POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale e dell'Informazione

Dipartimento di Energia

Dipartimento di Chimica, Materiali e Ingegneria Chimica

"Giulio Natta"

IMPLEMENTATION OF *IN SITU* ADAPTIVE TABULATION FOR THE CFD SIMULATION OF COMPLEX GAS-SOLID HETEROGENEOUS REACTORS

Relatori: Prof. Matteo MAESTRI

Prof. Alberto CUOCI

Correlatore: Dr. Tiziano MAFFEI

Tesi di Laurea in Ingegneria Chimica di:

Mauro BRACCONI 798765

Anno Accademico 2013-2014

a mio nonno...

Acknowledgements

Thanks to my family for your love, your support and your encouragements.

A gratefulness to Professor Maestri and Professor Cuoci for your supervision and your advises

Thanks to Dr. Maffei for your help and your suggestions

A special thanks to my friends Massimo, Davide e Daniela for having endured and supported me during these years.
Thanks to Matteo.

Abstract

The present Thesis proposes an improvement of the `catalyticFOAM` framework to reduce the simulation computational times of these two solvers increasing the speed of the chemistry calculation.

The `catalyticFOAM` framework was developed in a couple of previous Thesis [1, 2]. In order to investigate catalytic systems on the basis of the so-called “first principles”. Two different versions of the solver were implemented. Both of them have been tested either with simple geometries and complex cases, characterized by relevant industrial dimensions.

The first solver, named `catalyticFOAM`, is a CFD code able to simulate catalytic reactive systems in arbitrarily complex geometries with detailed microkinetic description of the surface chemistry. The modeling of the solid phase is neglected. The second solver, named `catalyticFOAM-multiRegion`, introduces a detailed description of the solid catalyst through a multi-region approach. A parallel version of both solvers has been implemented to exploit the possibility to use multi-processor architecture.

The microkinetic description of the chemistry coupled with the CFD simulation results in large computational costs. In particular, the solution of the chemistry, represented by a very stiff and strongly non-linear ordinary equations system, was experienced as the bottle-neck of the whole simulation. The reduction of the computational simulation time is one of the most important issues to allow for the simulation of industrial cases, characterized by very large and complex meshes.

The main aim of this work is to improve of the speed performances of the two solvers through the implementation of the ISAT, acronym for *In Situ* Adaptive Tabulation, which is a numerical algorithm developed by Pope [3]. In this work, ISAT has been applied to perform the CFD dynamic simulation of heterogeneous catalytic reactors for the first time. ISAT provides a storage and retrieval method giving an accurate approximation to high-dimensional functions which are computationally expensive to evaluate. The capability of the algorithm has been widely proved in homogeneous chemistry, whereas

its performance in heterogeneous catalytic systems has not been deeply investigated in the available literature.

A C++ object-oriented library, named *ISATLib*, implementing the ISAT algorithm has been developed. The accuracy of the predictions has been assessed through several test cases characterized by increasing complexity of the microkinetic scheme. The test cases consist in a set of isolated batch reactors integrated using both ISAT and an ODEs solver from different initial conditions. The reliability has been verified, thus the library has been implemented in both solvers to perform the solution of the chemical step.

The reliability and the predictive capabilities of the modified solvers have been tested by comparing the numerical results of ISAT simulation to the results obtained by direct integration in different operating conditions, in different configurations and with different heterogeneous and homogeneous kinetic mechanisms. The results show an excellent agreement between the profiles in all the operative conditions investigated.

The performance of the catalytictFOAM-*ISATLib* has been measured both in terms of speeding-up the chemical step and the overall simulation. This analysis has shown a very satisfactory capability of the *ISATLib* to reduce the computational time required to solve the chemistry, resulting in a considerable overall simulation speed-up.

Finally, several numerical simulations of packed bed catalytic reactors for steam reforming have been carried out during an internship period at the CFD department of the BASF production plant of Ludwishafen (Germany). The packed beds have been built up using a computational tool based on the DEM (Discrete Element Methodology). The tests have been performed to assess the performance of the library in these large dimension complex geometry test cases, resulting in satisfactory overall simulation speed-ups.

In conclusion, the capability of the ISAT algorithm to successfully reduce the computational simulation time has been demonstrated in the present Thesis. The possibility to apply this algorithm also to dynamic simulation of heterogeneous chemistry and the capability of modeling larger industrial cases highlights the potential of the codes, representing an important breakthrough in respect to the literature.

Sommario

Il presente lavoro di Tesi ha come obiettivo il miglioramento del solver `catalyticFOAM` al fine di ridurre i tempi computazionali aumentando la velocità di risoluzione della chimica.

In precedenti lavori di Tesi [1, 2] sono stati sviluppati due diversi strumenti di modellazione fluidodinamica di sistemi catalitici attraverso un approccio "first principle". I due codici sono stati testati sia in sistemi reattivi con geometrie semplici che in complessi casi industriali. Il primo solver, chiamato `catalyticFOAM`, è un codice capace di accoppiare efficientemente fluidodinamica computazionale con microcinetica dettagliata. Le reazioni eterogenee interessano solo la superficie del catalizzatore, trascurando la modellazione della fase solida. Il secondo solver, chiamato `catalyticFOAM-multiRegion`, introduce la descrizione dettagliata anche della fase solida, rendendo possibile una più completa analisi di sistemi costituiti da più regioni. Inoltre, il codice è stato completamente parallelizzato permettendone l'efficiente applicazione su macchine multi-processore.

Normalmente, la descrizione microcinetica della chimica accoppiata alla fluidodinamica computazionale determina elevati tempi di calcolo. In particolare, la risoluzione della chimica, rappresentata da un sistema di equazioni differenziali ordinarie fortemente non lineari e *stiff*, si è rivelata il collo di bottiglia di tutta la simulazione. La riduzione del tempo computazionale è di fondamentale importanza per permettere l'applicazione dei solver a casi industriali di sempre maggiori dimensioni.

Lo scopo di questo lavoro consiste nel miglioramento delle prestazioni dei due solver mediante l'implementazione di ISAT, acronimo di *In Situ* Adaptive Tabulation, un algoritmo numerico sviluppato da Pope [3]. Per la prima volta, in questo lavoro di Tesi, ISAT è stato utilizzato per effettuare la simulazione dinamica CFD di reattori catalitici eterogenei.

ISAT permette l'efficiente approssimazione di funzioni multidimensionali, costose da valutare direttamente, mediante un metodo di "*storage and retrieval*". Il miglioramento

delle prestazioni dovute all'algoritmo sono state largamente provate nel campo della combustione omogena, mentre le sue prestazioni nella soluzione di sistemi catalitici eterogenei non è stata investigata a fondo nella letteratura disponibile.

Si è, quindi, sviluppata una libreria numerica, chiamata *ISATLib*, in modo da implementare l'algoritmo. L'accuratezza dei risultati è stata verificata utilizzando casi, costituiti da un insieme di batch isolati aventi differenti condizioni iniziali. Al termine di questa procedura, la libreria è stata implementata in ambo i codici esistenti. La capacità dei codici `catalyticFOAM-`*ISATLib* di descrivere efficacemente i fenomeni reattivi è stata verificata attraverso casi, in differenti condizioni operative, con diverse geometrie del reattore e con differenti schemi microcinetici. In tutte le simulazioni effettuate, l'accordo tra i profili ottenuti con *ISATLib* e la integrazione diretta, sono risultati soddisfacenti.

Le prestazioni della libreria sono state valutate sia in termini di riduzione del tempo necessario per risolvere la chimica, che in termini di velocizzazione della simulazione nel suo complesso. L'analisi ha dimostrato una notevole capacità di *ISATLib* di diminuire il tempo computazionale richiesto per risolvere la chimica, con conseguente tempo inferiore per effettuare l'intera simulazione.

Infine, sono state effettuate numerose simulazioni di reattori a letto impaccato per lo *steam reforming* del metano, che sono state effettuate, durante un periodo di internship, presso il dipartimento di fluidodinamica computazionale della società BASF a Ludwigshafen (Germania). La struttura dei letti è stata generata attraverso l'ausilio di software basati sulla metodologia DEM (Discrete Element Method).

Sono stati, quindi, effettuati diversi test per verificare le prestazioni della libreria nella simulazione di queste complesse geometrie caratterizzate da gradi dimensioni. I risultati mostrano una soddisfacente riduzione dei tempi di calcolo.

La capacità della libreria di ridurre il tempo di calcolo è stato dimostrata, come anche l'applicazione dell'algoritmo a sistemi reattivi caratterizzati da chimica eterogenea. La modellazione di sistemi di rilevanza industriale ha mostrato le potenzialità del codice e l'innovazione che introduce rispetto alla letteratura esistente.

Table of contents

1 Introduction

1.1 Motivation

The numerical modeling of a catalytic system with complex kinetics and geometries encounters main difficulties due to the wide range of time and length scales involved, as shown in Figure 1.1. Therefore, the dominant reaction pathway is the result of the interplay among phenomena that occur at micro, meso and macro-scale.

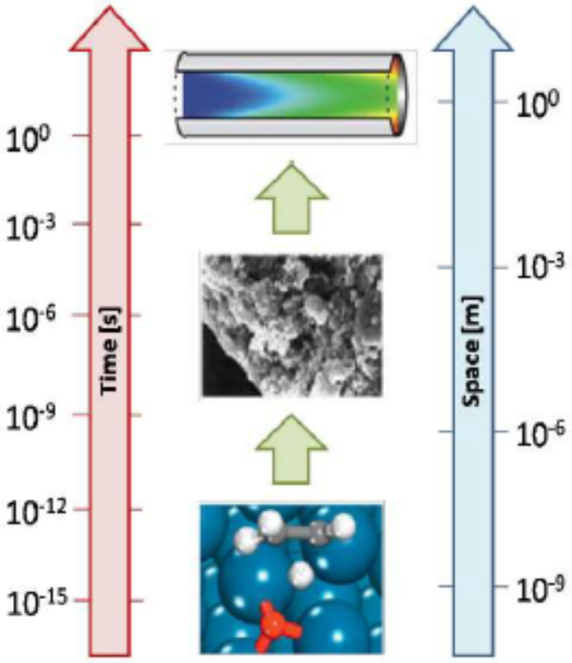


Figure 1.1 – Time and length scales involved in heterogeneous catalytic process

The microscopic scale is associated with making and breaking of chemical bonds between atoms and molecules. At the mesoscale, the interplay between all the elementary steps involved in the catalytic process determines the main reaction pathway. At the macroscopic scale transport of mass, energy and momentum determines local composition, temperature and pressure.

The dominant reaction mechanism is a multi-scale property of the system because of the interactions between these phenomena.

The description of different phenomena is performed by a "first principles" approach, i.e. at each scale the fundamental governing equations are considered. The behavior of the system at the molecular scale is described by detailed kinetic models, whose parameters are computed through first-principles electronic-structure calculations. At the mesoscale, statistical methods give a rigorous representation of the mechanism taking place at the catalytic surface; usually, the common approach is the mean field approximation. This approach assumes a perfect and rapid mixing of reactants, products and intermediates on the catalyst surface. The macro-scale is described through the continuum approximation, e.g. the resolution of Navier-Stokes equations with Computational Fluid Dynamics methodologies.

The approach requires to consider the fundamental aspects across the involved scales in order to link them in one multi-scale simulation.

The resulting numerical problem places high computational demands:

- The dimension of the problem is proportional to the number of species involved in the reacting process. Thus, the larger and more detailed is the kinetics scheme, the higher is the computational time required.
- The geometrical domain has to be discretized to solve the governing equations of the system. The number of cells in which the volume is divided depends both on the dimension and the geometrical complexity of the problem and it is proportional to the accuracy and the stability of the method.
- The presence of a wide range of different characteristic times among the scales leads to a very stiff problem. Furthermore, the presence of the reacting term implies a strong non-linearity of the governing equations.
- The accurate description of the problem should include the description of the catalytic phase and the modelling of intra-solid.

The growing importance of the Computational Fluid Dynamics is due to its capability to obtain data and information on a certain system. The CFD codes can provide enormous

quantities of data through a cheaper method because the cost of the simulation is quite independent from the quantity of data collected.

Nowadays rigorous implementation of optimal policies for maximizing conversion and integration of control strategies has become important. Industries as well as academia have focused many efforts to improve the performance of CFD tools. The availability of super-fast computing facilities has prompted the coupling between such detailed fluid dynamics and chemistry description. The frontier of the feasible simulations is set by the available computational resources. The processor speed is continuously increasing as the performance of the parallel super-computer, as shown in Figure 1.2.

Nevertheless, one of the main issues in the numerical simulation of complex geometries is the computational effort required which might be larger than the available resources. Thus, the importance of the reduction of the computational cost of these simulations is a critical issue, in order to enlarge the dimension of the feasible problems, getting closer to a real, multidimensional, complex industrial case.

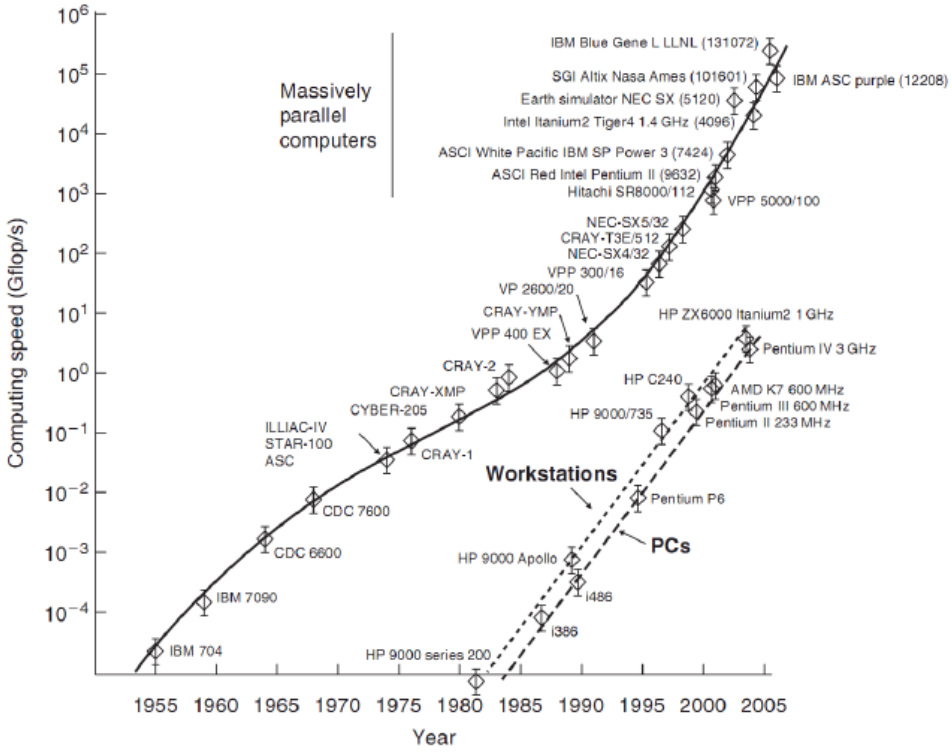


Figure 1.2 – Evolution of the computational power

In previous works [1, 2] two CFD solvers, named `catalyticFOAM` and `catalyticFOAM-multiRegion`, have been developed within the OpenFOAM® (Open Field Operation and Manipulation) framework, which is a free, open source CFD software package applied in solving complex fluid flows involving chemical reactions, turbulence and heat transfer framework [4].

The first one is able to investigate catalytic systems with a "first principle" approach, i.e. employing the governing equations to describe each scale, coupling fluid dynamics methodologies with a detailed microkinetic description of the surface reactivity. Nevertheless, the inner part of the catalytic phase is not modeled in detail: the reactions occur on a superficial catalyst layer and the intra-porous diffusion is not taken into account. The second solver was developed to introduce a detailed description of the catalyst by means of a multi-region approach. The dynamic solution of the catalyst domain considers the transport phenomena within the solid phase and the heterogeneous reactions, which occur in the whole solid volume.

The simulation of complex geometries has been carried out in previous work [1] for both solvers and one of the main issues was the required computational cost. Coupling the CFD simulation with the detailed microkinetic description of the chemistry determines large computational costs which limit the allowable dimension of the problem in order to be described in reasonable time.

1.2 Aim of the work

The main aim of this work is the reduction of the computational times of these two solvers through the implementation of ISAT algorithm. In particular, the solution of the chemistry, represented by a very stiff and strongly non-linear ordinary equations system, was experienced as the bottleneck of the whole simulation. This constraint on the maximum dimension of the problem limits the capability of the solver, avoiding its use in very large, industrial scale problems.

ISAT, acronym for *In Situ A*daptive Tabulation, is a numerical algorithm developed by Pope [3], which provides a storage and retrieval method giving efficiently an accurate approximation to high-dimensional functions which are computationally expensive to evaluate. It has been successfully applied in speeding-up the chemistry in complex

turbulent reactive flow calculations [5]. The capability of the algorithm has been widely tested in homogeneous chemistry, whereas its implementation in heterogeneous catalytic systems has not been deeply exploited in the available reference yet and the available literature regards ISAT implementation in steady state solver [6]. In particular, the application of the algorithm to dynamic CFD simulations of heterogeneous catalytic reactors has been carried out for the first time in this work.

A code implementation of the ISAT numerical algorithm, named *ISATLib*, has been developed and widely tested before being implemented in the `catalyticFOAM` framework. Then, reliability and performances of the modified solvers are tested in several simple-geometry test cases. Finally, the simulations of an industrial reactor for the steam reforming of methane have been carried out during an internship period in BASF.

A short description about the organization of the present Thesis is reported in the following paragraphs.

In **Chapter 1**, the description of the ISAT algorithm is provided. The method is: *in situ*, unstructured, adaptive tabulation of the composition space, with control of retrieval errors. This technique is able to tabulate a multi-dimensional function and to use the stored data to approximate the function values in other points of the domain, by means of linear interpolation. An essential aspect of ISAT is that the table is built up *in situ* as the simulation is carried out: at the beginning of the simulation the table is empty and the records are added only if necessary.

In **Chapter 2**, the *ISATLib* library is described in detailed explaining the code implementation. Its structure suits the requirement of being able to front every problem concerning the several evaluations of a multi-dimensional function. For this purpose the direct function evaluation has to be provided by the main numerical code letting the user to choose the best way to obtain the function value without modifying the library. The library uses a Binary Structure, as storage and retrieval structure, which well-manages the data stored providing a fast search strategy and the capability of modifying its dimension during the simulation. Moreover, other three different search techniques are implemented to increase the number of successful retrieves of the data stored. The Binary Tree structure is managed to be kept efficient through the use of a proper balancing algorithm. In this work, the library has been used to solve ODEs systems, but

can be applied to wide fields of application without any kind of modification. The reliability of the library, i.e. the agreement with the results obtained by direct function evaluations, is investigated in test cases of increasing complexity. Last, an analysis of the main parameter effects on the library performance is done.

In **Chapter 3**, the implementation of the *ISATLib* inside the `catalyticFOAM` framework is described. The solver implements an operator splitting algorithm. The library has been implemented to reduce the time required to perform the chemical step. Thanks to the library implementation, the direct solution of the ODEs system is carried out only when the ISAT table cannot provide an accurate solution based on the stored data. The performances and the reliability of the modified solvers have been investigated in simple-geometry test cases with increasing dimension of the kinetics schemes in adiabatic conditions.

