no greater than a factor β_R of the average CPU time for a function evaluation. By default β_R is set to 0.5. In Fig. 10 we investigate the effect of the parameter β_R on performance. As may be seen, with the very small value $\beta_R = 0.001$ there is a clear degradation in performance. For β_R between 0.1 and 0.9 there is a modest dependence of query time and table size in β_R, with $\beta_R = 0.5$ generally being near optimal.

6.5. Effect of the parameter controlling EOA growth

The ellipsoids of accuracy (EOAs) are initialized conservatively. The subsequent growth of the EOAs is essential to the efficiency of ISAT, but it can also result in large incurred errors.

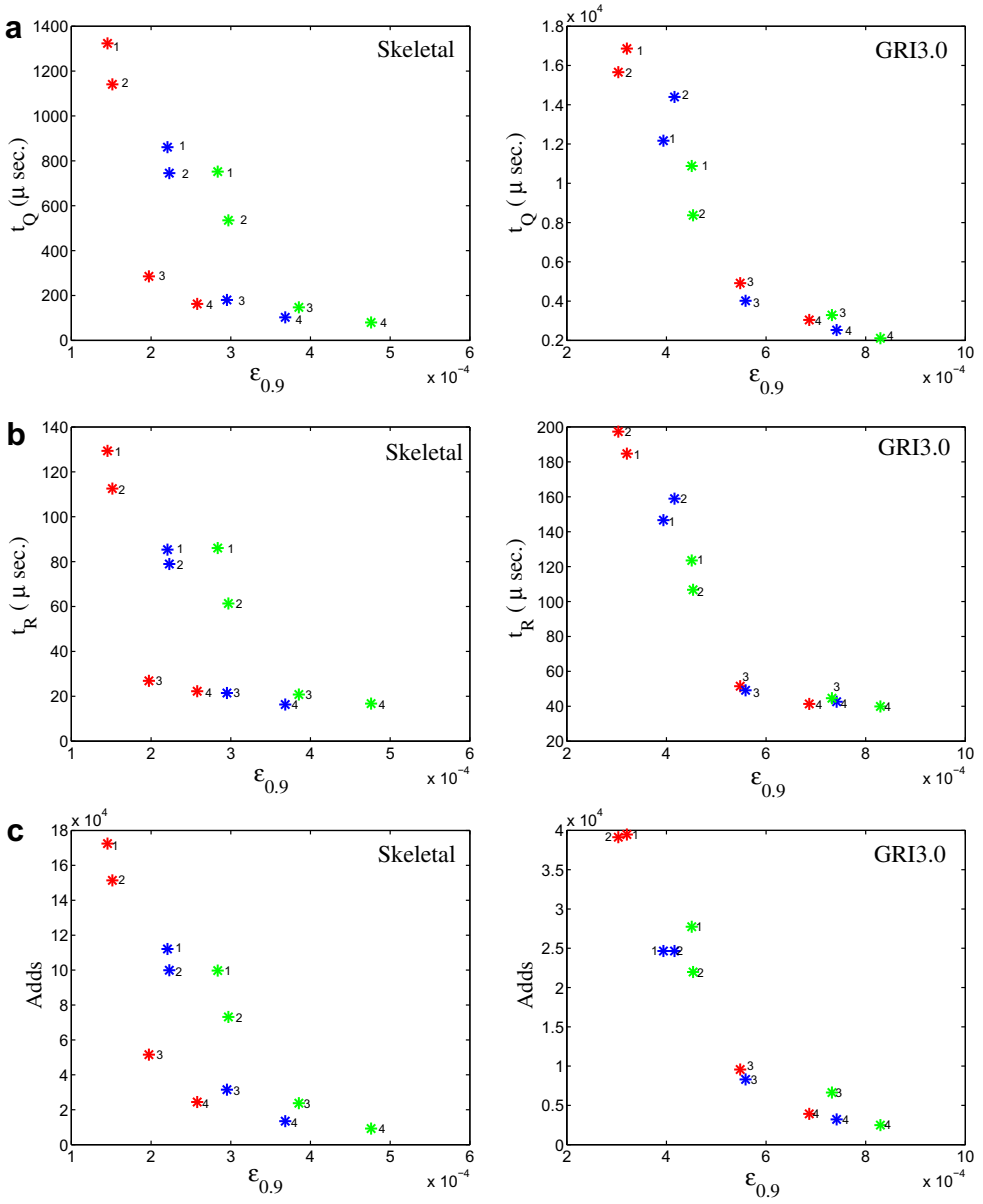


Fig. 12. For the PaSR test, the effect of different strategies (mode=1, 2, 3, or 4) taken to resolve the conflict between EOAs and EOIs: (a) CPU time μs per query, (b) CPU time μs per retrieve, and (c) number of adds against the 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 2 \times 10^{-4}$ (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 numbers next to the symbols indicate the mode used. 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

As described in Section 4, in a given query, if the retrieve attempts fail, then the EBT of PEOIs is traversed to identify a set of leaves whose EOIs cover the query point. Then, for each leaf so identified, the EOI or the corresponding EOA is modified. The amount of CPU time spent on this search is limited to be no greater than a factor β_G of the average time of a function evaluation.