In **Chapter 4**, different packed beds have been analyzed to allow the simulation of an industrial reactor of steam reforming. These structures are generated through the DEM (Discrete Element Method) approach. The DEM is a modelling technique capable of describing the mechanical behavior of assembling structures through an explicit numerical method in which the interactions among the particles are monitored contact by contact and the motion of particles is modeled particles by particles. The final bed configuration is made by the unit elements assembled in a random way. The implementation of the library in the parallel version of the solvers is also described. The *pure local processing* (PLP) [7] is implemented which is characterized by no message passing and no load redistribution among the *ISATLib* instances of each processor. Then, an industrial reactor for steam reforming has been simulated, in parallel, in order to analyze the capability of the modified solver in speeding-up the simulation. Then, the effect of the pellet shape and the different porosities of the catalyst on conversion are also investigated.

1.3 State of the art

ISAT has been applied to several problems since its first development. The primarily application was in the calculation of a non-premixed methane-air combustion in a statistically homogeneous turbulent reactor, using a detailed kinetic mechanism. The

results have shown excellent control of the approximation error with respect to a specified error tolerance. A speed-up factor of about 1000 has been obtained compared to the time required by a direct numerical integration of the equations, making feasible the use of detailed kinetic mechanism in PDF methods of turbulent combustion, as demonstrated by Pope in [3].

Moreover, an improved version of the algorithm, named ISAT5, has been developed [8] and implemented in a Fortran library [9], in order to increase the performances of the algorithm. In particular, several table search procedures have been introduced in the algorithm to enhance the probability of a successful record retrieve.

ISAT has been applied to the efficient solution of chemistry calculations in homogeneous turbulent reactive flows in conjunction with transported probability density function (PDF) method. In particular, Pope [10] and Tang et al. [11] have used ISAT in the computation of turbulent non-premixed piloted jet flames. The PDF/ISAT method has been demonstrated to be capable of representing the intense interaction between turbulence and finite rate chemistry in turbulent non-premixed flames. ISAT has also been used with artificial neural networks (ANN) and Large Eddy Simulation (LES) by Kapoor et al. [12]. Moreover, ISAT has been applied to the simulation of sprays in conjunction with PDF/LES methodology resulting in an improvement of the performances and allowing the implementation of a detailed kinetic mechanism [13, 14].

The algorithm has also been successfully combined with dimension reduction strategies in order to improve the speeding-up performances. As shown by Tang [15], the rate-controlled constrained equilibrium approach has been coupled with *in situ* tabulation to simulate a non-premixed methane-air combustion in a statistically homogeneous turbulent reactor, using a detailed kinetic scheme. The ISAT results show good agreement with the exact solution and a speed-up factor of 500 is obtained compared to direct integration approach, which demonstrates the high efficiency of the method.

The ISAT algorithm has also started to be used in Eulerian CFD calculations of reacting flows involving homogeneous reactions. Wang et al. [16] showed the capability of the algorithm coupled with PDF for carrying out time-evolving simulations of nano-particles formation by reactive precipitation. The results show that CFD simulation of reactive

precipitation requires a much smaller computational effort when the ISAT algorithm is implemented than when direct integration is used.

Singer et al. [17] have been also used ISAT with numerical splitting techniques based on the Strang algorithm: the portions of the governing equations containing stiff chemical reaction terms are separated from those parts containing less-stiff transport terms. To improve the computational efficiency, the pure reaction sub-steps use *in situ* adaptive tabulation while the diffusion sub-steps implement an implicit Crank-Nicolson finite-difference method. This scheme has been applied to a one-dimensional reaction diffusion model with detailed chemical kinetics obtaining a speed-up approximately 4.5-5 times faster than direct integration. Moreover, ISAT has been used to simulate a two-dimensional unsteady laminar reacting flow through a parallelized sub-splitting procedure where the *in situ* adaptive tabulation has been applied to the solution of the pure reaction steps. In this system, Singer et. al observed an overall speed-up factor due to ISAT of approximately 2.5-3 [18].

Hedengren et. al [19] have been successfully applied the *in situ* adaptive tabulation algorithm to non-linear model predictive control (NMPC) problems. Model predictive control (MPC) is now suggested as a candidate to replace PID control due to recent developments in computational reduction of the MPC algorithm. Usually, all the possible solutions are computed off-line and stored, the on-line portion is reduced to some conditional checking and simple matrix multiplications. The ISAT implementation allows to perform this operation on-line by adaptively building of the store reducing its dimension by using local linear solutions to approximate the region where the local solution is within a specified tolerance. Furthermore, the approach is applicable to MPC with linear and non-linear models.

A technique for the adaptive order reduction of large scale non-linear differential algebraic equations (DAEs), implementing ISAT, has been developed by Hedengren et. al [20]. In this work, it is highlighted that ISAT is superior to neural networks because of better interpolation and extrapolation abilities, resulting in an efficient approximation of the functions.

Mazumder et al. [6] have been applied an adapted version of the ISAT algorithm to heterogeneous reacting flow computations with complex surface description. it has been used into a steady-state solver to perform the solution of the DAEs system

associated to the surface reactivity. The algorithm implemented in the steady-state solver presents some modifications in respect to the usual version. First of all, the domain is divided in several subdomains in which a proper ISAT table is initialized. Secondly, the approximation error might alter the convergence behavior resulting in large-scale fluctuations of residuals and sometimes in stalling of convergence or even in divergence. Moreover, the retrieval rate was found to be high at the start of the steady state simulation, but deteriorated as the simulation proceeds toward convergence, resulting in extremely large tables. The implemented strategies periodically clear the ISAT table after a specified number of iterations and then rebuilt from scratch, primarily because the initial guess may be quite different from the final steady state solution. Deletion and re-creation of the tables has been found to be critical in maintaining retrieval rates above a certain minimum value. Furthermore, during the final stages of convergence, the errors from ISAT approximations are large enough to negatively affect convergence and thus, as the solution approaches convergence, the tolerance used for ISAT retrievals should be decreased. The studies figured out an overall speed-up factor in a range of 1.4-2.5 for the whole reacting flow calculation.

2 ISAT Algorithm

In this chapter the In Situ Adaptive Tabulation algorithm, as proposed by Pope [3, 8] is deeply described. The structure and the main feature of both the algorithm and the storage table are presented.

2.1 General overview

ISAT, acronym for *In Situ* Adaptive Tabulation, is a numerical algorithm base on the storage and retrieval method which efficiently provides accurate approximation to high-dimensional function that are computationally expensive to evaluate [8]. This method is particularly useful to solve system of differential equations [5], model predictive control [19] and to reduce large scale DAE models [20].

The purpose of ISAT is to tabulate a function $\boldsymbol{f}(\boldsymbol{x})$, where $\boldsymbol{f}$ and $\boldsymbol{x}$ are of dimension n. The function is defined in a n-dimensional space called *composition space*. The composition space is the domain of definition of the problem ISAT is dealing with; it is the set of all the possible values that can be assumed by the function.

For example, at any point and time, the thermochemical state [3] of a reactive gas flow can be characterized by the mass fractions of the species, temperature and pressure. All these quantities can be collected in a vector representing the state of the system. The mass fractions are not linearly independent because they sum to unity. The linear independent subset of the vector state defines the composition space. So, a point of the composition space represents a thermochemical state. In this case, $\boldsymbol{x}$ consists of the thermochemical state of a particle at the beginning of the time step, and $\boldsymbol{f}$ represents the composition at the end of the step. Evaluating $\boldsymbol{f}(\boldsymbol{x})$ requires to solve a stiff set of n ordinary differential equations on the chemical time step.

Given a query $\boldsymbol{x}^q$, which is a certain point of the composition space, the straightforward approach to determine the function value $\boldsymbol{f}(\boldsymbol{x}^q)$ is a direct evaluation, which means, for example, to integrate the set of ordinary differential equations numerically. For large dimension and stiff problems, the direct integration is computationally expensive.

An alternative approach used to evaluate the value of the function involves tabulation. In a pre-processing stage, direct integration is used to build up a large table consisting of many function values determined from different initial conditions. A large region of the composition space is mapped and, during the simulation, the value of the function is determined by a table look up using multi-linear interpolation.

Tabulation is one example of the concept of storage and retrieval. The storage and retrieval techniques can be judge using these criteria [3]:

- CPU time required to create the table;
- Memory required for the store
- Inaccuracies in the retrieved function value
- CPU time required to retrieve the function value

If the problem is low-dimensional, the tabulation technique satisfies the criteria: interpolation error are acceptably small using modest table dimension, the retrieval time could be faster than the time to perform direct evaluation, finally the time to create the store is negligible when the number of retrievals is high.

The application of this tabulation method to high dimensional problems is not feasible due to the overstated dimension of the table required to achieve good interpolation results.

The problem just encountered can be overcome using ISAT technique: it is defined, as suggested by Pope [3], *in-situ* adaptive tabulation of the accessed region, with control of retrieval error.

2.1.1 Accessed region

First of all it is important to focus the attention on what the accessed region is. The composition space consists of the whole realizable region which is bounded only by the laws that are governing the problem. Let be n the dimension of the problem. The realizable region is a convex polytope in n-space.

The accessed region is the subset of the realizable region consisting of all the points that occur in the solution of the given problem.

In order to use a tabulation technique, it is sufficient to tabulate only the accessed region, rather the whole composition space. The accessed region could have an irregular shape which is not known *a priori*. For this reason the table is not built up in a pre-processing phase but during the calculation. Each record stored corresponds to a point in the composition space that occurs in the calculation. This feature is referred as *in-situ* tabulation.

2.1.2 Linear approximation

The table contains the information about the function value in particular points, denoted generically $\boldsymbol{x}^0$. Figure 2.1 shows a sketch of the reaction trajectory from $\boldsymbol{x}^0$ to a query point $\boldsymbol{x}^q$.

The direct integration, performed starting from the tabulation point $\boldsymbol{x}^0$, leads to determine the value of the function $\boldsymbol{f}(\boldsymbol{x}^0)$.

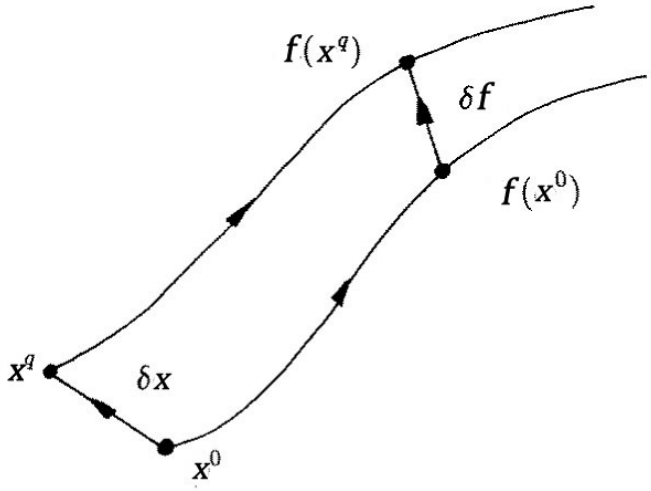


Figure 2.1 – Sketch showing the definitions of the displacement δx and δf [3]

Given a query point $\boldsymbol{x}^q$, the displacement in the initial condition $\delta \boldsymbol{x}$ and in the function value $\delta \boldsymbol{f}$ are defined as follows:

$$\boldsymbol{x}^q = \boldsymbol{x}^0 + \delta \boldsymbol{x} \qquad \boldsymbol{f}(\boldsymbol{x}^q) = \boldsymbol{f}(\boldsymbol{x}^0) + \delta \boldsymbol{f} \tag{2.1}$$

In the table also the sensitivity matrix, called mapping gradient, is stored which reveals the amount that $\boldsymbol{f}$ changes with a small perturbation $\delta\boldsymbol{x}$:

$$A_{ij} = \frac{\partial f_i(\boldsymbol{x})}{\partial x_j} \tag{2.2}$$

The method used to evaluate these sensitivity coefficients is described in Section 2.3.3.1. Using these stored data, a linear approximation of the function value in the query point can be obtained:

$$\boldsymbol{f^l}(x^q) = \boldsymbol{f}(\boldsymbol{x}^0) + \delta\boldsymbol{f^l} \approx \boldsymbol{f}(\boldsymbol{x}^q) \tag{2.3}$$

where:

$$\delta\boldsymbol{f}^l = \boldsymbol{A}\delta\boldsymbol{x} = \delta\boldsymbol{f} + O(\|\delta\boldsymbol{x}\|^2) \tag{2.4}$$

2.1.3 Local error

The error associated to the approximation of the function reported above is measured by the two-norm of the difference between the approximated value and the real value:

$$\varepsilon = \left\|\boldsymbol{f^l}(\boldsymbol{x}^q) - \boldsymbol{f}(\boldsymbol{x}^q)\right\| \tag{2.5}$$

The accuracy of the method is controlled using the linear approximation at a certain tabulation point $\boldsymbol{x}^0$ only if the local error is less than a specified maximum error value defined as error tolerance.

It is preferable to weight the different components of the error for an appropriate scaling error factor: it is introduced a scaling matrix $\boldsymbol{B}$ in order to define the local error as:

$$\varepsilon = \left\|\boldsymbol{B}\left(\boldsymbol{f^l}(\boldsymbol{x}^q) - \boldsymbol{f}(\boldsymbol{x}^q)\right)\right\| \tag{2.6}$$

2.1.4 Region of accuracy

The region of accuracy (ROA) is defined as the connected region containing the $\boldsymbol{x}^0$ consisting of all the points for which the local error is less than the specified tolerance

In order to set the region of accuracy, the following approximation is considered instead of the linear approximation:

$$\boldsymbol{f}(\boldsymbol{x}^{\boldsymbol{q}}) \approx \boldsymbol{f}^{\boldsymbol{c}}(\boldsymbol{x}^{\boldsymbol{q}}) \equiv \boldsymbol{f}(\boldsymbol{x}^{0}) \tag{2.7}$$

The function value is approximated around the tabulation point using the value in the tabulation point.

The local error in constant approximation is defined by:

$$\varepsilon_c = \left\| \boldsymbol{B}\left(\boldsymbol{f}^{\boldsymbol{l}}(\boldsymbol{x}^{\boldsymbol{q}}) - \boldsymbol{f}(\boldsymbol{x}^{\boldsymbol{q}})\right)\right\| = \|\boldsymbol{B}\delta\boldsymbol{f}\| \tag{2.8}$$

The difference between the real value of the function and the one in the constant approximation is, to leading order:

$$\delta\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}^{\boldsymbol{q}}) - \boldsymbol{f}^{\boldsymbol{c}}(\boldsymbol{x}^{\boldsymbol{q}}) = \boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{q}} - \boldsymbol{x}^{0}) = \boldsymbol{A}\,\delta\boldsymbol{x} \tag{2.9}$$

Thus:

$$\varepsilon_c = \|\boldsymbol{B}\,\boldsymbol{A}\,\delta\boldsymbol{x}\| \tag{2.10}$$

The region of accuracy in the constant approximation is given by:

$$R = \{\,\boldsymbol{x} \mid \varepsilon_c(\boldsymbol{x}) \leq \varepsilon_{tol}\} \tag{2.11}$$

In the constant approximation, on the boundary of the region of accuracy, the error is equal to the tolerance:

$$\delta\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{B}\boldsymbol{A}\,\delta\boldsymbol{x} = \varepsilon_{tol}^2 \tag{2.12}$$

The symmetric positive semi-definite matrix $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{B}\boldsymbol{A}\,/\varepsilon_{tol}^2$ can be expressed as:

$$\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{B}\boldsymbol{A}\,/\varepsilon_{tol}^2 = \boldsymbol{Q}^{\boldsymbol{T}}\boldsymbol{\Lambda}\,\boldsymbol{Q} \tag{2.13}$$

where $\boldsymbol{Q}$ is a unitary matrix and $\boldsymbol{\Lambda}$ is a diagonal matrix, with non-negative diagonal elements $\lambda_1, \lambda_2, \dots, \lambda_n$. Therefore:

$$\delta\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{B}\boldsymbol{A}\,\delta\boldsymbol{x} = \varepsilon_{tol}^2 \;\Rightarrow\; \delta\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{Q}^{\boldsymbol{T}}\boldsymbol{\Lambda}\,\boldsymbol{Q}\,\delta\boldsymbol{x} = 1 \tag{2.14}$$

Consequently, the region of accuracy, in constant approximation, is a hyper-ellipsoid, reported in Figure 2.2, where the principal semi-axes are $l_i = 1/\sqrt{\lambda_i}$.

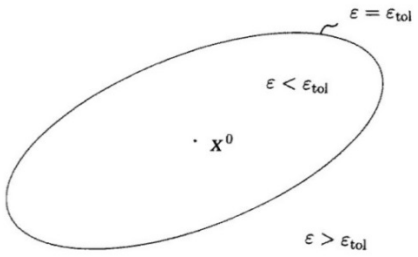


Figure 2.2 – Sketch showing the ellipsoid of accuracy around x^0 [3]

Using the singular value decomposition of $\boldsymbol{A}$, the left-hand of the equation (2.13), can be written:

$$\boldsymbol{A^T B^T B A} / \varepsilon_{tol}^2 = \boldsymbol{V \Sigma U^T U \Sigma V^T} / \varepsilon_{tol}^2 \tag{2.15}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrix and $\boldsymbol{\Sigma}$ is the diagonal matrix of singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

It is obtained that:

$$\boldsymbol{\Lambda} = \left(\frac{\boldsymbol{\Sigma}}{\varepsilon_{\mathrm{tol}}}\right)^2 \tag{2.16}$$

so that the half-length of the principal axes of the hyper-ellipsoid are $l_i = 1/\sigma_i$. The values of σ_i affected the shape of the region of accuracy.

The region of accuracy, in the constant approximation, is defined by:

$$E = \left\{ \boldsymbol{x} \mid \delta \boldsymbol{x^T V \Sigma^T \Sigma V^T} \delta \boldsymbol{x} / \varepsilon_{tol}^2 \leq 1 \right\} \tag{2.17}$$

The development of the region of accuracy uses the constant approximation whereas ISAT method employs the linear approximation. The corresponding analysis in the second case would be more complicated and would lead to a tensor equation for the region of accuracy. Therefore, the ellipsoid of accuracy is an approximation of the true region of accuracy, so it cannot be guaranteed that the local error is below the tolerance for all points within the EOA. On the other hand, the EOA is developed in order to limit the error in the constant approximation, whereas the more accurate linear approximation is used in retrieving, so that the retrieving error should be small and less than the tolerance providing an excellent control of local errors.

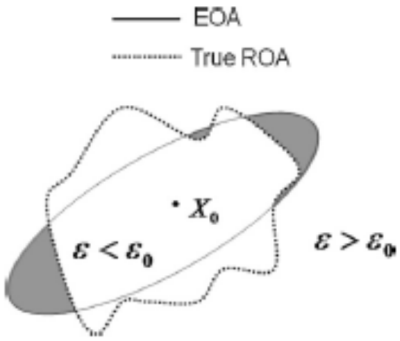


Figure 2.3 – Sketch of the real ROA and of the EOA [6]

2.2 Binary Tree

The data structure mainly used by ISAT is a Binary Tree. It is a method of placing and locating records, especially useful when all the data are known to be stored in random access memory (RAM).