Fig. 11 shows the effects of different values of β_G on the performance of ISAT. It is clear that increasing β_G from 0.5 to 1.0 is significantly beneficial in reducing both average query time and the number of table entries. The values $\beta_G = 2, 3$ and 4 all appear to be close to optimal, and superior to $\beta_G = 1$. The default value of $\beta_G = 2$ yields an average query time within 25% of optimal for all cases studied.

6.6. Effect of the treatment of EOA/EOI conflicts

When a leaf is initialized, the EOI covers the EOA. Subsequently, after the EOA is grown, or the EOI is modified, it is possible that part of the EOA is no longer covered by the EOI, a situation which is referred to as a "conflict": the uncovered part of the EOA is deemed to be both accurate and inaccurate. We regard a conflict, not as a contradiction or inconsistency, but as a manifestation of the shortcomings of the EOA and EOI to represent the regions of accuracy and inaccuracy.

We have investigated four different strategies for treating conflicts, which are referred to as modes 1, 2, 3 and 4. They are defined as follows:

1. Restore the EOA to its state immediately prior to the conflict; do not allow further growing of the EOA (i.e., on subsequent queries).
2. Shrink the EOA to be covered by the EOI; do not allow further growing of the EOA.
3. Shrink the EOA to be covered by the EOI; allow further growing of the EOA.
4. Do not modify the EOA (nor the EOI), so that the conflict remains; allow further growing of the EOA.

Fig. 12 shows the performance of ISAT using each of these modes. It is clear that, as may be expected, modes 1 and 2 result in the smallest errors, but at a considerable cost in CPU time per query and number of table entries. For a fixed value of the incurred error, $\varepsilon_{0.9}$, it is clear that mode 4 is best in yielding the smallest query times and table sizes, with mode 3 being marginally inferior. Accordingly, mode 4 is taken as the default.

6.7. Effect of the frequency of error checking and correction

As described in Section 4.5, on randomly selected queries resulting in successful retrieves, error checking and correction (ECC) 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 $\mathbf{x}^q$. ECC is an expensive process due to the extra direct evaluations required. Two variants, denoted ECC-Q and ECC-G, have been implemented and tested in which the frequency of ECC is controlled relative to queries or grow events, respectively.

In ECC-Q, the frequency of ECC is controlled so that the CPU time consumed in ECC is less than a specified fraction ζ_Q of the total CPU time. Fig. 13 shows results demonstrating the performance of ISAT with different values of ζ_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 $\varepsilon_{\text{tol}} = 1 \times 10^{-4}, 2 \times 10^{-4}, 3 \times 10^{-4}$ and $\zeta_Q = 0, 0.01, 0.03, 0.1, 0.3, 0.5$. For $\varepsilon_{\text{tol}} = 3 \times 10^{-4}$ and $\zeta_Q = 0$, the incurred error $\varepsilon_{0.9}$ is a little less than 10^{-3}. Increasing ζ_Q from 0 to 0.01 results in more than halving $\varepsilon_{0.9}$ with a negligible CPU penalty. As ζ_Q is increased, $\varepsilon_{0.9}$ decreases, but the CPU time increases. Notice that

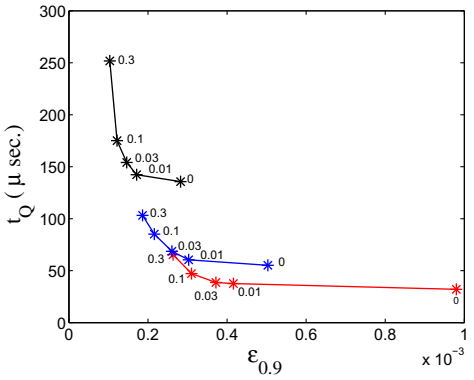


Fig. 13. For the PaSR test with the 16-species skeletal mechanism, the CPU time μs per query against the 90% error of ISAT with ECC-Q. Error tolerance $\varepsilon_{\text{tol}} = 1 \times 10^{-4}$ (black $*$), 2×10^{-4} (blue $*$), 3×10^{-4} (red $*$); the numbers 0, 0.01, 0.03, 0.1, 0.3 indicate the value of ζ_Q 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

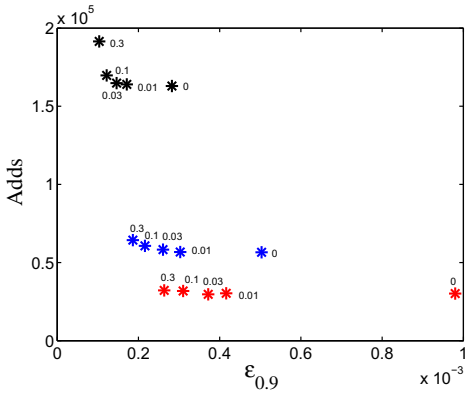


Fig. 14. For the PaSR test with the 16-species skeletal mechanism, the number of adds against the 90% error of ISAT with ECC-Q. Error tolerance $\varepsilon_{tol} = 1 \times 10^{-4}$ (black $*$), 2×10^{-4} (blue $*$), 3×10^{-4} (red $*$); the numbers 0, 0.01, 0.03, 0.1, 0.3 indicate the value of ζ_Q 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the optimal performance corresponds to the envelop of the different curves. The optimal value of ζ_Q 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 $\zeta_Q = 0$, i.e, without ECC).

Fig. 14 demonstrates the effect of ECC on 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 $\zeta_Q = 0$, $\varepsilon_{tol} = 2 \times 10^{-4}$ with $\zeta_Q = 0.03$, and $\varepsilon_{tol} = 3 \times 10^{-4}$ with $\zeta_Q = 0.3$; however the storage required is in the ratio 4:2:1.

Figs. 15 and 16 show the corresponding results for a test case using the GRI mechanism. There are in total 24 separate tests corresponding to all combinations of $\varepsilon_{tol} = 5 \times 10^{-4}, 6 \times 10^{-4}, 7 \times 10^{-4}, 8 \times 10^{-4}$ and $\zeta_Q = 0$, 0.01, 0.03, 0.1, 0.3, 0.5. For $\varepsilon_{tol} = 8 \times 10^{-4}$ and $\zeta_Q = 0$, the incurred error $\varepsilon_{0.9}$ is a little less than 1.6×10^{-3}. Increasing ζ_Q 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_Q is increased, $\varepsilon_{0.9}$ decreases, but the CPU time increases, which illustrates the computational penalty incurred to achieve high accuracy. The optimal value of ζ_Q is around 0.1. Fig. 16 demonstrates the effect of ECC on the table size. As observed for the skeletal mechanism, the use of ECC can reduce by about a factor of 2 the number of table entries required for a given incurred error.

As shown above, ECC-Q (with the ECC frequency based on the total CPU time) achieves a significant gain in computational performance for the cases where a sufficiently large storage is allowed for the ISAT table. However, with limited storage allowed for the ISAT, some pathological cases are observed where, due to the excessive shrinking of EOAs caused by ECC, an undesirable sharp increase in direct function evaluations is observed after the table is full, which severely degrades the overall computational performance. To remedy this problem, the second variant, ECC-G, is introduced in which the frequency of ECC is controlled so that the number of ECC events is less than a specified fraction ζ_G of the number of grow events.

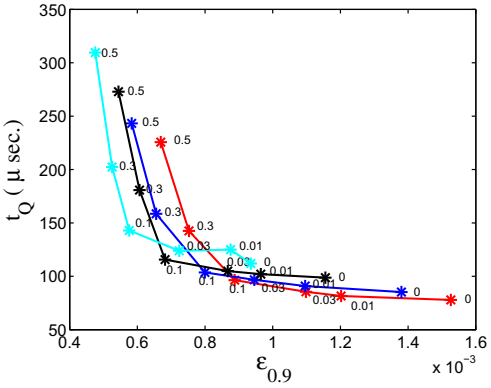


Fig. 15. For the PaSR test with the GRI3.0 mechanism, CPU time (μs) per query against the 90% error of ISAT with ECC-Q. Error tolerance $\varepsilon_{tol} = 5 \times 10^{-4}$ (cyan $*$), 6×10^{-4} (black $*$), 7×10^{-4} (blue $*$), 8×10^{-4} (red $*$); the numbers 0, 0.01, 0.03, 0.1, 0.3, 0.5 indicate the value of ζ_Q 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

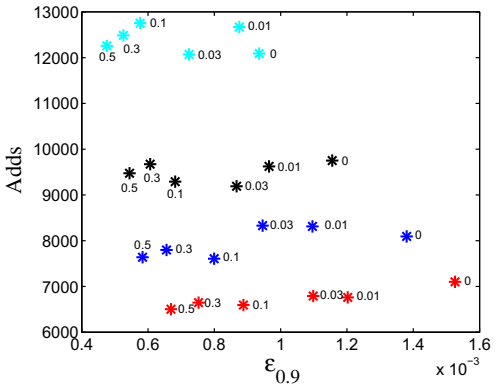


Fig. 16. For the PaSR test with the GRI3.0 mechanism, the number of adds against the 90% error of ISAT with ECC-Q. Error tolerance $\varepsilon_{tol} = 5 \times 10^{-4}$ (cyan $*$), 6×10^{-4} (black $*$), 7×10^{-4} (blue $*$), 8×10^{-4} (red $*$); the numbers 0, 0.01, 0.03, 0.1, 0.3, 0.5 indicate the value of ζ_Q 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