A Binary Tree is a type of data structure where the nodes are arranged in order: for each node, all elements in its left sub-tree are less or equal to the node, and all the elements in its right sub-tree are greater than the node. The algorithm finds data by repeatedly dividing the number of accessible records in half until one remains comparing in succession the value stored in the table with the query.

In a Binary Tree, records are stored in location called leaves whereas branch points are the nodes. The order of a tree is the number of sub-trees per node: in a Binary Tree, there are always two children per node, so the order is two. The topmost node is called root.

Basically, Binary Trees are fast at insert and lookup; on average, a Binary Tree search algorithm can locate a record in a tree of N nodes in order of $\log_2 N$ time.

The ISAT implementation of the Binary Tree [3] is characterized by two different elements: the leaves and the nodes, as reported in Figure 2.4. Each leaf corresponds to a record stored in the tree, whereas each node contains the information to navigate within the Binary Tree

The features of a Binary Tree are effective for ISAT purpose, because the tree is enlarged only if necessary and it is easy and fast to traverse to reach one of its leaves.

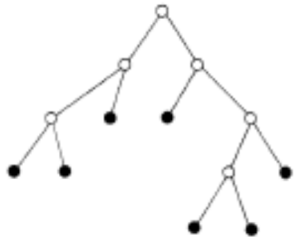


Figure 2.4 – Sketch showing the Binary Tree; at each node there is a cutting plan, at each leaf ● there is a record [3]

2.2.1 Cutting plane

The nodes contain the information needed to steer in the Binary Tree: at each node the cutting plane is stored, which consists of a n-vector $\boldsymbol{v}$ and a scalar a. All the points $\boldsymbol{x}$ with $\boldsymbol{v}^T\boldsymbol{x} > a$ are on the right of the cutting plane, the others on the left, as shown in Figure 2.5. The cutting plane plays the same role described above for the node of the integer tree: they allow computing a simple scalar product to determine whether a point is on the right or on the left of the cutting plane. Given a point, it is easy to traverse the tree to reach the record which is close to the point. For this purpose, it is necessary to compare in succession point value with nodes information. The power of this feature is that with a reduced number of tests the tree can be easily traversed, reaching the bottom edge where the records are stored. The procedure to initialize the cutting plane is described in a Section 2.3.3.3.

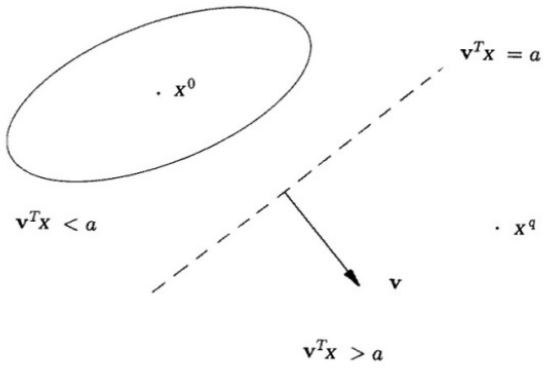


Figure 2.5 – Sketch of the cutting plane [3]

2.2.2 Leaf of the Binary Tree

The leaves are the node at the bottom edge of the tree, they have empty sub-trees but, at the same time, they store the information needed to ISAT:

- the tabulation point $\boldsymbol{x}^0$ point is an n-vector and it is the location of the leaf in the composition space. It consists in the initial states and inputs
- the function value $\boldsymbol{f}(\boldsymbol{x}^0)$ value is an n-vector and it is the value of the tabulation point after a direct integration on a given time step
- the mapping gradient matrix $\boldsymbol{A}(\boldsymbol{x}^0)$ is an $n \times n$ matrix which components are [8]

$$A_{ij}(\boldsymbol{x}) = \frac{\partial f_i(\boldsymbol{x})}{\partial x_j} \tag{2.18}$$

  This matrix is used to construct the linear approximation used in ISAT. The mapping gradients are related to the sensitivity coefficients of the system.
- The information concerning the region of accuracy is stored in the leaf.

Furthermore, the leaf stores also some counters like the number of times of retrieves and grows and also the last time that has been used.

2.2.3 Depth and height

The depth of a Binary Tree is defined as the number of nodes along the path from the first node (root) to the deepest leaf node. Note that the depth of an empty tree is zero. The height is the difference between the depth of left and right root sub-trees.

These two features are important to understand if a binary is balanced: a tree is balanced when the height is one or less. A balanced Binary Tree has a predictable root depth, which means that the number of nodes traversed to reach one leaf is known and let be N the depth of the Binary Tree, then the number of nodes traverse is the integer part of $\log_2 N$. For example a Binary Tree which consists of 100 nodes, if it is balanced, has a depth of 6.64. This means that to reach the termination leaf are necessary only six comparisons between the query and the value stored in the node making fast the search process within the tree.

The shape of the tree is an important feature which can condition the search time: an unbalanced tree requires more time to reach the terminal leaf than a balanced one. If a

Binary Tree is balanced the leaf retrieve process is the most effective because it requires the minimum time to be executed.
In Section 2.3.5, of this work the importance of balancing will be discussed together with the methods used to get it.

2.2.4 Memory allocation

Each node and each leaf of the Binary Tree requires a certain amount of the available memory. In particular the leaf of the tree contains vectors and matrix of double real-precision variables, so that the memory occupied by a single leaf is not negligible.
When the Binary Tree reaches a given dimension it can fulfil all the available memory, so it is necessary to limit di maximum number of the table entries and to specify the actions that have to be taken when it is full.

2.3 Algorithm

As suggested by Pope [3], the following procedure is followed to store and retrieve data from the Binary Tree:

- At the beginning of any computation the binary table is empty. On receipt of the first query, the first record is generated, and the Binary Tree is initialized as a single leaf. The exact value of the mapping is returned;
- For next queries, the algorithm is as follows:
    1. **Retrieve**: the Binary Tree is traversed in order to identify a leaf whose EOA covers the query point; if such an EOA is found the linear approximation based on that leaf is returned.
    2. **Growth**: if the retrieve procedure is unsuccessful, which means that the EOA doesn't cover the query point, a direct integration is performed. The error in the linear approximation is evaluated and if it is less than the error tolerance the region of accuracy of that leaf is grown to cover the query point.

3. **Addition**: if the growing attempt fails, a new record is generated based on the query composition and it is added to the tree table.

2.3.1 Retrieve

Given a query $\boldsymbol{x^q}$, the aim of the retrieve procedure is to find a EOA which covers the query point. There could be zero, one or more leaves with this feature. Among all those leaves, it would be identified the leaf, denoted by $\boldsymbol{x^0}$, that is closest to $\boldsymbol{x^q}$, which means that the error in the linear approximation is minimized. It is computationally expensive to find the closest leaf, thus the retrieve procedure tries to identify one of this leaves using different search strategies. Nevertheless, whatever leaf, which region of accuracy covers the query point, will return an acceptable linear approximation of the function.
If at least one EOA exists, that covers the query point, the search process is deemed complete if it is guaranteed to find one of these EOAs, otherwise it is incomplete. If the search identifies a covering EOA it is successful.

2.3.1.1 Ellipsoid covering test

Any search strategy requires to determine whether or not a certain EOA covers the query point. The EOA is a hyper-ellipsoid represented by its center $\boldsymbol{c}$ (an n-vector) and by an lower-triangular Cholesky matrix $\boldsymbol{L}$ (an $n \times n$ matrix). Thus, the ellipsoid E is defined by [8]:

$$E \equiv \{\, \boldsymbol{x} \mid \|\boldsymbol{L^T}(\boldsymbol{x} - \boldsymbol{c})\| \leq 1 \,\} \tag{2.19}$$

Let's consider $\boldsymbol{L} = \boldsymbol{U\Sigma V^T}$ as the singular value decomposition (SVD) of $\boldsymbol{L}$: the columns of the orthogonal matrix $\boldsymbol{U}$ are the directions of the principal axes of the ellipsoid and the singular values σ_i, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$, (i.e. the components of the diagonal matrix $\boldsymbol{\Sigma}$) are the inverse of the length of the principal semi-axes λ_i, $\lambda_i = \sigma_i^{-1}$.
The smallest and the largest semi-axes are the radii of the inscribed and circumscribed hyper-spheres $r_{in} = 1/\sigma_1$ and $r_{out} = 1/\sigma_n$, respectively. Figure 2.6 shows a sketch of the inscribed and circumscribed hyper-spheres of an ellipsoid. The region of accuracy of every table record is represented by these four data: $\boldsymbol{c}$, $\boldsymbol{L}$, r_{in}, r_{out}.

The procedure to determine if an ellipsoid covers a point $\boldsymbol{x}$ of the composition space exploits first the radii information and, only if necessary, performs the evaluation of $\|\boldsymbol{L}^{\boldsymbol{T}}(\boldsymbol{x}-\boldsymbol{c})\|$.

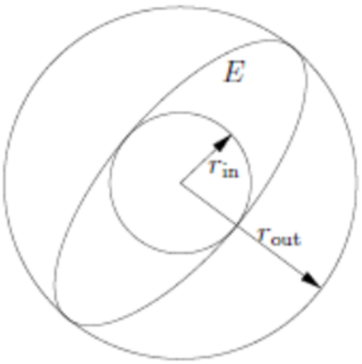


Figure 2.6 – Sketch of the ellipsoid showing the radii of the inscribed and circumscribed hyper-spheres [21]

First, the distance between the point $\boldsymbol{x}$ and the center of the ellipsoid $\boldsymbol{c}, r = \|\boldsymbol{x}-\boldsymbol{c}\|$, is calculated. This operation requires an order of n operations. Then, the distance is compared to the radii of the inscribed and circumscribed hyper-spheres. If the distance is greater than r_{out}, radius of the largest ball covered by the ellipsoid, the point is certainly outside of E. On the other hand if the distance is lower than r_{in}, radius of the inscribed hyper-spheres, the point is surely inside the ellipsoid. Only if r lies between r_{in} and r_{out}, the evaluation of $\|\boldsymbol{L}^{\boldsymbol{T}}(\boldsymbol{x}-\boldsymbol{c})\|$, which requires an order of n^2 operations, has to be performed to determine whether the point is covered by E.

2.3.1.2 Search methods

Four search methods can be used in ISAT in order to find if a leaf covers a given query point. These methods are denoted as: Binary Tree Search (BT), MRU Search, MFU Search [8] and Brute Force Search (BF). The first three strategies perform an incomplete search, whereas the last is complete.

The first issue regards which methods and in which order should be applied, with the aim to minimize the expected cost to successfully perform the search. As suggested by Pope et al. [8], the most efficient strategy is achieved by using all the methods in sequence and calling them in order of increasing cost. Moreover a method should be used only if its cost is less than the time spent to compute direct integration.

The methods are invoked successively starting with BT, secondly MRU, thirdly MFU and finally BF.

The Binary Tree Search uses a Binary Tree data structure which mainly consists of two different elements: the records, one for each leaf, and the cutting planes, one for each node. Given a query, the Binary Tree is traversed exploiting the information stored in the nodes until a leaf is reached; this leaf is called "primary leaf" of the query.

As mentioned before, each cutting plane consists of an n-vector $\boldsymbol{v}$ and a scalar a, all points $\boldsymbol{x}$ with $\boldsymbol{v}^T\boldsymbol{x} > a$ are deemed to be on the right of the cutting plane, whereas the others are on the left. Every node has two children, which can be two leaves of two other nodes: in the first case it is a terminal node whose children are two records; in the second case the node has the role to steer the query to the closest leaf.

The information stored in the cutting planes permit to steer the search to a leaf. Given a query $\boldsymbol{x}^q$, the root node of the Binary Tree is tested: $p = \boldsymbol{v}^T\boldsymbol{x}^q$ is evaluated and it is compared to the stored scalar a. If $p > a$ the search continues in the right branch of the tree, otherwise in the left one. The same procedure is repeated in each node encountered traversing the tree until the primary leaf is reached.

The EOA of the primary leaf is tested, using the ellipsoid covering test, to determine whether covers the query point. If the query point is inside, the search process is successful and the entire retrieve procedure terminates, otherwise the Binary Tree Search is unsuccessful.

After an unsuccessful BT search, the procedure continues using MRU search. MRU data structure is a linked list maintained on the leaves. The list is ordered so that the most *recently* used (MRU) leaf is on the top: every time that the search process is successful with whatever method, the retrieved leaf becomes the head of the list, thus the leaf used on the last retrieve is on the top. In this situation, "used" means used to retrieve from the table. At the beginning, MRU list is empty and has a fixed maximum dimension. Every time a leaf is retrieved, which means its EOA covers a query point, the leaf is added to the MRU list.

The MRU leaf addition algorithm is as follow:

1. If the given leaf is already in the list, it is moved to the head;

2. If the leaf is not in the list and the dimension of the list is less than the maximum, it is inserted at the head; if the list is full, the last entry is deleted and the given leaf is added on the top.

The MRU search procedure tests in succession a specified number of leaves at the head of the MRU list to determine whether their EOA covers the query point. The search terminates successfully when an EOA covering the query point is found.

After an unsuccessful MRU search, the procedure follows with MFU search. In the MFU list, the leaves are in the order in which they have most *frequently* been used, thus the head of the MFU is the leaf used more often in previous retrieves. Every time a leaf is retrieved, with whatever method, it is added to the MFU list.

The MFU leaf addition algorithm is as follow:

1. If the given leaf is already in the list, it is moved to the correct position that depends on the number of retrieve of the leaves;
2. If the leaf is not in the list and the dimension of the list is less than the maximum, it is inserted at the end. If the list is full, the leaf is added only if it has been used more times than the last element in the list.

A specified number of leaves in the MFU list are tested, skipping the one were previously tested in the MRU search. The search ends successfully if an EOA covers the query point. The number of leaves tested in the MFU is greater than the number in the MRU list, because in MFU search the ones already tested are skipped.

The likelihood of success of this two search methods depends on the nature of the application: MRU is efficient if there is high probability that successive queries are identical or differing by a small amount; MFU is effective when the most of the queries can be covered by a small group of leaves.

The final search method is Brute Force Search, this is the only method which performs a complete search: if an EOA covering the query point exists this methods guarantee to find it. The data structure used is the Binary Tree of the first methods. The tree is traversed starting from the most left record and every leaf is tested in succession to determine whatever it covers the query point. For large tables, this complete search becomes computational expensive and its cost can be grater the time need to a direct integration.

In this ISAT implementation, users could turn on and turn off the MRU, MFU and BF methods of search whereas Binary Tree Search is always working. Moreover, the user can set the dimensions of the linked lists and of the number of the leaf at the head that will be tested.

In this way, the user has some additional degree of freedom to enhance the performance of ISAT setting its feature depending the nature of the problem.

2.3.1.3 Interpolations

If the retrieve procedure is successful, which means that one of the above methods has found a leaf whose EOA covers the query point, the data stored in the leaf are used to evaluate the linear approximation of the function:

$$\boldsymbol{f}^{l}(\boldsymbol{x}^{q}) = \boldsymbol{f}(\boldsymbol{x}^{0}) + \boldsymbol{A}(\boldsymbol{x}^{0})(\boldsymbol{x}^{q} - \boldsymbol{x}^{0}) \tag{2.20}$$

If the error tolerance is small ISAT returns a linear approximation of the function with an error which is less than the tolerance

2.3.2 Grow

If the retrieve attempt is unsuccessful, the "primary leaf", i.e. the one returned from the retrieve procedure, becomes candidate for having its EOA grown. The function value is computed using direct integration and it is compared to the linearized estimate based on the leaf properties. If the approximation error is within the error tolerance the region of accuracy can grow:

$$\varepsilon = \|\boldsymbol{f}^{l}(\boldsymbol{x}^{q}) - \boldsymbol{f}(\boldsymbol{x}^{q})\| \tag{2.21}$$

Despite the query point is outside the ellipsoid the linear approximation is accurate. As mentioned above, the EOA is only a conservative approximation of the true region of accuracy and the ellipsoid may be grown as further information are generated.

The growing procedure is computationally more expensive than the direct integration, because it requires to compute the real value of the function and also to modify the EOA of the leaf. The additional computational expense incurred during the growth is offset by the increased probability of retrieving from that larger EOA during future steps.

For a given ISAT table size, growing the EOA is a cost-effective method that allows to increase the percentage of retrieves.

On the other hand, the enlarged EOA could be one of the main reasons of large local errors: the true region of accuracy (ROA) is the connected region of the composition space, containing the tabulation point $\boldsymbol{x}^0$, consisting of all the points $\boldsymbol{x}$ for which the error between the real value and the linearized estimate is under the specified tolerance. The EOA is a region that approximates the region of accuracy consisting in an hyper-ellipsoid. As a leaf is created, the EOA is initialized in a conservative way so there is a high probability that it is completely within the ROA. After the growing process, the EOA could be anymore entirely inside the true region of accuracy and this can originate local errors greater than tolerance.

As suggested by Pope et. al [5], there are three possible reasons to explain why inaccuracies rise due to growing process. First of all, the tabulated function could be significantly non-linear around the EOA, so there is the possibility of non-monotonic behavior of the error. In this situation, given a query point $\boldsymbol{x}^q$, sections of the segment between $\boldsymbol{x}^0$ and $\boldsymbol{x}^q$ may show an error greater than the tolerance, even though the grow point itself has an error less than the tolerance. The EOA is grown despite some regions are not inside the region of accuracy because the query point fulfil the growing conditions. The enlarged EOA encompasses these inaccurate regions, introducing the possibility that subsequent retrieves will be inaccurate. Secondly, the region of accuracy may be not convex and after the growing procedure the enlarged EOA could include inaccurate regions. As mentioned before, the geometry of the ROA in case of piecewise linear approximation could be, in some cases, hyperbolic (non-convex) instead of elliptic. Finally, even if the ROA is convex the growing procedure can lead to encompass inaccurate regions.

For this reason, ISAT implementation provides a method to control the number of growing times of a leaf, when it is reached the fixed maximum number of growth, the leaf is removed to reduce the probability of large errors.

2.3.2.1 Growing strategy

The growing method implemented is the one proposed by Pope [3]: given an existent EOA centered in $\boldsymbol{x}^0$, the new EOA is the unique hyper-ellipsoid of minimum volume, centered in $\boldsymbol{x}^0$, which encloses both the original EOA and the query point $\boldsymbol{x}^q$.

This is achieved performing the linear transformation that transforms the ellipsoid to the unit hyper-spheres and the query point $\boldsymbol{x}^q$ to a point $\widetilde{\boldsymbol{x}^q}$ in the transformed space. Then, the hyper-sphere is extended in the first direction to form an ellipsoid with $\widetilde{\boldsymbol{x}^q}$ on its edge. Finally, the linear transformation is inverted to obtain the grown EOA.