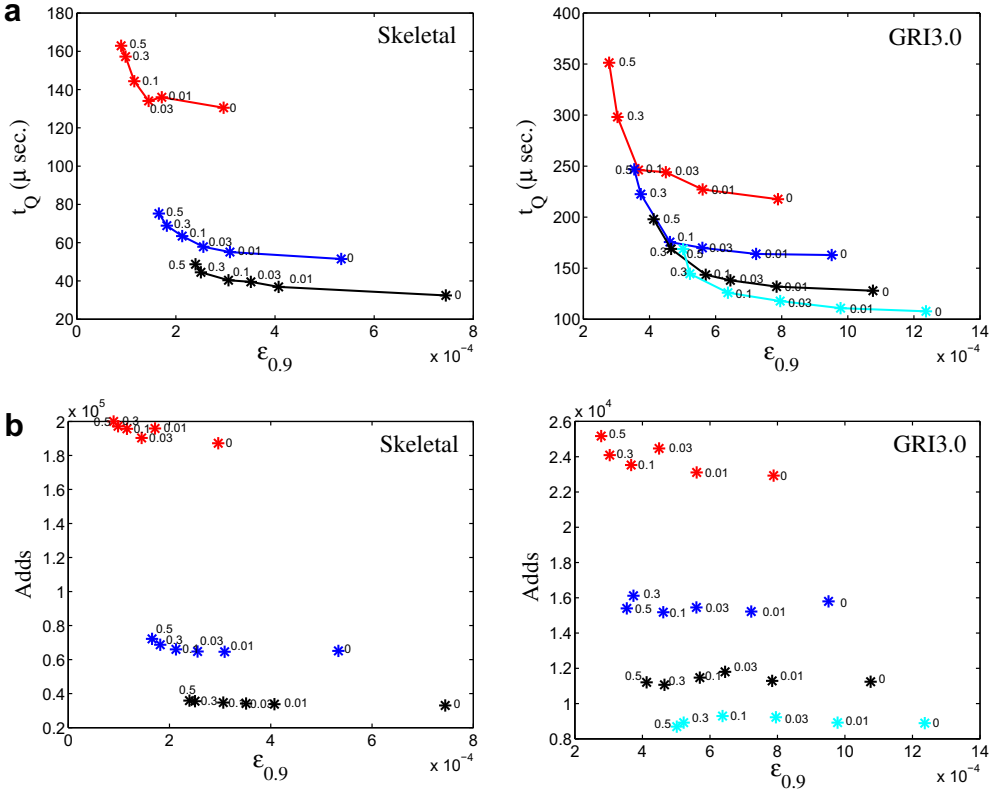


Fig. 17. For the PaSR test, the effect of ECC-G on ISAT performance: (a) CPU time μs per query and (b) number of adds against the 90% error. Left column: skeletal mechanism with $\varepsilon_{tol} = 1 \times 10^{-4}$ (red $*$), 2×10^{-4} (blue $*$), 3×10^{-4} (black $*$); right column: GRI3.0 mechanism 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_G 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 17 shows results demonstrating the performance of ISAT ECC-G for different values of ζ_G. 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 $\varepsilon_{tol} = 1 \times 10^{-4}$, $2 \times 10^{-4}, 3 \times 10^{-4}$ and $\zeta_G = 0, 0.01, 0.03, 0.1, 0.3, 0.5$. For the GRI3.0 mechanism, there are 24 separate tests corresponding to all combinations of $\varepsilon_{tol} = 3 \times 10^{-4}, 4 \times 10^{-4}, 5 \times 10^{-4}, 6 \times 10^{-4}$ and $\zeta_G = 0, 0.01, 0.03, 0.1, 0.3, 0.5$. As observed for ECC-Q,

the use of ECC-G 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 $\varepsilon_{\text{tol}} = 3 \times 10^{-4}$ and $\zeta_G = 0$, the incurred error $\varepsilon_{0.9}$ is about 8×10^{-4}. Increasing ζ_G from 0 to 0.1 results in about halving $\varepsilon_{0.9}$ with a negligible CPU penalty. As ζ_G 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_G is between 0.1 and 0.3. With this value, the CPU time required to achieve a given value of $\varepsilon_{0.9}$ is about halved (compared to using $\zeta_G = 0$).

In summary, the ECC augmentation to the ISAT algorithm produces a significant decrease both in the ratio of $\varepsilon_{0.9}/\varepsilon_{\text{tol}}$ and in the number of table entries required for a given incurred error. For the first variant ECC-Q, where the frequency of ECC is controlled by the query time, the optimal value of ζ_Q is about 0.1 and it incurs a modest additional computational cost. For the second variant ECC-G, where the frequency of ECC is controlled by the number of grow events, the optimal value of ζ_G is between 0.1 and 0.3, and it incurs a negligible computational cost.

Nearly all of the tests reported in this paper were performed using ECC-Q, before the advantages of ECC-G were appreciated. Specifically, with the exception of the results reported in this subsection and the next, all of the tests reported in this paper used ECC-Q with $\zeta_Q = 0.1$. However, the preferred variant is ECC-G, with ζ_G=0.1.

6.8. Comparison of implementations

Prior to the present work, several significant improvements were made to the original ISAT algorithm (but these have not been documented in the literature). To distinguish different implementations, the algorithm described in this paper is designated ISAT5, and its immediate predecessor is ISAT4. In this sub-section, we make comparisons in the performance of these two implementations.

In contrast to ISAT5, ISAT4 does not use affine spaces, EOIs, MFU, MRU or EBTs. The retrieve search uses the binary tree in two ways. First (as in the original algorithm and in ISAT5), the binary tree is traversed to identify the primary leaf: this is called a "primary retrieve" attempt. If this fails (i.e., the query point is not covered by the EOA of the primary leaf) then a "secondary retrieve" is performed, which consists of successively testing neighboring leaves. As with the EBT in ISAT5, the amount of secondary retrieving is limited based on CPU time.