The initial ellipsoid E is defined by:

$$E \equiv \{\, \boldsymbol{x} \mid \; \|\boldsymbol{L}^T(\boldsymbol{x}-\boldsymbol{c})\| \leq 1 \,\} \tag{2.22}$$

Defining the transformed variable $\boldsymbol{y}$ by:

$$\boldsymbol{y} = \boldsymbol{L}^T(\boldsymbol{x}-\boldsymbol{c}) \tag{2.23}$$

the ellipsoid in the $\boldsymbol{y}$-space becomes a unit hyper-sphere $\|\boldsymbol{y}\| \leq 1$.

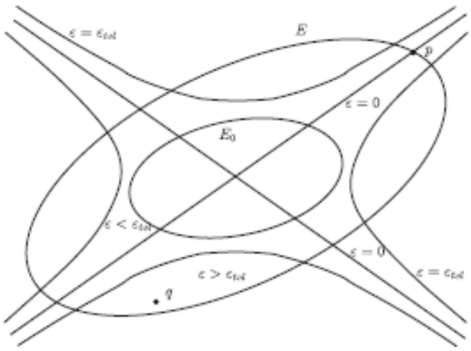


Figure 2.7 – Sketch of the ellipsoid growing for a hyperbolic (non-convex) ROA [8]

The transformation applied to the query point leads to:

$$\widetilde{\boldsymbol{x}^q} = \boldsymbol{L}^T(\boldsymbol{x}^q - \boldsymbol{c}) \tag{2.24}$$

In $\boldsymbol{y}$-space, the modified ellipsoid E' is the unit hyper-sphere extended to intersect the query point and it is defined using the rank-one modification algorithm [21]:

$$E \equiv \{\, \boldsymbol{y} \mid \; \boldsymbol{y}^T\boldsymbol{G}^2\boldsymbol{y} \leq 1 \,\} \tag{2.25}$$

where:

$$\boldsymbol{G} = \boldsymbol{I} + \gamma \widetilde{\boldsymbol{x^q}}\, \widetilde{\boldsymbol{x^q}}^{\boldsymbol{T}} \tag{2.26}$$

and γ is defined by the condition $\widetilde{\boldsymbol{x^q}}^{\boldsymbol{T}} \boldsymbol{G^2} \widetilde{\boldsymbol{x^q}} = 1$:

$$\gamma = \left(\frac{1}{\left\| \widetilde{\boldsymbol{x^q}} \right\|} - 1 \right) \cdot \frac{1}{\left\| \widetilde{\boldsymbol{x^q}} \right\|^2} \tag{2.27}$$

Thus, the Chloesky matrix $\boldsymbol{L'}$ of the grown ellipsoid can be obtained applying the LQ decomposition on the matrix $\boldsymbol{LG}$:

$$\boldsymbol{L' L'^T} = (\boldsymbol{L}\, \boldsymbol{G})(\boldsymbol{L}\, \boldsymbol{G})^{\boldsymbol{T}} \tag{2.28}$$

2.3.3 Adding

For a given query, if the retrieve and grow attempts fail, a new entry is added to the table. The location of the leaf is the query $\boldsymbol{x^q}$, the function value $\boldsymbol{f}(\boldsymbol{x^q})$ is known from the grow attempt and the mapping gradient $\boldsymbol{A}(\boldsymbol{x^q})$ has to be evaluated.

The operations involved in the adding process are the initialization of the leaf, in particular of the ellipsoid of accuracy, and its insertions into Binary Tree and into the search lists. The methods used when the table is full are discussed later.

The Binary Tree is unaffected by retrieving and growing, but it is modified as a results of an adding: the primary leaf is replaced by a new node whose two children are the new leaf, on the right, and the primary leaf, on the left.

First of all, it is discussed the evaluation of the mapping gradient matrix, secondly the initialization of the region of accuracy, thirdly how to set the new node and, finally, how to insert the new leaf.

2.3.3.1 Evaluation of the mapping gradient

The mapping gradient $\boldsymbol{A}$ is an $n \times n$ matrix whose components are:

$$A_{ij}(\boldsymbol{x}) = \frac{\partial f_i(\boldsymbol{x})}{\partial x_j} \tag{2.29}$$

In this situation, the equations are a set of ordinary differential equations with their initial conditions, thus these coefficients can be easily obtained.

The tabulation point is the initial condition of the direct integration, thus the first order sensitivity coefficients with respect to the initial condition are given by:

$$B_{ij}(\boldsymbol{x}^0, t) = \frac{\partial x_i(t)}{\partial x_j^0} \tag{2.30}$$

Then, the mapping gradients matrix is:

$$\boldsymbol{A}(\boldsymbol{x}^0) = \boldsymbol{B}(\boldsymbol{x}^0, \Delta t) \tag{2.31}$$

where Δt is the direct integration time interval.

It is possible to demonstrate that $\boldsymbol{B}$ evolves according to the linear system of differential equations:

$$\frac{d}{dt}\boldsymbol{B}(\boldsymbol{x}^0, t) = \boldsymbol{J}\big(\boldsymbol{x}(t)\big)\boldsymbol{B}(\boldsymbol{x}^0, t) \tag{2.32}$$

where $\boldsymbol{J}$ is the Jacobian of the system equations (evaluated through a numerical algorithm) and the initial condition is:

$$\boldsymbol{B}(\boldsymbol{x}^0, 0) = \boldsymbol{I} \tag{2.33}$$

where $\boldsymbol{I}$ is the identity matrix. The mapping gradients matrix is evaluated through the integration of the above differential equation system from the initial condition over the integration time interval.

In this ISAT implementation, this integration is calculated using the implicit (or backward) *Eulero* method, so that the mapping gradients evaluation is not too much time consuming.

Let $\boldsymbol{A}(\boldsymbol{x}^0) = \boldsymbol{B}(\boldsymbol{x}^0, \Delta t)$ and $\boldsymbol{B}(\boldsymbol{x}^0, 0) = \boldsymbol{I}$:

$$\begin{gathered}\frac{d}{dt}\boldsymbol{B}(\boldsymbol{x}^0, t) = \boldsymbol{J}\big(\boldsymbol{x}(t)\big)\boldsymbol{B}(\boldsymbol{x}^0, t) \Rightarrow \boldsymbol{A}(\boldsymbol{x}^0) - \boldsymbol{I} = \boldsymbol{J}(\boldsymbol{x}^0)\boldsymbol{A}(\boldsymbol{x}^0) \cdot \Delta t \\ \Rightarrow \boldsymbol{A}(\boldsymbol{x}^0) \cdot [\boldsymbol{I} - \boldsymbol{J}(\boldsymbol{x}^0) \cdot \Delta t] = \boldsymbol{I}\end{gathered} \tag{2.34}$$

Then:

$$A_{ij} \cdot (1 - J_{ii}\Delta t) - \Delta t \sum_{k \neq i}^{n} J_{ik} A_{kj} = I_{ij} \quad (i, j = 1, \dots, n) \tag{2.35}$$

The linear system reported above is sparse and structured: it consists of n diagonal, uncoupled blocks. Then, the mapping gradients are evaluated by solving n times dense $n \times n$ system instead of solving single, global $n^2 \times n^2$ original system. Another interesting feature of this system is that the coefficient matrix of all the n linear system is the same, so it is possible to factorize only one time that matrix, saving additional computational time. Figure2.8 shows the sparsity of the system matrix. In this ISAT implementation, a LU factorization is adopted and it is possible to choose between partial or full pivoting.

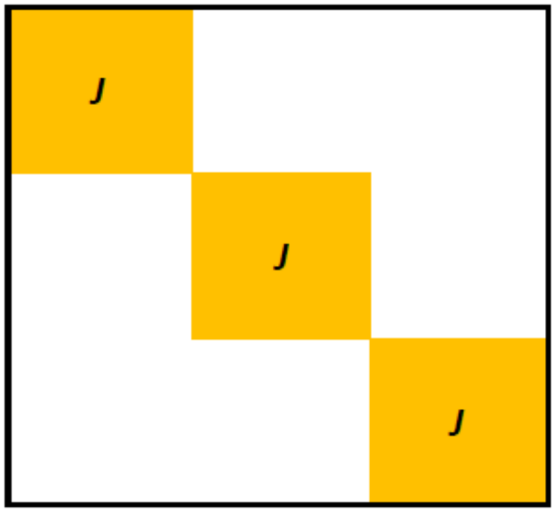


Figure 2.8 – Sparsity of the system matrix

2.3.3.2 Initialization of the EOA

The region of accuracy is given by:

$$R = \{ \boldsymbol{x} \, | \| \boldsymbol{A} \, (x - x^0) \| \leq \varepsilon_{tol} \} \tag{2.36}$$

In the constant approximation, on the boundary of the region of accuracy, the error is equal to the tolerance:

$$\delta \boldsymbol{x}^T \boldsymbol{A}^T \boldsymbol{A} \, \delta \boldsymbol{x} = \delta \boldsymbol{x}^T \boldsymbol{V} \boldsymbol{\Sigma}^T \boldsymbol{\Sigma} \boldsymbol{V}^T \delta \boldsymbol{x} = \varepsilon_{tol}^2 \tag{2.37}$$

where $\boldsymbol{U\Sigma V^T}$ is the singular value decomposition of $\boldsymbol{A}$, thus the region of accuracy, in the constant approximation, is defined by:

$$R = \{ \boldsymbol{r} \mid \boldsymbol{r^T V \Sigma^T \Sigma V^T r} / \varepsilon_{tol}^2 \leq 1 \} \tag{2.38}$$

This equation shows that the region of accuracy is a hyper-ellipsoid and the half-lengths of the principal axes are: $\lambda_i = \varepsilon_{tol} / \sigma_i$ [3].

The EOA is initialized as an ellipsoid with principal semi-axes in the directions of the columns of $\boldsymbol{V}$ and of length:

$$\lambda_i = \frac{\varepsilon_{tol}}{\widetilde{\sigma_i}} \tag{2.39}$$

where $\widetilde{\sigma_i} = \max(\sigma_i, 0.5)$. This assumption is taken into account to prevent that small singular values could generate unduly large principal axes.

The EOA is stored in the leaf as the lower-left triangular Cholesky matrix $\boldsymbol{L}$, that is evaluated performing the LQ decomposition:

$$\boldsymbol{A = L L^T = V \widetilde{\Sigma}^2 V} \tag{2.40}$$

Finally, the radii of the inscribed and circumscribed hyper-spheres are stored as the largest and smallest λ_i.

2.3.3.3 Initialization of a node

The cutting plane at the node is defined by a n-vector $\boldsymbol{v}$ and a scalar a: points $\boldsymbol{x}$ with $\boldsymbol{v^T x} > a$ are deemed to be on the right of the cutting plane, whereas the others are on the left.

The cutting plane is set to be the perpendicular bisector of the line that connects the center of the two leaves, in the linearly transformed space where the EOA of the primary leaf is a unit hyper-sphere.

The algorithm to evaluate vector $\boldsymbol{v}$ and a scalar a is reported below.

Let E_1 the ellipsoid of the primary leaf, centered in $\boldsymbol{x^0}$:

$$E_1 \equiv \{ \boldsymbol{x} \mid \left\| \boldsymbol{L_1^T (x - x^0)} \right\| \leq 1 \} \tag{2.41}$$

Let $\boldsymbol{x}^q$ the center of the second ellipsoid. It can be defined:

$$\boldsymbol{x}_h = \frac{1}{2}(\boldsymbol{x}^q + \boldsymbol{x}^0) \tag{2.42}$$

The separating hyper-plane is defined as:

$$H = \{\, \boldsymbol{x} \mid \boldsymbol{v}^T(\boldsymbol{x} - \boldsymbol{x}_h) = 0 \,\} \tag{2.43}$$

and the vector $\boldsymbol{v}$:

$$\boldsymbol{v} = \frac{\boldsymbol{L}_1^T \boldsymbol{L}_1 (\boldsymbol{x}^q - \boldsymbol{x}^0)}{\|\boldsymbol{L}_1^T \boldsymbol{L}_1 (\boldsymbol{x}^q - \boldsymbol{x}^0)\|} \tag{2.44}$$

Finally the value of the scalar a is defined by:

$$a = \boldsymbol{v}^T \boldsymbol{x}_h \tag{2.45}$$

2.3.3.4 Insertion of the new leaf

If the Binary Tree is not full, the new leaf is inserted into the table. The primary leaf is replaced by a node, set up as mentioned before, whose children are the primary and the new leaves, as shown in Figure 2.9.

The insertion of a new leaf is the most time consuming task of ISAT because it requires to perform a search procedure, to compute the direct integration, to solve n linear systems to evaluate the mapping gradient, to initialize the region of accuracy and to set up the new node. It is certainly more expensive than to perform a direct integration, but the benefits are that the new leaf could be used in future to perform very fast linear approximation.

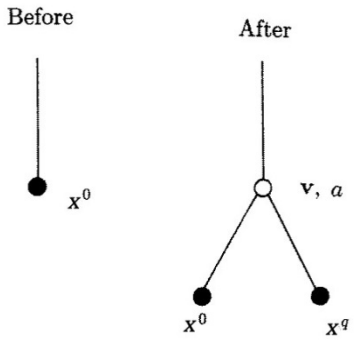


Figure 2.9 – Sketch of part of the Binary Tree before and the after the addition of the record at $\mathbf{x}^0$

2.3.4 Treatment of the full table

The number of table entries can continually increase, especially for problems with large n, until the available memory is fulfilled. It is necessary to limit the table size fixing a maximum number of leaves and to specify the behavior of the adding procedure if the table is full.

When the table is full and a new has to be added, the three different actions considered are:

1. Return the value of the function (evaluated in the grow attempt) and do not alter the table.
2. Clear all the table and insert the leaf in MRU and MFU lists.
3. Delete the least recently used leaf (at the tail of MRU list) and replace by the new leaf [8].

The best strategy depends on the nature of the problem, in particular relies on whether or not the problem is statistically stationary. In test performed, it is found that the first method is inefficient due to the non-stationary behavior of the problems analyzed. The distribution of the queries moves away from a region of the composition space and never return. The leaves stored at the beginning of the simulation, after a certain point, will be not used anymore because the composition space is different from the one they represents. Thus, some of the leaves of the table will not be used anymore after a certain number of events. Returning the function value without altering the table determines that the most part of the queries, after a certain point, will be solved using the direct integration because the stored leaves represents a composition space that it is no longer present in the system. The result is that the simulation will slow down. For this reason the second and the third methods are more effective.

The second strategy clears all the table entries and reinsert into the empty Binary Tree only the leaves that were in the MRU and MFU lists. This method reaches acceptable yields only if the dimension of the two lists is sufficiently high. The new tree contains only a few of the previous leaves and it could lead to loss of computational efficiency, because of the high number of addition procedure needed in the near future. If the lists are very large, the retrieve procedure could become slower due to the high number of ellipsoid covering tests required. To overcome this problem, it is possible to set to

different number the dimension of the lists and the number of the leaves tested in each of them. A large dimension of the lists is set to increase the efficiency of the full table treatment, but, at the same time, a small number of them are tested in each search procedure to reduce the number of ellipsoid covering tests.

The best performances are reached using the third method. The leaf at the tail of the MRU is deleted from both the linked list and the Binary Tree and it is replaced by a new leaf. The size of the table keeps constant avoiding full memory problems. On the other hand, the new leaf has a small EOA compared with the one of the removed leaf and this can be a penalty due to the subsequent growth of the new leaf. If the problem is unsteady, the distribution of queries shift in the composition space and old leaf can be deleted without penalty.

2.3.5 Cleaning and Balance

The retrieve procedure traverses the tree to find the leaf which is closest to the query point. The time spent to reach the primary leaf increases as the size of the table increases. This happens because the query has to follow a longer path through the nodes of the tree. Moreover, the BT data structure performs a time-optimal search if it is balanced which means, as mentioned before, that the depth of the left and right sub-trees differs by one or less. A larger, unbalanced tree performs a time-worse search procedure than a balanced one.

In this work, ISAT deals with unsteady problems. The distribution of the queries moves away from a region of the composition space and never returns. After a certain time, some of the leaves stored in the table become useless, because they are in a region of the composition space that is not accessed anymore. The size of the table increases but only the “newest” leaves are used.

The growing procedure could cause inaccurate retrieve, as discussed above, so it is important to control the growth of the ellipsoid to minimize the possibility to encompass, inside the EOA, region of the composition space which shows a local error greater than the tolerance.

Due to these three reasons, a clean and balance procedure is implemented in the present work:

1. As suggested by Contino [22], after every growing procedure, it is important to check the number times the EOA is grown. If the maximum number of growth allowed is overtaken, the leaf is removed from the table.
2. Periodically, the table is entirely browsed to find the leaves that are too “old”, which means the leaves that have been used the last time before a fixed time and, now, are useless due to the shift of the distribution of the queries.
3. The height of the Binary Tree is compared to a user maximum value to decide if it is necessary to balance the tree. The balance algorithm, as proposed by Contino [22], looks for the directions of the composition space which show the greater variance, separates the space in two parts with an hyper-plane perpendicular to the variance vector. Then, this hyper-plane is used as the cutting plane of the root node and re-insert the leaves randomly to achieve a good balance.

The first strategy avoid large errors due to inaccurate retrieve. The second allows to control the dimension of the table deleting the useless leaves without speed penalties, because the removed entries are not used anymore; the third allows to maintain balanced the data structure optimizing the search time.

The second and the third procedures are quite time consuming because the former needs to scan the entire tree and the latter to re-built the tree starting from an empty one.

In particular the balancing procedure is executed only if strictly necessary which means that the tree is very unbalanced.

3 ISAT Implementation and validation

In this chapter, the `ISATLib` library implementation is explained in detail and, afterwards, `ISATLib` predictions of simple test cases are validated comparing to direct integration results. Test cases consist in a set of stand-alone batch reactors which are integrated over a fixed number of time steps using both ISAT and direct integration. Last, a sensitivity analysis upon the main table parameters is carried out to investigate their effects on the performance.

3.1 `ISATLib` library

`ISATLib` is a C++ object-oriented library implementing the ISAT algorithm. It consists of four classes, each with its particular feature, as presented in Figure 3.1:

- `ISATLib` class: performs the main task of ISAT algorithm such as retrieve, grow, addition, interpolation;
- `binaryTree` class: manages the Binary Tree data structure used by ISAT;
- `binaryNode` class: represents the nodes of the Binary Tree;
- `chemComp` class: represents the leaves of the Binary Tree;

In this implementation of the ISAT algorithm [3], the library provides only storage and retrieval features. The main framework has to provide both a direct method to evaluate the function and a way to compute the sensitivity coefficients.
These functions are not inside the library to let ISAT framework being intentionally general and to allow a wide application field. In this way, the library can be as often it is necessary to evaluate many times a certain function $\boldsymbol{f}(\boldsymbol{x})$. In the present work, `ISATLib` will be used to solve initial value problems, thus the main numerical code has to provide the method to integrate di differential equations system and to evaluate the

mapping gradient matrix. Different ODE solvers can be implemented by the main numerical code without affecting the library due to its general structure not linked to the differential equation system solver adopted.