For both the PaSR test cases, we examine the relative performance of ISAT4 and three variants of ISAT5. There are: ISAT5-N (ISAT5 without ECC); ISAT5-Q (ISAT5 with the ECC frequency based on queries, $\zeta_Q = 0.1$); ISAT5-G (ISAT5 with the ECC frequency based on grow events, $\zeta_G = 0.1$). We first investigate the local error incurred by different implementations of ISAT through an examination of the 90% 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 an 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 evaluation 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.

Fig. 18 shows the 90% error $\varepsilon_{0.9}$ against the user-specified error tolerance ε_{tol} for the four different implementations of ISAT considered. 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 $\varepsilon_{0.9}/\varepsilon_{\text{tol}}$ 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. The two ECC methods considered achieve comparable performance. In other words, the new algorithms in ISAT5 (especially ECC and EOI) 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 test cases.

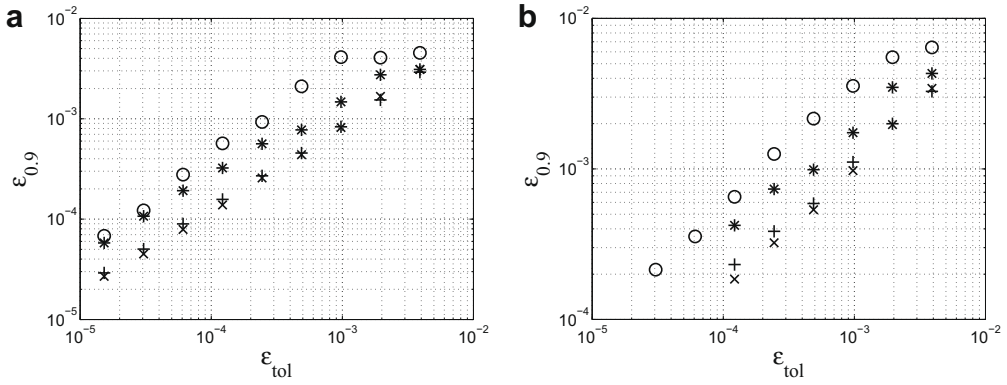


Fig. 18. The 90% error against the user-specified error tolerance from different implementations of ISAT for (a) the skeletal mechanism and (b) the GRI3.0 mechanism. Symbol ∘, ISAT4; ∗, ISAT5-N; +, ISAT5-Q ($\zeta_Q = 0.1$); ×, ISAT5-G ($\zeta_G = 0.1$). Each calculation results in 1.2×10^8 queries.

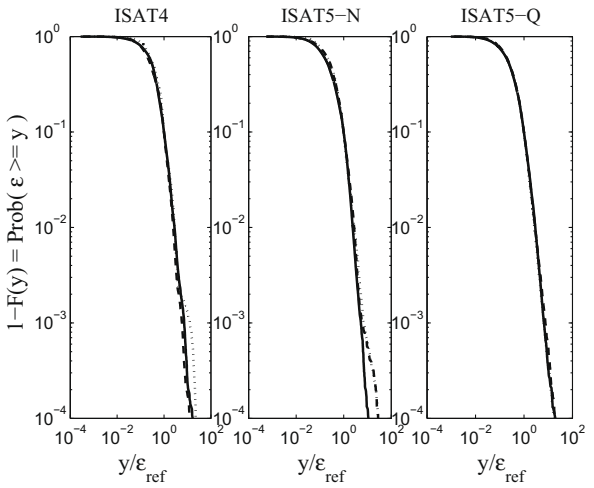


Fig. 19. CDF of local error against $y/\varepsilon_{\text{ref}}$ with $\varepsilon_{\text{ref}} = \varepsilon_{0.9}$ from different implementations of ISAT (ISAT4, ISAT5-N, ISAT5-Q) for the GRI3.0 mechanism. Dotted line: $\varepsilon_{\text{tol}} = 2^{-8}$; dashed line: $\varepsilon_{\text{tol}} = 2^{-9}$; solid line: $\varepsilon_{\text{tol}} = 2^{-10}$. Each calculation results in 1.2×10^8 queries. The ECC shown is ECC-Q, which is based on the query time.

Fig. 19 shows the CDF of incurred local error for different implementations of ISAT with three different values of error tolerance ε_{tol}. The quantity $1 - F(y) = Prob\{\varepsilon \geqslant y\}$ 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 cut-off 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