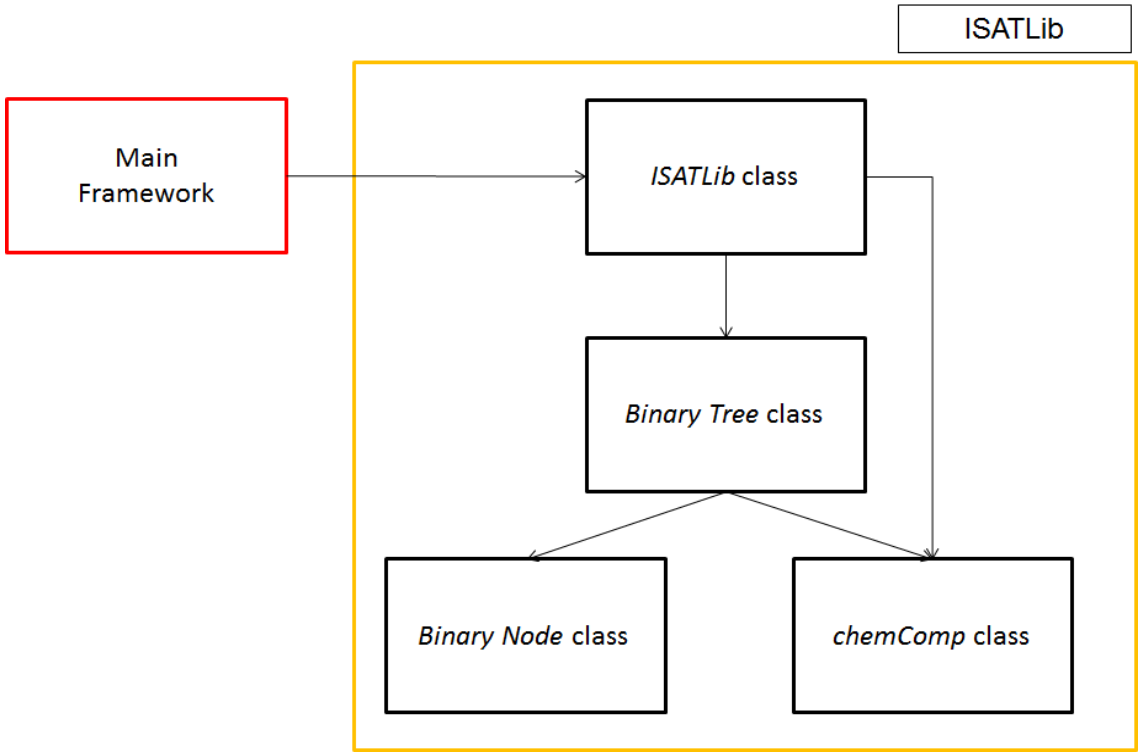


Figure 3.1 – `ISATLib` library architecture

The structure of the library let its implementation, also, in other problem which requires several function evaluation as optimization problem [19] or DAE solving [6, 20]. The differences between these problems and the one considered in this work are not in a different version of the library, but in a different way the main numerical code will evaluate the function and the sensitivity coefficients.

To improve library performances, it is important that the order of magnitude of each element of the composition vector is quite the same. If the order of magnitude of the different components cover a very large spectrum of values, the function variations are surely larger than ones leading to very frequent unsuccessful retrieves. Thus, it is necessary to scale these values to have, during the simulation time, the input vectors of *`ISATLib`* consisting of elements of the same order. The main numerical code has to perform this scaling operation, because the ISAT library works only with the scaled

vectors. The scale operation is performed multiplying each component of the composition vector for a proper scaling value set by the user.

Moreover, the user can set a value of the scaling error factor (Section 2.1.3) for each species in order to increase or decrease the accuracy of their approximation. If the scaling value is larger than one, the error on this species is artificially enhanced during the error evaluation obtaining a high accurate prevision.

ISATLib uses Eigen library [23] and its function in order to manage vectors and matrixes operations.

3.1.1 *ISATLib* class

It is the main class of the library and performs the main tasks of the algorithm, as presented in Figure 3.2:

- query management (retrieve, grow, addition, interpolation);
- table management (clean and balance);
- search list management;

It is the only class that communicates directly with the main framework, In order to be initialized the user has to provide the following data:

1. *scaleError*: a n-vector whose components are the weights of the errors on the species;
2. *epsTol*: the error tolerance of ISAT
3. *nCols*: number of the species;

During the initialization, the Binary Tree is created through a new instance of the *binaryTree* class. Also the simulation start time is stored and will be used during the balance operations.

3.1.1.1 Query management

In *ISATLib* class the functions necessary to perform retrieve, grow, addition and interpolation are collected. They represents the core of the library because they manage all the queries submitted to the library.

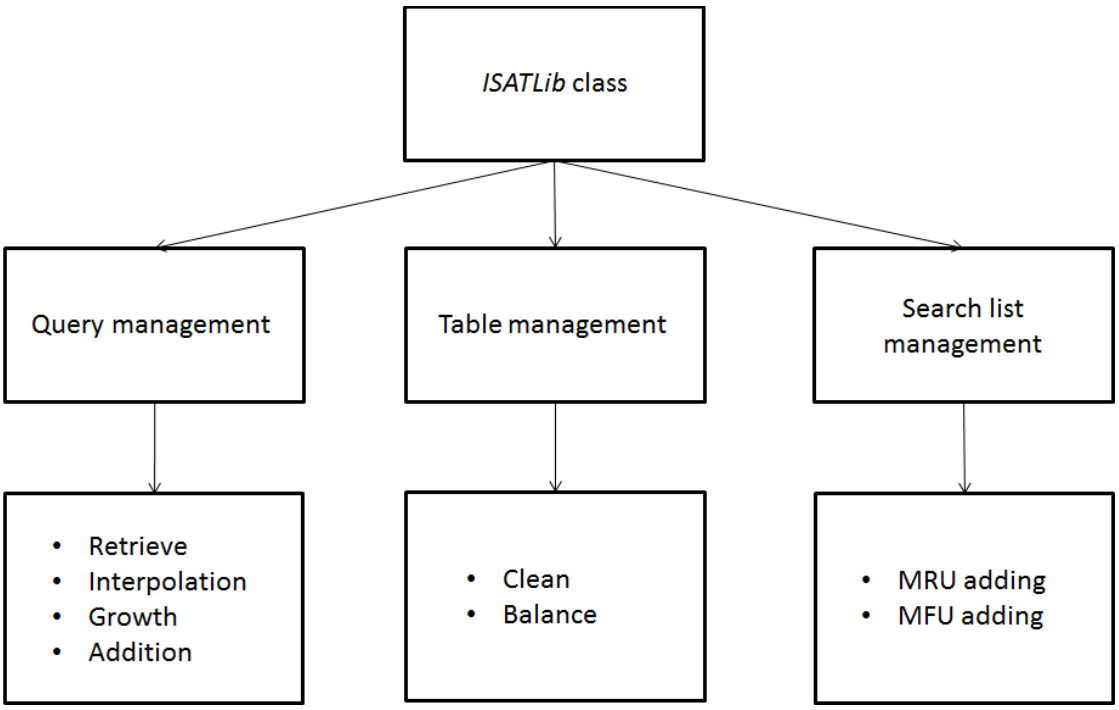


Figure 3.2 – `ISATLib` class structure and methods

3.1.1.1.1 retrieve function

The main software calls the retrieve function providing the query point, i.e. scaled composition vector, and the function uses four different strategies to search if at least a leaf covers the query point. The retrieve procedure is presented in Figure 3.3.

The first technique is the Binary Tree search (BT Search), i.e. the query navigates in the tree exploiting the information in the nodes in order to reach a leaf. The data stored in the leaf, concerning the region of accuracy, are used to make the *Ellipsoid covering test* which is performed by a function of the *chemComp*.

If the result of the test is positive the search procedure ends and the found leaf is added both to the MRU and MFU lists, i.e. *most recently used* and *most frequently used* lists, using dedicated functions of this class.

The BT Search is the only search method that is always used, the others – MRU search, MFU search, BruteForce Search - can be turned on or off by the user according to the needs. It is suggested [7] to switch on MRU and MFU searches because these methods improve ISAT performance. As described in Section 2.3.1, the BT Search is incomplete,

i.e. it is not guaranteed that the search ends with success even if more than one leaves cover the query point. Thus, the MRU and MFU lists permits to increase the probability to find a leaf that covers the query point reducing the number of slow function evaluation.

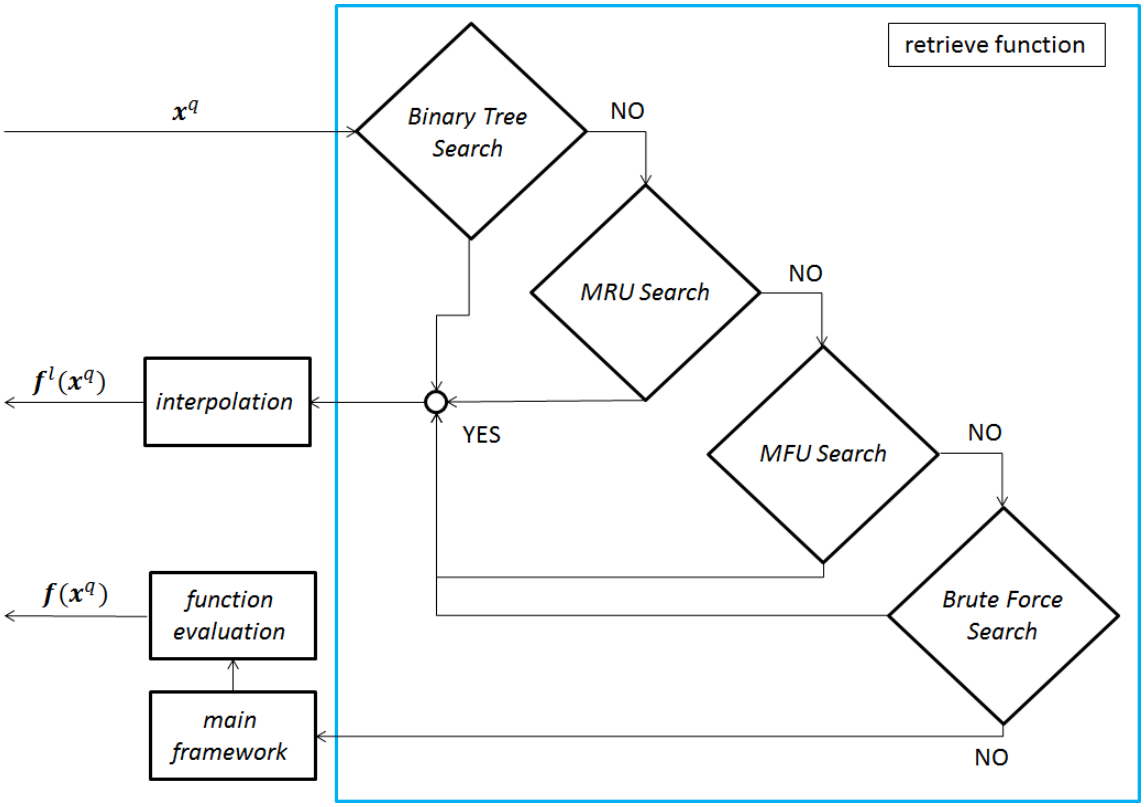


Figure 3.3 – Flowchart describing retrieve function

The BruteForce Search would be recommended only if the tree is very small because, when the tree becomes larger, it slows down too much the retrieve procedure.

The MRU search method performs the *Ellipsoid covering test* on a certain number of leaves that are on the top of the *most recently used* list. The user can define both the maximum number of leaves and the maximum number of leaves to be checked inside the list. If one of the selected leaves covers the query point, this method is successful, the retrieve procedure terminates and the leaf is added both to MRU and MFU.

The MFU search method checks if a certain number of leaves on the top of the *most frequently used* list covers the query point. Also here, the user can define the number of leaves contained inside the list and the number of leaves in the list to be checked.

Among the leaves of the MFU list, only the leaves that are not been already checked are tested to determine whether an EOA covers the query point.
The last search procedure is the BruteForce search: starting from the first leaf on the left, the entire set of the tree leaves is scanned, the procedure ends when a leaf that covers the query point is retrieved or if the rightmost leaf is reached. In the second situation, it means that none of the leaves covers the query point. This method is used only when the size of the tree is small, because if the tree is large the time spent to retrieve a leaf is of the same order of magnitude of a direct integration.
The retrieve procedure identifies, using the methods mentioned above, the leaf which is the closest to the query point, thorough the *Ellipsoid covering test* it is evaluated if the leaf covers the query point: if it is true the interpolation is performed and the linear approximation is returned to the main numerical code. If none of the leaves tested during the procedure covers the query point, the retrieve is unsuccessful and the library asks to the main numerical code to evaluate the function value directly, i.e. solving the ODEs system.

3.1.1.1.2 `interpolation` function

If the retrieve procedure is successful the main software calls the *interpolation* function to evaluate the linear approximation. It is necessary to provide the query point and the retrieve leaf as argument of the function.
The function values can be forced to satisfy some kind of constraints like non negative-values. The library returns the approximations without checking if these constraints are respected, thus the main software has to carry out this control itself.

3.1.1.1.3 `grow` function

If the retrieve procedure is unsuccessful, i.e. the closest leaf do no cover the query point, the direct evaluation of the function is performed by a main numerical algorithm dedicated method. The function structure is presented in Figure 3.4.
The query point and its function value are provided to the *grow* function to check if it is possible to modify the EOA in order to cover the query point.

This function calls a method in the *chemComp* class which evaluates the error in the linear approximation and compares it to the tolerance; only if the former is lower than the latter a *chemComp* function performs the growing algorithm.

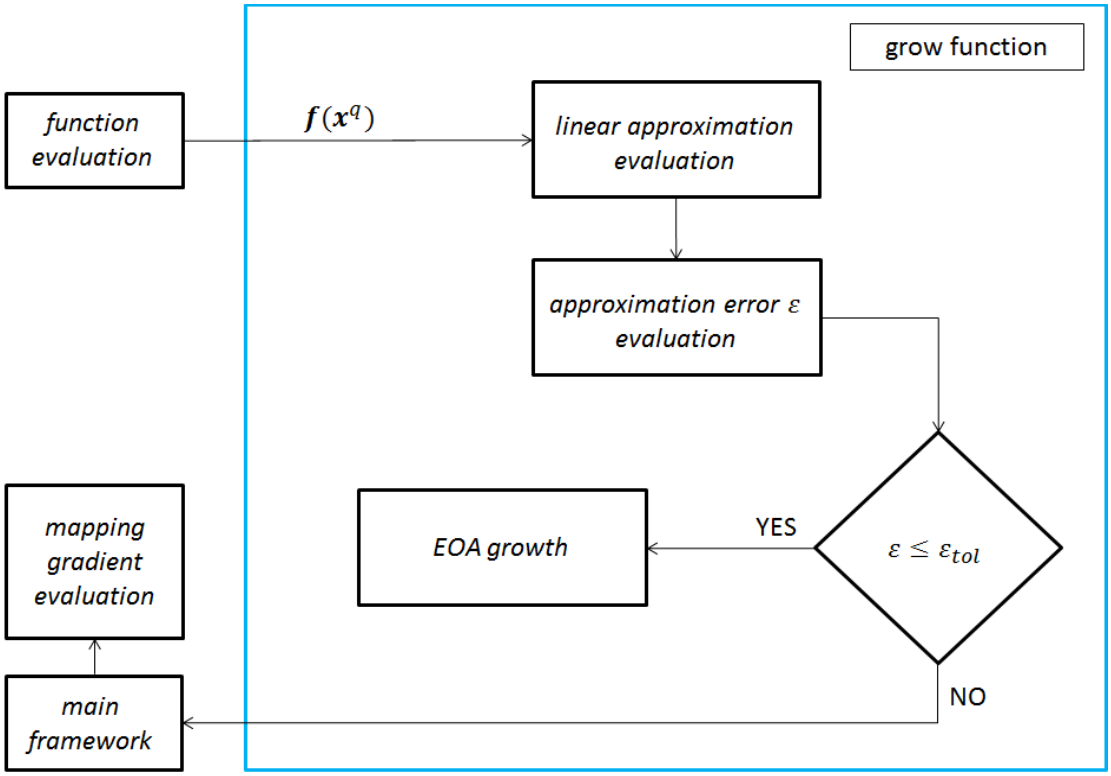


Figure 3.4 – Flowchart describing grow function

Following up a certain number of growths, the EOA could include region of the composition space where the error is larger than the tolerance, leading to an undesirable large approximation error. Thus, if the leaf is grown more than a threshold value fixed by the user, the procedure is interrupted and the leaf is added to a list of records that has to be removed. The user sets the *maxGrowCoefficient* and the routine compares the number of the growth of the leaf with the limit value obtained multiplying the coefficient per the maximum allowed size of the binary tree. If the first is larger the leaf does not grow and it is added to the remove list.

3.1.1.1.4 add function

If the error is larger than the tolerance, the main software evaluates the mapping gradient matrix and, then, calls the *add* function, providing the query point, its function

value, the mapping gradients and the closest leaf as arguments. The function structure is presented in Figure 3.5

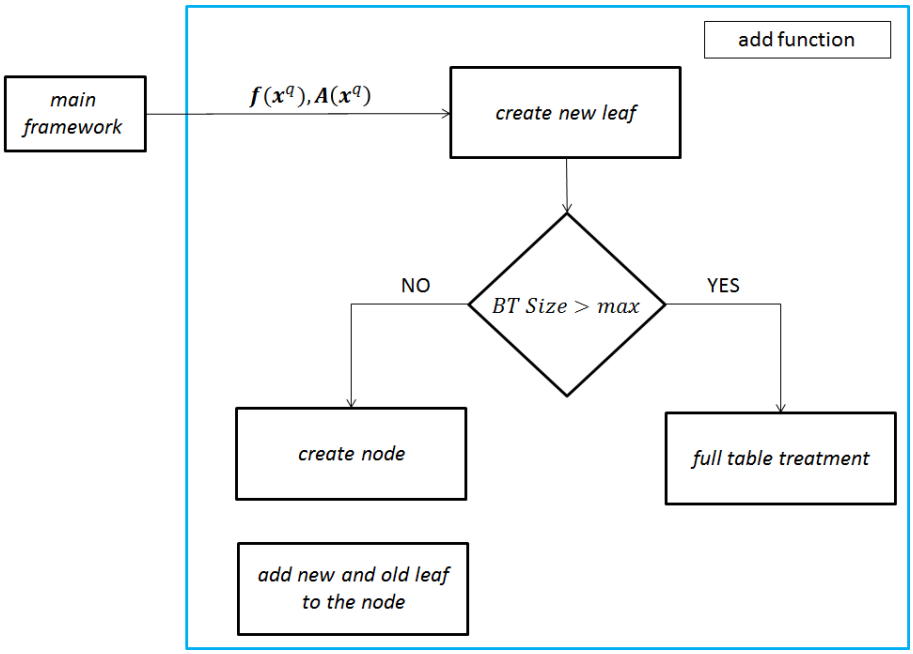


Figure 3.5 – Flowchart describing add function

The function calls a `binaryTree` method that creates a new leaf instance and substitutes the old leaf with a node – creating a new instance of the `binaryNode` class – whose children are the old leaf and the new one, respectively on the leaf and on the right.