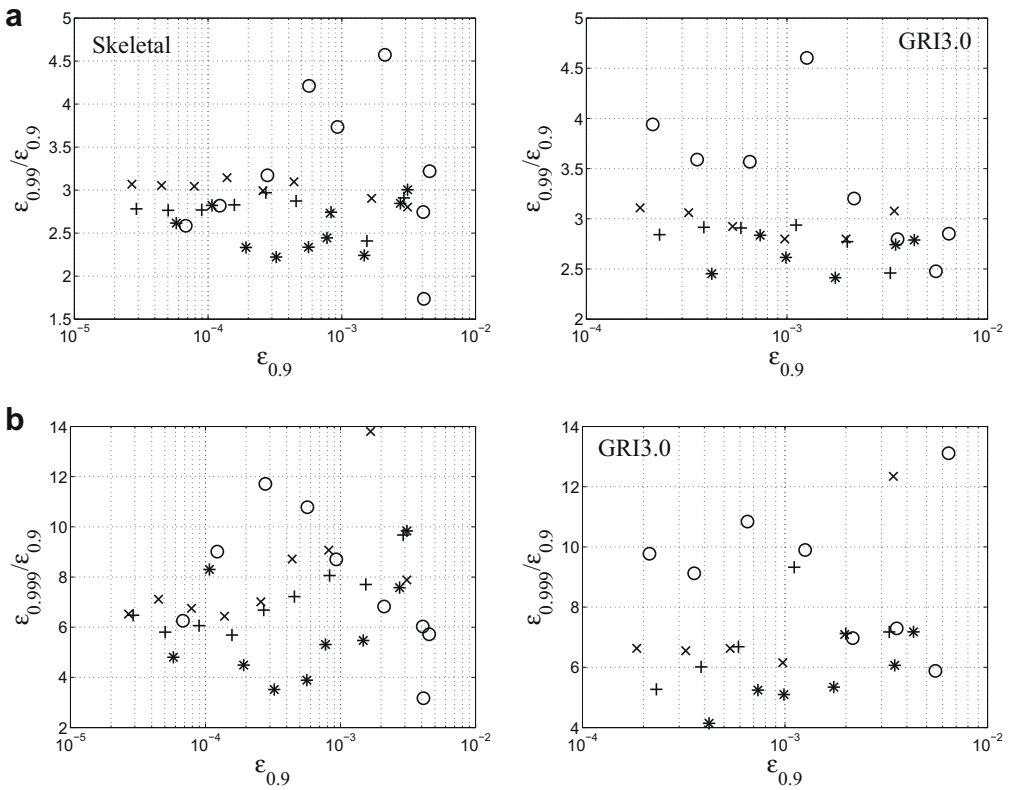


Fig. 20. For different implementations of ISAT, (a) the ratio of $\varepsilon_{0.99}/\varepsilon_{0.9}$ and (b) the ratio of $\varepsilon_{0.999}/\varepsilon_{0.9}$ against the 90% error. Left column: skeletal mechanism; right column: GRI3.0 mechanism. Symbol ∘, ISAT4; ∗, ISAT5-N; +, ISAT5-Q ($\zeta_Q = 0.1$); ×, ISAT5-G ($\zeta_G = 0.1$). Each calculation results in 1.2×10^8 queries.

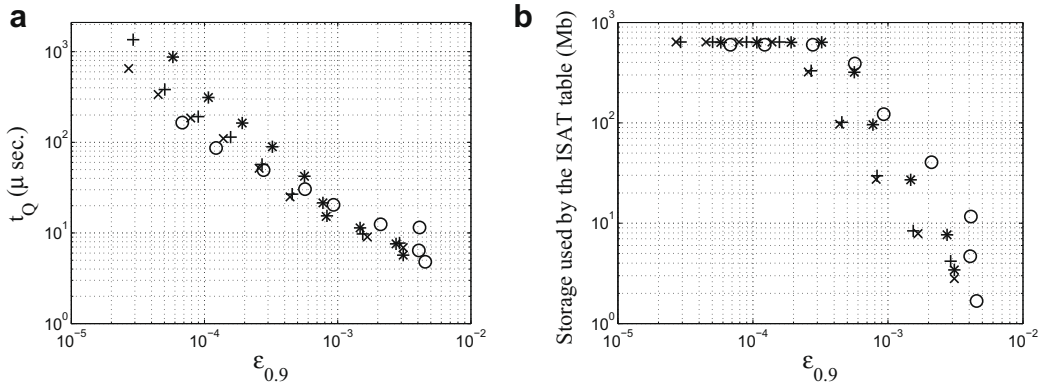


Fig. 21. For the PaSR test with the 16-species skeletal mechanism, comparison of different ISAT implementations (a) CPU time μs per query; (b) storage required (megabytes) against the 90% error. Symbol ∘, ISAT4; ∗, ISAT5-N; +, ISAT5-Q ($\zeta_Q = 0.1$); ×, ISAT5-G ($\zeta_G = 0.1$). Each calculation results in 1.2×10^8 queries. The allowed storage of 600 megabytes was reached for some cases.

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 $y/\varepsilon_{\text{ref}}$ even though ISAT5 with ECC produces less variation in the tail among different error tolerances. (The results for ISAT5-G [not shown] are similar to those for ISAT-Q.)

We further examine the tail of the CDF of the incurred error by studying the ratios $\varepsilon_{0.99}/\varepsilon_{0.9}$ and $\varepsilon_{0.999}/\varepsilon_{0.9}$, where $\varepsilon_{0.99}$ and $\varepsilon_{0.999}$ are the 99% and the 99.9% errors. Fig. 20 shows these ratios against the 90% error from different implementations of ISAT for both the skeletal and the GRI3.0 test cases. As may be seen from the figure, for ISAT4, there is a large variation in these ratios. The mean over the range of the 90% error considered is about 3.2 and 8 for $\varepsilon_{0.99}/\varepsilon_{0.9}$ and $\varepsilon_{0.999}/\varepsilon_{0.9}$, respectively. Thus, roughly speaking, there is a 0.1% probability of the incurred error exceeding 8 times the reference error, $\varepsilon_{0.9}$ The new implementation ISAT5, particularly ISAT5 with ECC, significantly reduces the fluctuations in these ratios. There is a slight improvement in the means of these ratios over the range of the 90% 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 $\varepsilon_{0.999}/\varepsilon_{0.9}$ inevitably contains some level of statistical error.)

We turn now to examine the performance in terms of CPU time per query and the size of the table generated. It is again stressed that the performance of different algorithms and implementations must be made at a fixed incurred error $\varepsilon_{0.9}$, not at a fixed error tolerance ε_{tol}.

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

ISAT5-G is computationally slightly more efficient than ISAT4, especially for the more demanding test case. 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-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

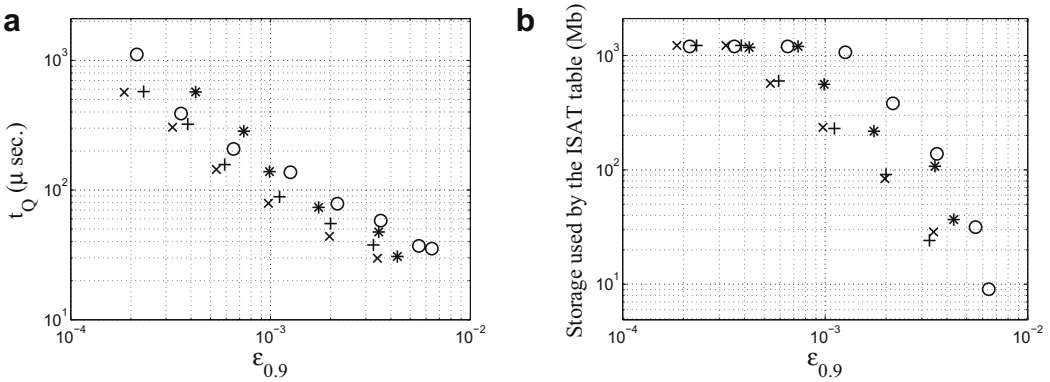


Fig. 22. For the PaSR test with the 53-species GRI3.0 mechanism, figure showing of different ISAT implementations (a) CPU time μs per query against 90% error and (b) storage required (megabytes) against 90% error. Symbol ∘, ISAT4; ∗, ISAT5 without ECC; +, ISAT5 with ECC-Q (ζ=0.1); ×, ISAT5 with ECC-G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

ratio 2:1 for ISAT4 and ISAT5-G. (The incurred error $\varepsilon = 1 \times 10^{-3}$ is significant in the sense that this small incurred error is found to be sufficient to guarantee the numerically accurate prediction of all species in computations of the turbulent methane/air flames considered in [3].)

Another important observation is that ISAT5-G reaches the specified allowed storage at a much smaller incurred error than ISAT4. For example, for the 53-species GRI3.0 mechanism, ISAT4, ISAT5-N, and ISAT5-G reach the allowed storage at about $\varepsilon_{0.9} = 1 \times 10^{-3}, 8 \times 10^{-4}$, and 4×10^{-4}, respectively. For a given incurred error, ISAT5-G 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-N, and ISAT5-G. This has significant practical importance because ISAT5-G can significantly alleviate the storage constraint on ISAT when applied to challenging reactive flows where large storage is required.

7. Conclusions

In situ adaptive tabulation (ISAT) has proved to be an effective method to reduce the computer time required to evaluate (approximately) high-dimensional functions, $\mathbf{f}(\mathbf{x})$. In this paper, several new algorithms for ISAT are described and demonstrated. These include: new search strategies (MRU, MFU, EBT); the use of low-dimensional affine spaces to reduce the cost of searching; ellipsoids of inaccuracy (EOIs) to identify candidates for growing; and error checking and correction (ECC).

The performance of the improved ISAT algorithm has been examined and compared to that of a previous version for two test cases of a partially-stirred reactor (PaSR). The dimensions of the two test cases are $n_x = n_f = 17$ (for the skeletal mechanism) and $n_x = n_f = 54$ (for the GRI mechanism).

The principal conclusions from the tests are as follows.

(a) In comparing different algorithms it is essential to do so at a fixed value of the incurred error (not at a fixed error tolerance).
(b) For a given ISAT problem involving very many queries, the average query time t_Q depends strongly on the allowed number of table entries, A, if this is less than a critical value, A^* (defined by Eq. (29)).
(c) The use of a low-dimensional ($n_a < 5$) affine space in searching is beneficial in reducing the time and storage by, typically, factors of 3 and 2, respectively.
(d) All search strategies (BT, MRU, MFU, EBT) are advantageous, and it is optimal to use them in order of increasing effective cost (i.e., time per successful retrieve).
(e) The EBT search is particularly effective. It is generally near optimal to limit the amount of EBT searching to consume at most half of the time for a direct function evaluation.
(f) The use of error checking and correction (ECC) typically halves the computer time required (for given incurred error).
(g) For the more challenging GRI test case, the CPU time and storage required by the improved version of ISAT (ISAT5-G) is smaller than that required by the previous version (ISAT4) by factors of 2 and 5, respectively.