The add function checks if the tree is full, by comparing its size with the maximum dimension allowed by the user. If the tree is not full, it calls a `binaryTree` class function, which substitutes the closest leaf with a node increasing the tree dimension.

If the number of leaves is larger than the maximum value, two different scenario can occur. In the first case, the entire tree is cancelled. Then, both the new record and the leaves in the MRU list are inserted in the empty tree. In the second case, the least used leaf of the MFU list is erased – the one on the tail of the list – and the new leaf is inserted. In this way the dimension of the tree remains constant and equal to the maximum size. The second method has the best speed-up performance because all the existent leaves are maintained and there is a high probability that the next query can be covered by these ones. On the other hand, the first procedure can manage good results

if the system is strongly unsteady and the dimension of the MRU list is over 150 leaves. The structure of the full table treatment is presented in Figure 3.6.

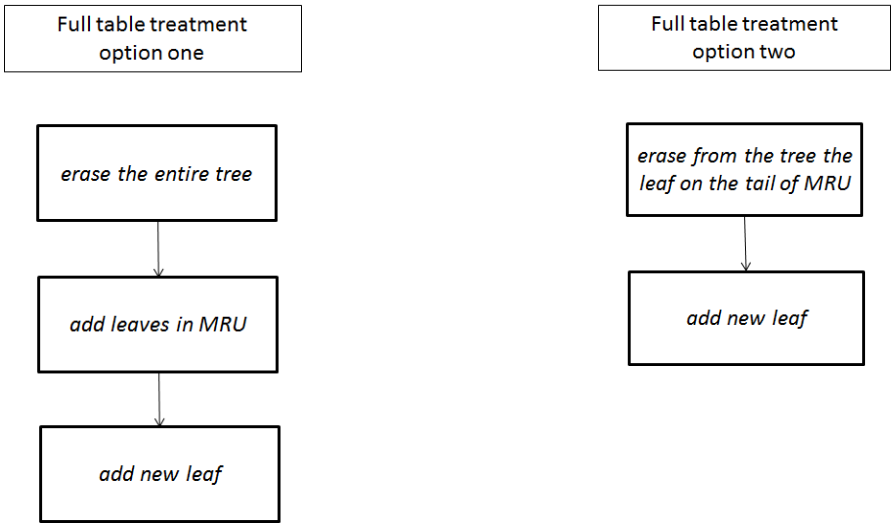


Figure 3.6 – Full table treatment strategies

After every leaf insertion in the tree, the leaf is added to the MRU and MFU list using the dedicated ISAT class function.

3.1.1.2 Table management

The table management consists in the operations performed to ensure good computational performances of the storage table.

The procedure is performed by the *cleanAndBalance* function which is called by the main software at the end of every query management procedure.

Firstly, the leaves inside the remove list are cancelled. This operation is accomplished every time the function is called to remove from the tree the leaves that are grown too many times with the purpose to minimize the possibility of large approximation error.

The subsequent cleaning operations are performed after a certain number of events, which means indifferently retrieve, grow or add. The number of added leaves and the number of retrieves from the previous balance are compared to threshold values. If at least one is larger than the fixed value, the clean operations are performed. The entire

tree is scanned and a leaf is added to the removed list only if at least one of the following requirements is satisfied:

- the number of retrieves of the leaf is larger than a threshold value obtained multiplying the *maxUsedCoeff* times the maximum table size.
- the leaf last retrieve is older than a limit value obtained multiplying the total simulation time – from the start time till the balance time – times the *maxTimeOldCoeff*.

If the leaf is too old or too retrieved, it is added to the remove list and cancelled from the tree. The cleaning operations remove from the table the records that are unused and the ones that can lead to gross errors. They can arise for two reasons: the EOA is grown too much or the leaf has been used too many times.

At last, the height of the tree is compared to a threshold value obtained multiplying the *maxHeigthCoeff* per the ideal depth of the tree evaluated on the actual size of the tree. If the former is larger than the limit value the *binaryTree* balance function is called. The table balance procedure is time consuming and it has to be performed only when the value of the height of the tree is elevated, because only in this case the benefits are sufficient to recover the time spent and to speed-up the following table procedure.

3.1.1.3 Search list management

ISAT class provides two other functions whose purpose is to manage the search lists. In particular, they perform the list add operation ensuring that the leaves in the list are in the correct order. A leaf is inserted in the lists after every successful both retrieve and adding procedures.

Inserting a leaf in the MRU requires the following steps: first, the leaf is searched in the list: if already inside, it is removed and inserted on the head of the list. If not inside, the list dimension is compared to the maximum allowed: if it is equal, the record on the tail is removed and the new leaf is added on the head; if it is smaller, the leaf is inserted on the head.

The procedure to insert a leaf in the MFU, schematized in Figure 3.7, is as follows:

1. The list is scanned to check if the leaf is already inside:

2. If the leaf already belongs to the MFU, it is moved through to the correct position. To perform this task the list is scanned from the bottom comparing the number of retrieves of the leaf with the same one of the list leaves. The leaf is inserted when its retrieve number is larger than the one of the previous leaf.
3. If not inside, the list dimension is compared to the maximum allowed: if it is equal, the number of retrieves of the last leaf in the list is compared to the one of the inserting: if the latter is larger than the former, the leaf on the tail is removed and the new one is inserted. If not full, the leaf is inserted on the tail.

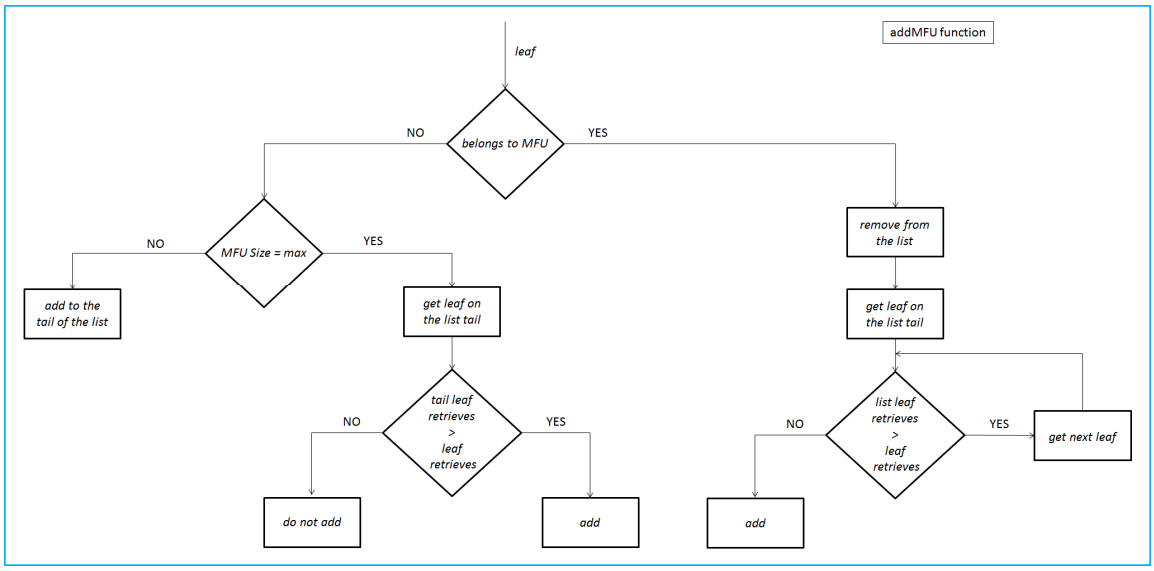


Figure 3.7 – Flowchart of addMFU function

3.1.2 `binaryTree` class

The ISAT tabulation method is a Binary Tree data structure, so it is necessary to provide a class that can create, manage this storage technique.

Creating a new `binaryTree` instance means to initialize the tree as a single node without any children. Starting from this first node, the tree is built up adding leaves when necessary.

The class provides, also, the functions to perform the required operation on the tree:

- search of a leaf
- insertion of a leaf and node

- removal of a leaf
- balance the tree

3.1.2.1 Tree modify operation

The procedures to search, insert and remove a leaf are described in this Section. The tree is built using the pointer C++ structure in order to reach good performances. This lets to create the virtual memory links that are necessary to build the tree from the roots to the leaf through all the branches exactly as happens in a natural tree.
In the next subsections, the functions used to manage the tree are described in detail.

3.1.2.1.1 `insert` function

The first insertion procedure sets the adding leaf as the only child of the root node, storing it on the left side. The second addition modifies the tree as follows: in the root node is evaluated the cutting plane in order to have the new leaf on the right and the old leaf on the left, the existing leaf becomes the left child and the new is stored on the right. From the third addition onward, each inserting procedure determines that an existing leaf becomes a node: the function creates a new node instance, which evaluates the cutting plane, and adds the existing leaf and the new one respectively on the leaf and on the right of the separating plane.

3.1.2.1.2 `search` function

Given a certain query point, the search procedure always permits to find a leaf that is, in some sense, close to the query. Figure 3.8 show the flowchart of the function.
Each search procedure exploits the cutting planes information to stir across the tree: starting from the root node, the dot product between the query point and the v vector is evaluated. The result is compared with the a value: if it is larger the search procedure continues in the right branch; on the other case on the left one. The procedure continues performing the mentioned above comparison until a terminal leaf is reached.
Every comparison between the query point and the information stored in the node is computational cheap but permits to neglect in the next searches a large number of leaves, speeding up the procedure. To achieve the best performance, it is necessary that

the tree is well-balanced, because, in this case, its depth is minimized and the number of comparisons to reach a leaf is minimized. Moreover, if the tree is perfectly balanced every comparison permits to neglect the half of the remaining leaf, enhancing the speed of the search procedure It also allows to increase the possibility that the reached leaf covers the query point.

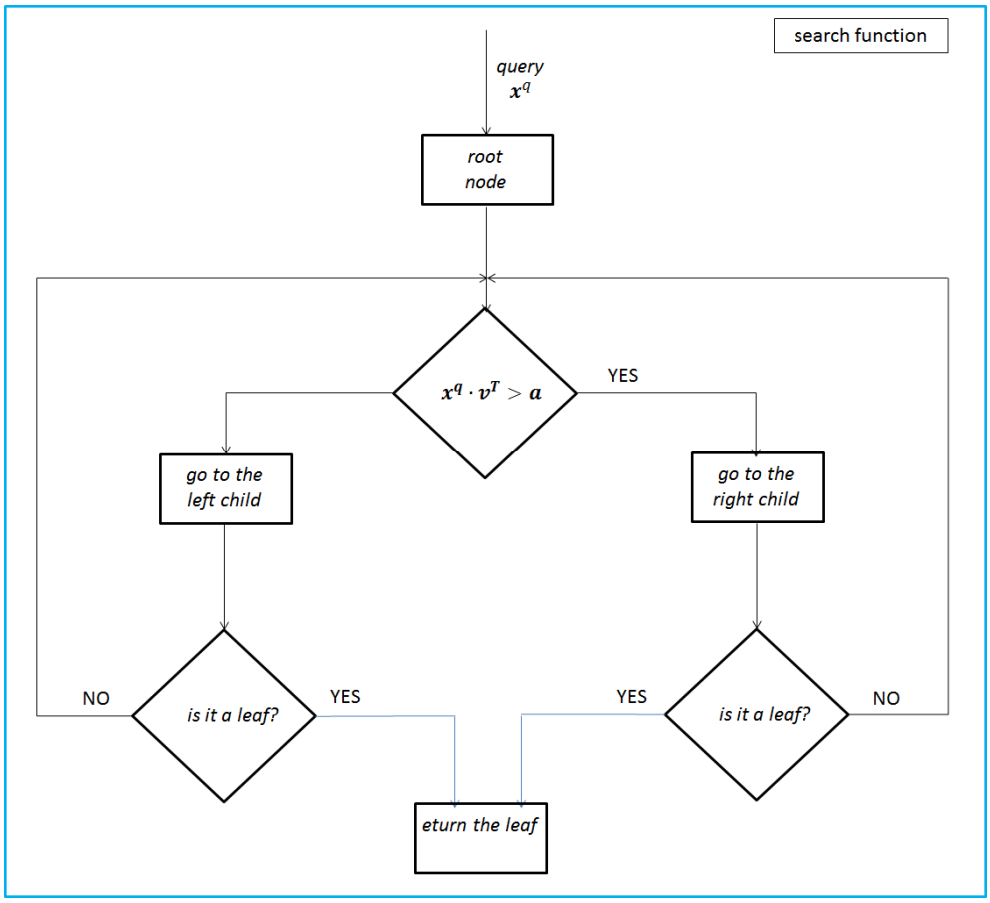


Figure 3.8 – Flowchart of search function

3.1.2.1.3 `balance` function

The balance function operates collecting the entire set of leaves in the binary tree inside a list. As suggested by Contino [22], the list is traversed and the mean value and the variance in each direction of the composition space are evaluated. The direction showing the highest variance is identified and it is named principal direction. The list is traversed again to identify which leaf shows the maximum and minimum value in the principal direction. Ideally, these leaves are the rightmost and left most records of the balanced tree. The balance is performed considering the high variance direction,

because it is the one which shows the highest fluctuating values permitting a good leaf distribution in the balanced tree.

In the next step, the tree is cleaned, which means that each leaf and each node are deleted. The selected leaves are inserted in the empty tree. The list containing the remaining leaves is randomly shuffled and, then, the leaves are inserted in the tree.

The function allows to reshape the tree in order to reach a depth that is equal to the ideal one, good results are achieved because the balanced tree shows a very low height and a balanced depth value similar to the ideal one. The resulting height of the tree is certainly below 10 and this guarantees that the subsequent search procedures are faster.

On the other hand, this procedure is quite time consuming, especially in the case of large trees, because it requires to scan the entire tree, to memorize the leaf in a list and to reinsert the set of leaves again. As mentioned in Section 2.2.3, the height of the tree is the difference between the depth of the left and right branches. A perfectly balanced tree shows a zero height. Due to the procedure computational expense, it is called only when it is compulsory, which means that the height of the tree is too much larger than zero.

3.1.2.1.4 `remove` function

The purpose of this function is to remove a certain leaf. It is used when the tree has to be cleaned or cleared. It removes the selected leaf and reshapes the tree in order to respect that every node must have two children which can be either nodes or leaves.

First, it is checked if the sibling of the remove leaf is another leaf or a node. If the sibling is a leaf, the parent node has to be removed and the remaining leaf has to be inserted as a child of the “grandparent” node on the correct side of the cutting plane, which is the one of the removed node. If the sibling is a node, it is necessary to perform a transplant operation, which permits to respect the correct side of the parent node with respect to the “grandparent”.

3.1.3 *chemComp* class

The *chemComp* class represents each leaf of the Binary Tree. Thus, every time it is necessary to add a new record to the tree, a new class instance is created. The table record consists in:

- $\boldsymbol{x}^0$: center of the leaf
- $\boldsymbol{f}(\boldsymbol{x}^0)$: function value in the leaf center
- $\boldsymbol{A}$: mapping gradient
- $\boldsymbol{scaleError}$: vector representing the scaling error of each species
- $\boldsymbol{epsTol}$: error tolerance
- $\boldsymbol{L}$: matrix representing the EOA
- r_{min} and r_{max}: minimum and maximum radius of the hypersphere circumscribing and inscribing the EOA
- *nUse*, *nGrow*: counter which represents the number of time a leaf is retrieved and grown

The first three items are computed once and are fixed, whereas $\boldsymbol{L}$, r_{min} e r_{max} change whenever the EOA is grown.

The leaf initialization requires the three first data as arguments and evaluates the EOA parameters using the algorithm described in Section 2.3.3.4. The *chemComp* class provides also the function to manage the ellipsoid of accuracy

3.1.3.1 EOA management

The class provides the function to initialize the EOA and also the method to check if the OEA covers a certain point and the ones necessary to perform its growth

3.1.3.1.1 `initializeEOA` function

The *initializeEOA* function permits to set up the ellipsoid of accuracy using the algorithm described in Section 2.3.3.2.

First, the singular value decomposition (SVD) of the mapping gradient $\boldsymbol{A}$ is evaluated and the singular values are modified so their maximum value can be at least 0.5. This modification is performed to prevent unduly large principal axes due to small singular

values. Then, the modified mapping gradient $\widetilde{\boldsymbol{A}}$ is computed again using the information obtained in the SVD decomposition.

Second, the QR factorization of the transposed of product between the scaling error matrix and the modified mapping gradient, $\boldsymbol{B}\widetilde{\boldsymbol{A}}$, is evaluated. The results of the factorization are two matrixes: an orthogonal matrix and an upper-triangular matrix. The second one is the transposed of the lower-triangular Cholesky matrix $\boldsymbol{L}$ which represents the ellipsoid of accuracy.

The main diagonal of the Cholesky matrix is controlled to avoid the presence of negative values. if this is the case, these values are substituted by their opposite.

The best accuracy and stability performance are obtained using a full pivoting Householder algorithm to perform the decomposition.

Last, the singular value decomposition of the lower-triangular Cholesky is evaluated in order to obtain the largest and the smallest singular values which are the smallest and the largest radii of the inscribed and circumscribed hyper-spheres, r_{min} and r_{max}.

3.1.3.1.2 `inEOA` function

The *`inEOA`* function performs the *ellipsoid covering test* as described in Section 2.3.1.1; the query point is provided as an argument and the function returns a positive or negative flag depending on the test.

First, the distance r between the query and the center of the leaf is evaluated, computing the norm-2 of the difference between the vectors. It is compared to the r_{min} and r_{max}: if larger than r_{max} a negative flag is returned, whereas if lower than r_{min} a positive flag is returned. If the value is between r_{min} and r_{max}, the product between the transposed of lower-triangular Cholesky matrix $\boldsymbol{L}$, representing the EOA, and the distance is evaluated: a positive flag is returned if it is lower than one.

3.1.3.1.3 `canGrow` function

The *`canGrow`* function verifies if the EOA can grow to cover the query point. The query point and the function value, evaluated by direct integration, are provided as arguments.

The linear approximation of the function value is calculated using the information stored in the leaf.

Then, the approximation error is evaluated and a positive flag is returned if lower than the tolerance.

3.1.3.1.4 `growEOA` function

The *`growEOA`* function performs the algorithm of growth of the ellipsoid of accuracy of the leaf. The result is an EOA which covers the query point and the previous ellipsoid. The query point is provided as an argument of the function.

First, the distance between the query point and the center of the leaf is computed. Then, the product between lower-triangular Cholesky matrix $\boldsymbol{L}$ and that distance is evaluated.

Second, the rank-one modification algorithm permits to shrink the ellipsoid so the query point becomes on the boundary of the EOA. The QR factorization of the matrix obtained from the algorithm is evaluated in order to store the lower-triangular Cholesky matrix $\boldsymbol{L}$ which represents the grown ellipsoid.

Last, the SVD of $\boldsymbol{L}$ is computed in order to obtain the new r_{min} and r_{max} values which are the inverse of the largest and smallest singular value.

3.1.4 *binaryNode* class

The nodes of the Binary Tree are described using the *`binaryNode`* class. Every time it is necessary to create a new node a new instance of this class is created.

The purpose of this class is to allow the navigation within the Binary Tree storing the pointer to its children, which can be either leaves of nodes, and the information of the cutting plane.

The initialization is performed giving as arguments the children of the node and the parent node.

The function evaluates the cutting plane parameters through the calculation of the $\boldsymbol{v}$ vector and the scalar a using the algorithm described in Section 2.3.3.3.

3.2 *ISATLib* validation

The *ISATLib* has been validated in order to prove its reliability and accuracy. The validation has been carried out comparing the numerical results and performance of ISAT method with those of a direct integration method.

The test case consists in a set of cells, each of them represents a batch reactor. The internal composition is uniform by definition and changes only due to the chemical reactions. The cells are completely independent.

The thermochemical of the reacting mixture in a batch reactor can be characterized by the mass fraction y_i of the species, the enthalpy h and the pressure p. Moreover, the batch reactor contains also a catalytic surface where heterogeneous reactions occur, characterized by the site active coverages ϑ. Constant pressure batch reactors are assumed, so the state system is determined by:

$$\boldsymbol{\phi} = \{\boldsymbol{y}, \boldsymbol{\vartheta}, h\} \tag{3.1}$$

The evolution of the reacting state is represented by a set of ordinary differential equations, one for each element of the $\boldsymbol{\phi}$ vector:

$$\frac{d\boldsymbol{\phi}}{dt} = \boldsymbol{S}(\boldsymbol{y}, \boldsymbol{\vartheta}, h) \tag{3.2}$$

where $\boldsymbol{S}$ is the rate of change due to the chemical reactions.

ISAT and direct integration are separately used, over fixed number of time steps, to integrate the ODE system equations which govern the reacting state evolution of the system. After every time step, the approximation error is directly evaluated to verify the capability of the algorithm to control the local error.

3.2.1 Validation test case

The validation test case consists in a set of 100 cells, with different initial conditions, integrated over 500 time steps corresponding to 50,000 ISAT queries. Figure 3.9 shows a sketch showing the validation case structure.

To generate the initial conditions, a single batch reactor is integrated with an ODE solver over a fixed number of time steps, using the kinetic schemes selected for the test. The

results at each time step are collected in a file. The validation methodology randomly takes a record from the file and assigns it as initial condition of a cell until every cell is filled with a different initial condition.

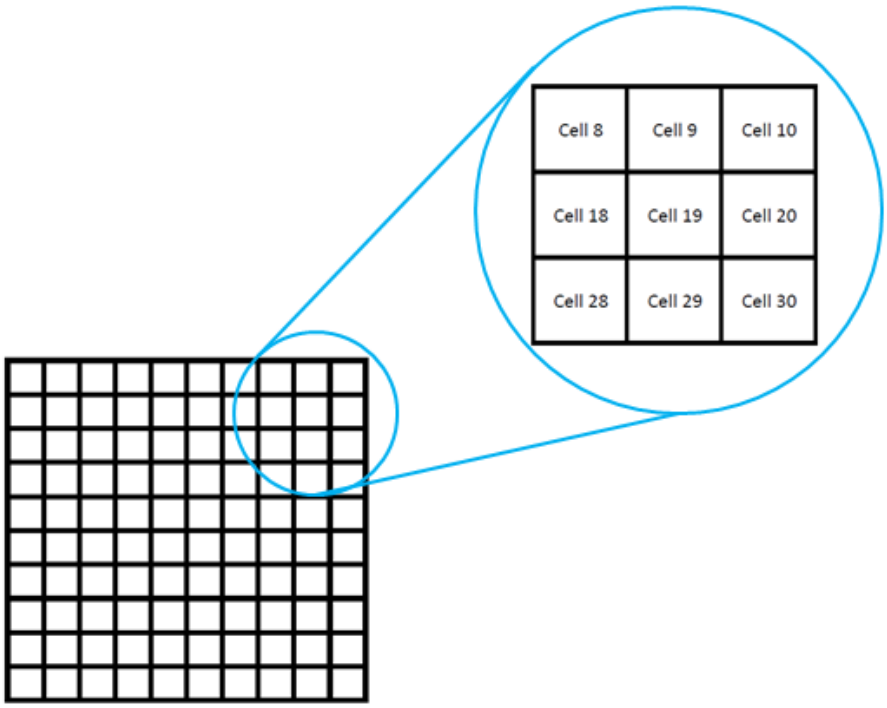


Figure 3.9 – Sketch showing the validation case

The composition vector passed to ISAT contains the species mass fraction, the site coverages and the temperature. Moreover, the elements of the vector are properly scaled to guarantee that each of them has the same magnitude order

In order to take into account the contribution of the heterogeneous catalytic reactions; each cell has one catalytic boundary surface.

3.2.2 Control of errors

The goal of the test cases is to verify the accuracy of the ISAT algorithm describing the evolution of the composition in each cell. The effect of different tolerance values is exploited in order to find a reasonable value to use in heterogeneous chemistry simulations. To this aim, each kinetic scheme is tested using three different tolerance values.

As suggested by Pope [3], a useful parameter that might describe the accuracy of the method is the global error:

$$\varepsilon_G = \frac{1}{K\,N}\sum_{k=1}^{K}\sum_{n=1}^{N}\left\|\boldsymbol{B}\left(\boldsymbol{\phi}_{\boldsymbol{ISAT}}^{(n)}(k\Delta t)-\boldsymbol{\phi}_{\boldsymbol{DI}}^{(n)}(k\Delta t)\right)\right\| \tag{3.3}$$

where N is the number of cells, K is the number of time steps, Δt is the time step value, $\boldsymbol{B}$ is the scaling error matrix.

In the subsequent Sections are presented the control error results.

3.2.3 Test simulations

The tests are performed accordingly to an increasing complexity: initially, low dimension problems are considered, afterward high-dimension tests are performed. For this purpose, the following detailed kinetics mechanism has been adopted:

a. *Hydrogen UBI* heterogeneous [24] in adiabatic condition
b. *Methane UBI* heterogeneous [24] in adiabatic condition
c. *Methane DRM19* heterogeneous and homogeneous [25] in adiabatic condition

3.2.3.1 Hydrogen UBI heterogeneous kinetic scheme

The *Hydrogen UBI* [24] heterogeneous kinetic scheme is as follows:

- 7 species in homogeneous phase
- 5 species in heterogeneous phase
- 18 surface reactions

As already mentioned, the initial conditions are generated using a direct integration of a single batch reactor. Some species evolution profiles are shown in Figure 3.10.

The initial values of the cells are randomly taken from the profile to increase the variability of the inlet composition. Figure 3.11 shows the distributions of the initial conditions of the cells for some of the species. In this way each cell has a composition different from the previous, ensuring that the initial value has no particular patterns.

The test is performed using three different tolerances – $5 \cdot 10^{-2}$, $5 \cdot 10^{-3}$, $5 \cdot 10^{-4}$ – to exploit its effect on the accuracy of the method and on the different number of ISAT events.

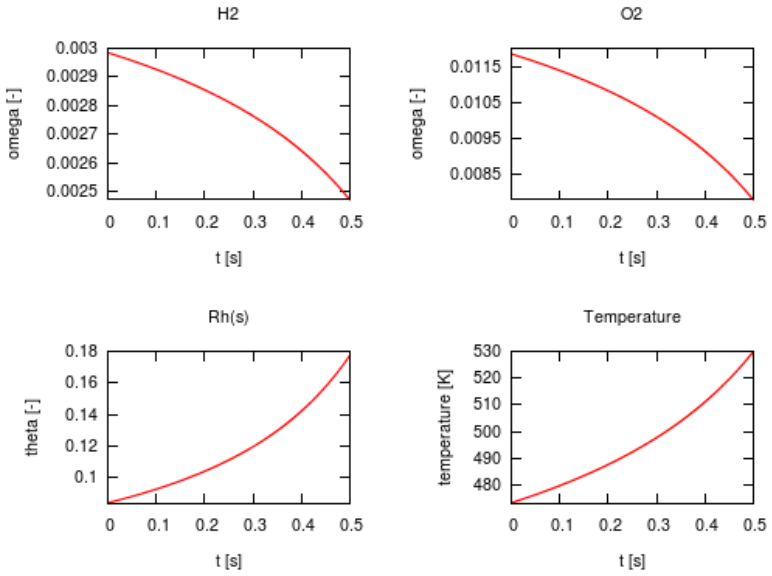


Figure 3.10 – Initial conditions – evolution profile of H2, O2, Rh(s), temperature

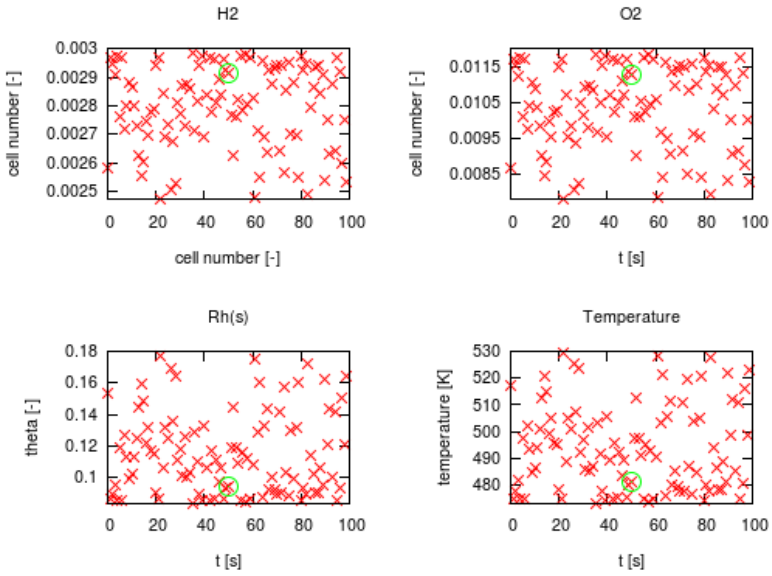


Figure 3.11 – Initial conditions – distribution of initial values, with a green circle is highlighted the initial condition of the cell used to show the simulations results – H2-UBI

Figure 3.12 and 3.13 show the simulation results of one of the cells are shown for some of the species, the agreement is good for the two lower tolerance values, but unacceptable for the largest one.

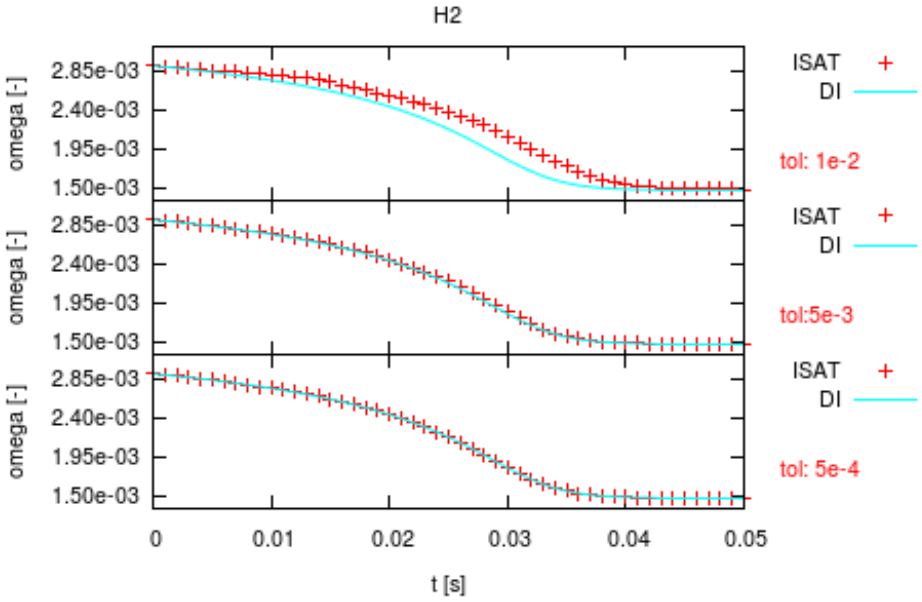


Figure 3.12 - Simulation results – gas phase – H_2-UBI

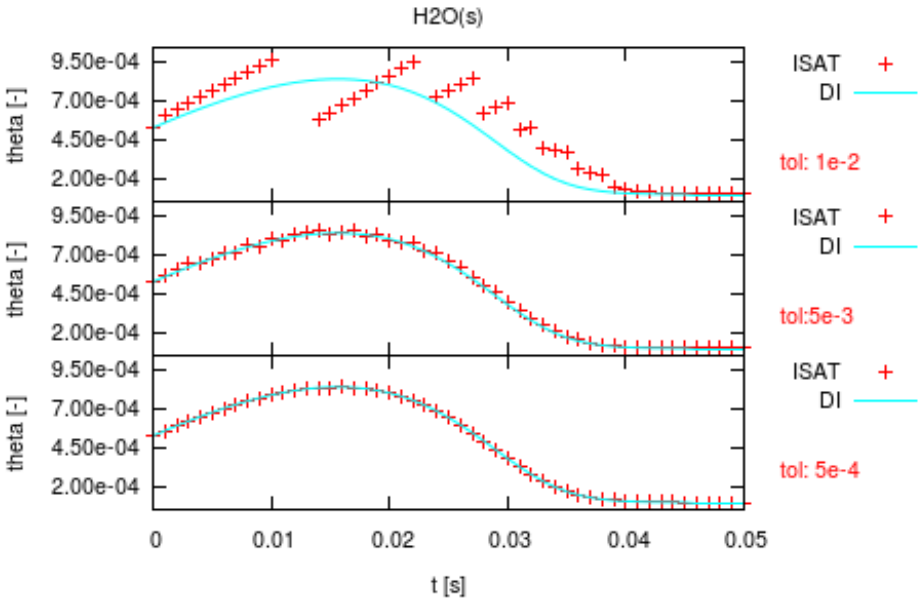


Figure 3.13 – Simulation results – adsorbed species – H_2-UBI

Figure 3.14 shows the tolerance, the average error and the maximum error are shown for each tolerance used. The average value is always lower than the tolerance query and the largest error is at most less than two times the tolerance.

The global error and the fraction of the query violating the tolerance are shown in Table 3.1. The fraction of query violating the tolerance is evaluated on the total number of events.

The error is within the tolerance for the 97 % of the query for the two lower tolerances whereas it is larger than the tolerance for the 15 % of the query in the case of the highest tolerance. Thus, a high tolerance value leads to an inaccurate approximation despite the average error value is lower than the tolerance.

Tolerance	$5 \cdot 10^{-2}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$
ε_G	$2.7 \cdot 10^{-2}$	$2.0 \cdot 10^{-3}$	$2.2 \cdot 10^{-4}$
Fraction of violations	13.9 %	2.5 %	2.8 %

Table 3.1 – Global error versus tolerance – H_2-UBI

The algorithm based on ellipsoid of accuracy is satisfactory in control approximation error, but large tolerance values determine a bad overlap between the profile obtained by ISAT and direct integration

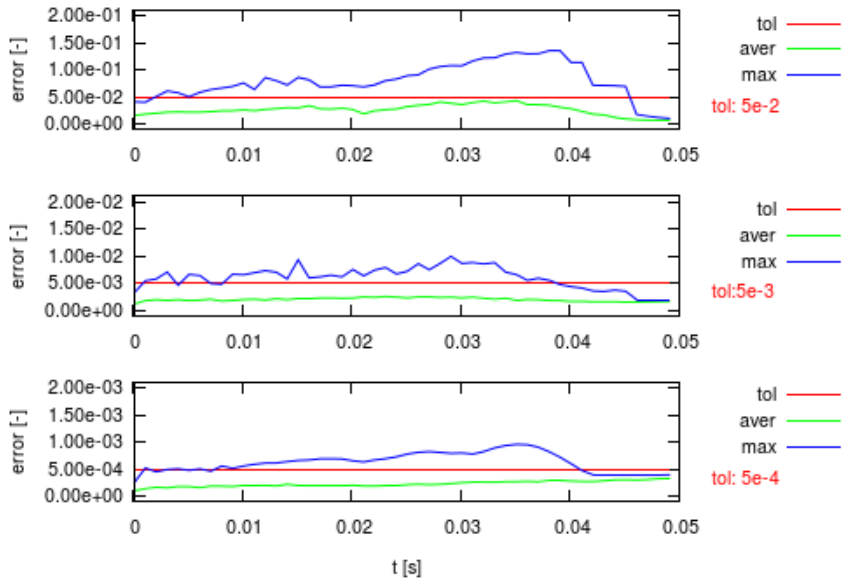


Figure 3.14 – Local error plotted against time – H_2-UBI

The ISAT library decreases the time required to complete the integration over the fixed time step, as shown in Table 3.2 showing the average clock-time required to perform each ISAT and direct integration tasks.

Average Time DI [μs]	13199		
Tolerance	$5 \cdot 10^{-2}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$
Average Time ISAT [μs]	14	29	78
Average Time Retrieve [μs]	8	14	22
Average Time Growth [μs]	2158	1877	1124
Average Time Addition [μs]	2258	1353	1234
Speed-up factor	81	46	29

Table 3.2 Average query time – H_2-UBI

The average time is evaluated as the mean value of set of time required to perform that task.

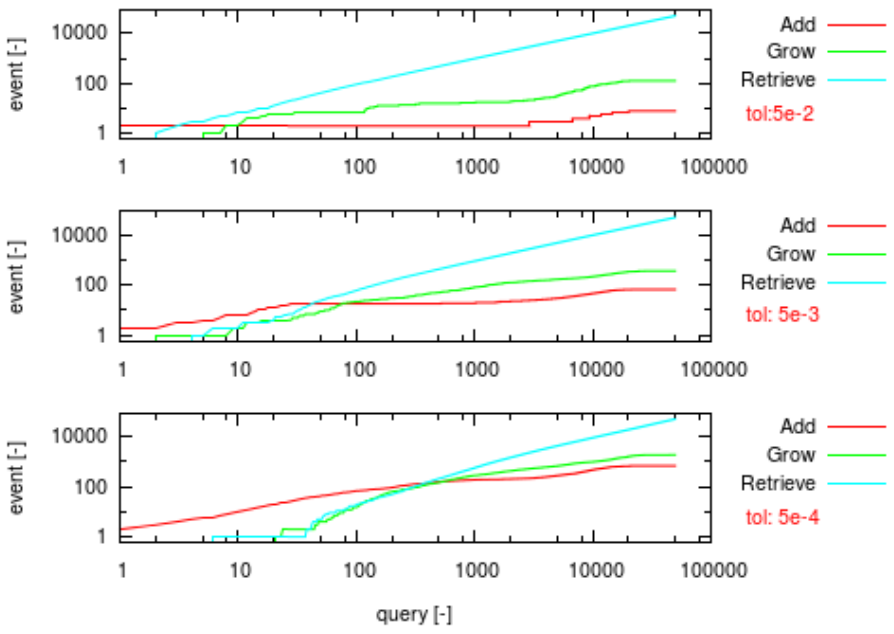


Figure 3.15 – Number of events plotted against number of query – H_2-UBI

The evolutions of the ISAT events are reported in Figure 3.15. At the beginning of the simulation, the table is empty so it is necessary to perform some additions and growths to settle the table. Then, the number of retrieves increases and quickly becomes larger

than the other types of events. After a certain number of events the growths and the additions restart due to the evolutions of the system. The ratio between the number of retrieves and the number of additions or grows is affected by the tolerance. A higher tolerance value determines a narrow region of accuracy, thus the probability of successful retrieves is lower and the number of additions and growth are larger.
The number of each different event depends on the tolerance value: the lower is the tolerance, the larger is the number of additions and growths, because it is less probable that a query is covered by one of the leaves EOA.