Acknowledgments

For technical discussions and suggestions, the authors are grateful to several colleagues at Cornell, especially to Biswanath Panda, Zhuyin Ren and Paul Chew. This research was supported in part 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] S.B. Pope, Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation, Combust. Theory Modell. 1 (1997) 41–63.
[2] S.B. Pope, PDF methods for turbulent reactive flows, Prog. Energy Combust. Sci. 11 (1985) 119–192.
[3] B.J.D. Liu, S.B. Pope, The performance of in situ adaptive tabulation in computations of turbulent flames, Combust. Theory Modell. 9 (2005) 549–568.
[4] M.A. Singer, S.B. Pope, H.N. Najm, Modeling unsteady reacting flow with operator-splitting and ISAT, Combust. Flame 147 (2006) 150–162.
[5] M.A. Singer, S.B. Pope, H.N. Najm, Operator-splitting with ISAT to model reacting flow with detailed chemistry, Combust. Theory Modell. 10 (2006) 199–217.
[6] R. Cao, H. Wang, S.B. Pope, The effect of mixing models in PDF calculations of piloted jet flames, Proc. Combust. Inst. 31 (2007) 1543–1550.
[7] Q. Tang, W. Zhao, M. Bockelie, R.O. Fox, Multi-environment probability density function method for modelling turbulent combustion using realistic chemical kinetics, Combust. Theory Modell. 11 (2007) 889–907.
[8] S. James, J. Zhu, M.S. Anand, Large eddy simulations of turbulent flames using the filtered density function model, Proc. Combust. Inst. 31 (2007) 1737–1745.
[9] B. Merci, B. Naud, D. Roekaerts, Impact of turbulent flow and mean mixture fraction results on mixing model behavior in transported scalar PDF simulations of turbulent non-premixed bluff body flames Flow, Turbulence Combust. 79 (2007) 41–53.
[10] 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, Combust. Theory Modell. 11 (2007) 351–376.
[11] S. Mazumder, Modeling full-scale monolithic catalytic converters: challenges and possible solutions, J. Heat Trans. 129 (2007) 526–535.
[12] N.H. Kolhapure, R.O. Fox, A. Dai, F.-O. Mahling, PDF simulations of ethylene decomposition in tubular LDPE reactors, AIChE J. 51 (2005) 585–606.
[13] J.J. Shah, R.O. Fox, Computational fluid dynamics simulation of chemical reactors: Application of in situ adaptive tabulation to methane thermochlorination chemistry, Ind. Eng. Chem. Res. 38 (1999) 4200–4212.

[14] A. Arsenlis, N.R. Barton, R. Beckera, R.E. Rudda, Generalized in situ adaptive tabulation for constitutive model evaluation in plasticity, Comp. Methods Appl. Mech. Eng. 196 (2006) 1–13.
[15] J.D. Hedengren, T.F. Edgar, Approximate nonlinear model predictive control with in situ adaptive tabulation, Comput. Chem. Eng. 32 (2008) 706–714.
[16] A. Varshney, A. Armaou, Multiscale optimization using hybrid PDE/kMC process systems with application to thin film growth, Chem. Eng. Sci. 60 (2005) 6780–6794.
[17] S. Mazumder, Adaptation of the in situ adaptive tabulation (ISAT) procedure for efficient computation of surface reactions, Comput. Chem. Eng. 30 (2005) 115–124.
[18] J.-Y. Chen, J.A. Blasco, N. Fueyo, C. Dopazo, An economical strategy for storage of chemical kinetics: fitting in situ adaptive tabulation with artificial neural networks, Proc. Combust. Inst. 28 (2000) 115–121.
[19] I. Velijkovic, P.E. Plassmann, D.C. Haworth, A scientific on-line database for efficient function approximation, in: 2003 International Conference on Computational Science, Saint Petersburg, Russian Federation and Melbourne, Australia, 2003.
[20] J.-Y. Chen, Analysis of in situ adaptive tabulation performance for combustion chemistry and improvement with a modified search algorithm, Combust. Sci. Technol. 176 (2004) 1153–1169.
[21] I. Velijkovic, P.E. Plassmann, Parallel heuristics for an on-line scientific database for efficient function approximation, Appl. Parallel Comput.: State Art Sci. Comput. 3732 (2006) 644–653.
[22] G. Dong, B.C. Fan, Y.L. Chen, Acceleration of chemistry computations in two-dimensional detonation induced by shock focusing using reduced ISAT, Combust. Theory Modell. 11 (2007) 823–837.
[23] B. Panda, M. Riedewald, S.B. Pope, J. Gehrke, L.P. Chew, Indexing for function approximation, in: Umeshwar Dayal, Kyu-Young Whang, David B. Lomet,