3.2.3.2 Methane UBI heterogeneous kinetic scheme

The *Methane UBI* [24] heterogeneous kinetic scheme is as follows:

- 16 species in homogeneous phase
- 13 species in heterogeneous phase
- 82 surface reactions

The kinetic scheme is larger and more complex than the previous one. In this way, ISAT performances are tested in a more complicated situation.
The test is performed using three different tolerances – $1 \cdot 10^{-3}$, $5 \cdot 10^{-4}$, $1 \cdot 10^{-4}$ – to exploit its effect on the accuracy of the method and on the different number of ISAT events. The initial conditions are generated as already mentioned to provide different initial value to each batch reactor.

The evolution profiles for some species are shown in Figure 3.16 and 3.17. The agreement for methane, oxygen and the other major species is good for all the tolerance values, whereas radicals and species characterized by low mass fraction value shows a good overlap between ISAT and direct integration only with the two lower tolerances.
Thus, the evolution of the species with the lower concentration value is better described when the tolerance is smaller.

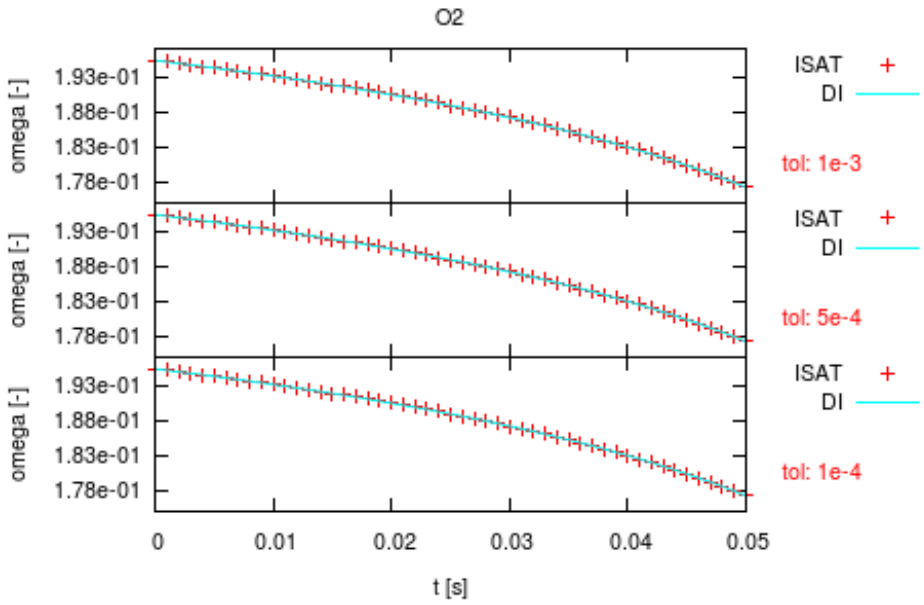


Figure 3.16 – Simulation results – gas-phase – CH_4-UBI

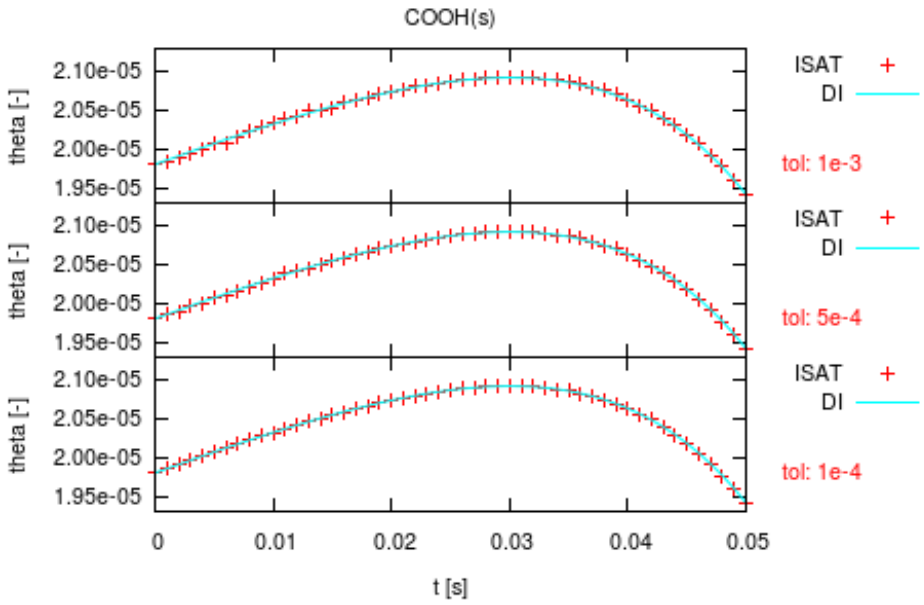


Figure 3.17 – Simulation results – adsorbed species – CH_4-UBI

In Table 3.3, the global error and the fraction of violations are reported.

Tolerance	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
ε_G	$3.4 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$	$3.3 \cdot 10^{-5}$
Fraction of violations	0.8 %	0.4 %	4.1 %

Table 3.3 – Global error versus tolerance – CH_4-UBI

The average error is always below the tolerance value highlighting the good performance, also in this case, of the ellipsoid algorithm; the maximum error is always lower than the double of the tolerance, as shown in Figure 3.18.

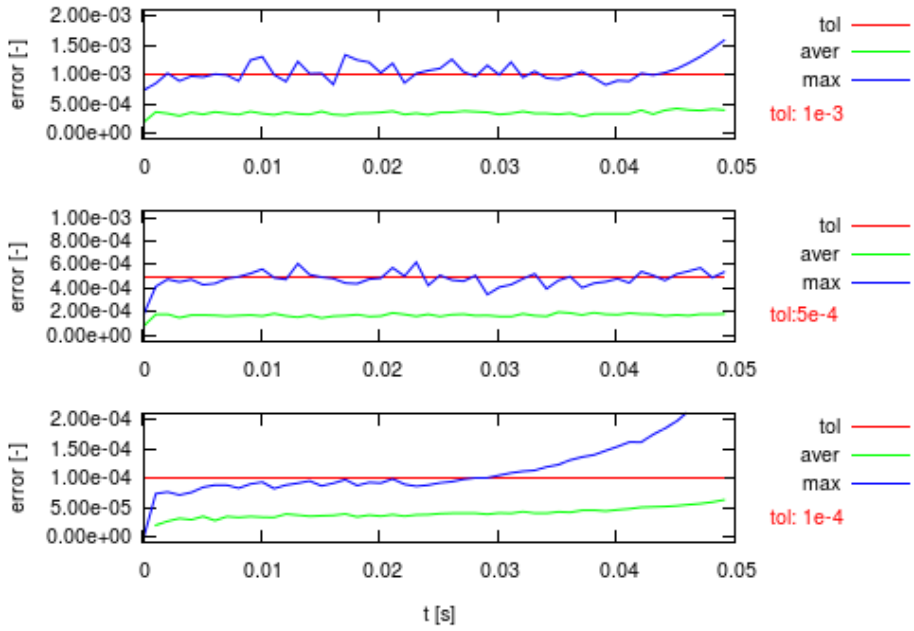


Figure 3.18 – Local error plotted against time – CH_4-UBI

The global error is lower than the tolerance for each simulation, and more than the 95 % of the query are within the fixed maximum error. The fraction of violations for the tolerance of $1 \cdot 10^{-4}$ is the larger due to the low tolerance value that lead to easier retrieve with local error slightly more than the tolerance.

In Table 3.4, the average time required to manage a query using both ISAT and direct integration is listed.

Average Time DI [μs]	5900		
Tolerance	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
Average Time ISAT [μs]	486	848	1717
Average Time Retrieve [μs]	28	30	34
Average Time Growth [μs]	13709	12891	8691
Average Time Addition [μs]	11632	11476	11540
Speed-up factor	12	7	3.4

Table 3.4 Average query time – CH_4-UBI

The number of additions grows if the tolerance decreases because the ellipsoids of accuracy are tinier. This leads to a table with a larger number of records so the time required performing a retrieve becomes larger.

As shown in Figure 3.19, initially the table performs only additions, then some growths occur and last the retrieve becomes to be effective. The number of events occurred before the retrieves start depends on the tolerance: larger values permits earlier successful retrieves due to the less accuracy required. The ratio between the number of retrieves and the other events is higher as much as the tolerance is larger.

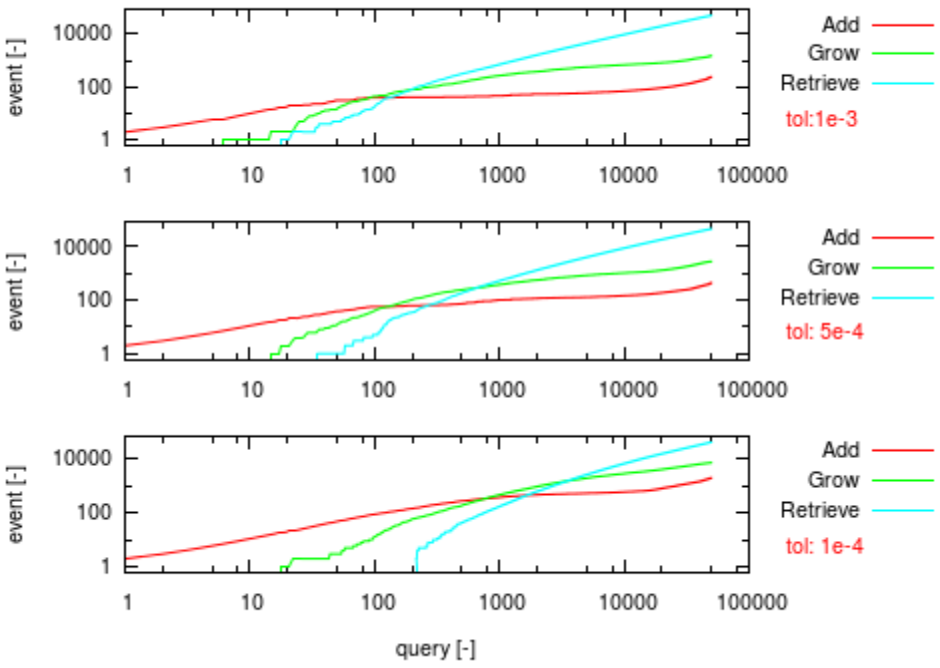


Figure 3.19 – Number of events plotted against number of query – CH_4-UBI

3.2.3.3 Methane DRM homogeneous and heterogeneous kinetic scheme

The *Methane DRM19* [25] kinetic scheme is as follows:

- 23 species in homogeneous phase
- 12 species in heterogeneous phase
- 84 homogeneous reactions
- 38 surface reactions

The composition vector consists in 36 different elements. The scheme is larger than the previous one but it is not coverage dependent.

The test is performed using three different tolerances – $5 \cdot 10^{-3}$, $1 \cdot 10^{-3}$, $5 \cdot 10^{-4}$ – to exploit its effect on the accuracy of the method and on the different number of ISAT events.

The evolution profiles are shown in Figure 3.20 for some species: the overlap between ISAT and direct integration is good.

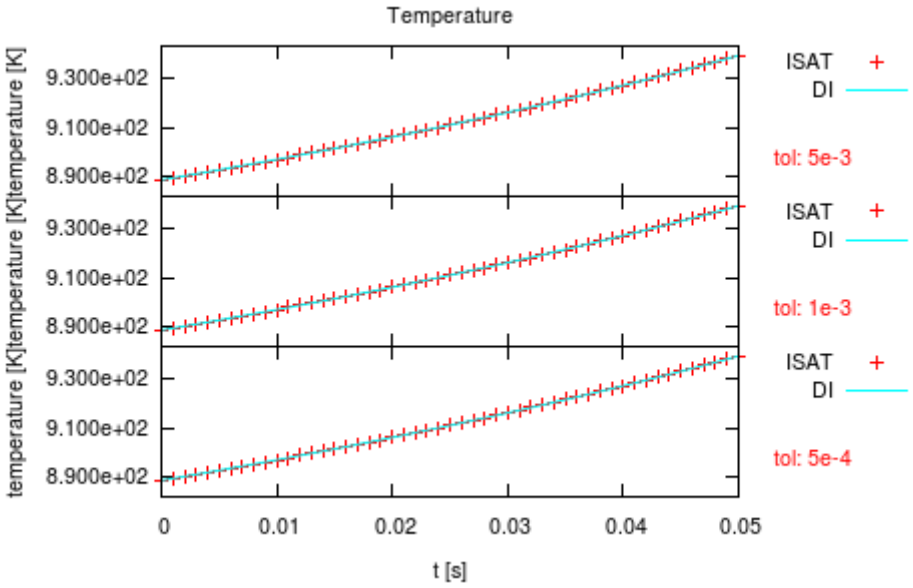


Figure 3.20 – Simulation results - CH_4-DRM19

The global error is lower than the tolerance for each simulation. The 95 % of the queries are within the tolerance and the global error is lower than the maximum value, as shown in Table 3.5.

Tolerance	$5 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$
ε_G	$1.6 \cdot 10^{-3}$	$3.2 \cdot 10^{-4}$	$1.7 \cdot 10^{-4}$
Fraction of violations	0.24 %	0.87 %	4.6 %

Table 3.5 – Global error versus tolerance – CH_4-UBI

The average error is always below the tolerance. Thus, a good error control is achieved using the ellipsoid algorithm, as it is shown in Figure 3.21.

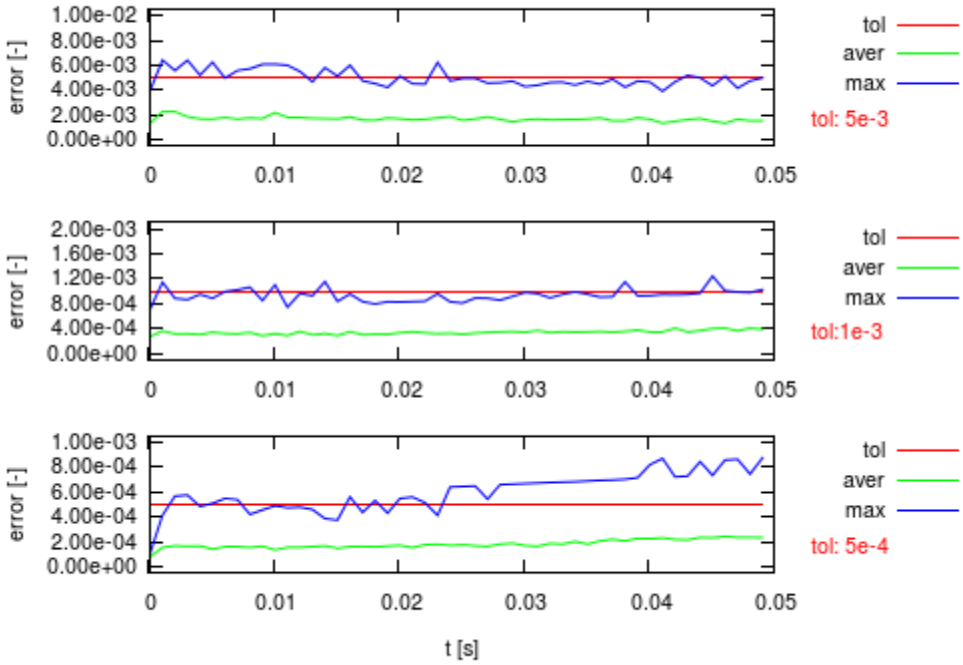


Figure 3.21 – Local error plotted against time – CH_4-DRM19

In Table 3.6, the average time required to manage a query using both ISAT and direct integration is listed.

Average Time DI [μs]	720		
Tolerance	$5 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$
Average Time ISAT [μs]	50	106	235
Average Time Retrieve [μs]	17	21	22
Average Time Growth [μs]	2274	2271	2313
Average Time Addition [μs]	3716	3529	3543
Speed-up factor	14	6.8	3

Table 3.6 Average query time – CH_4-DRM19

At last, Figure 3.22 shows the evolution of the number of different events: the lower is the tolerance, the more are the leaves that have to be added before that the retrieves begin. Moreover, the number of total retrieves depends on the tolerance and it decrease with the tolerance.

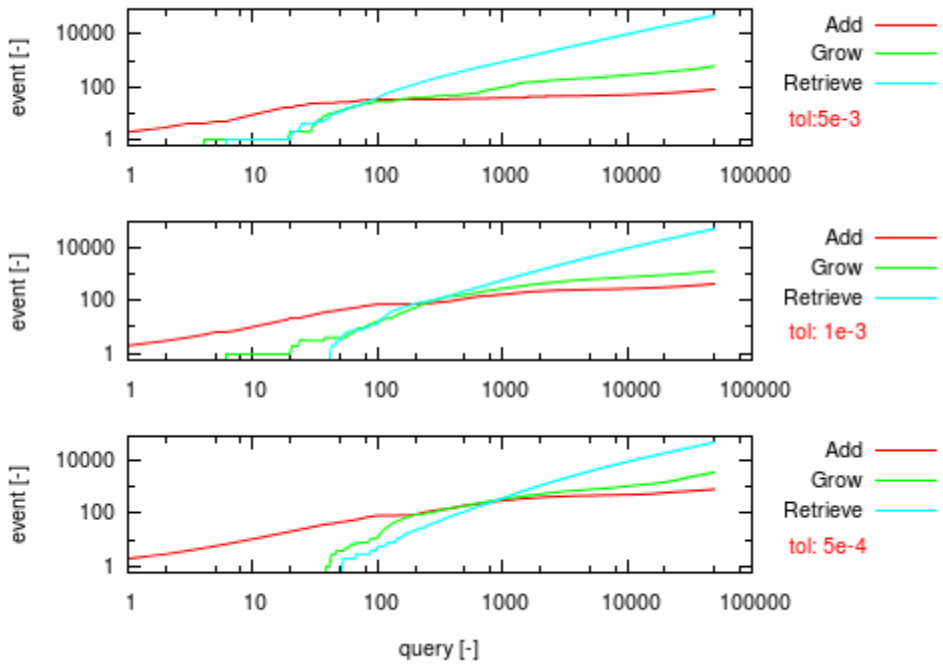


Figure 3.22 – Number of events plotted against number of query – CH_4-DRM19

3.3 Parameter sensitivity

The effect of the parameters values upon the ISAT performance has been widely investigated to obtain the set that optimize the time to manage a query.

The investigated parameters are:

- Maximum dimension of the table
- Balance parameters

The simulations are performed using the methane UBI kinetic scheme using initial conditions that leads to a high stiff ODEs system to strongly stress the performance of the table.

3.3.1 Maximum dimension of the table

In this Section, the effects of the maximum table dimension, MRU list dimension and full table treatment strategies on algorithm performance are analyzed. As already mentioned, the ISAT algorithm can adopt different strategies to clean the table: (i) the

last entry of the MRU is removed from the tree and a new leaf is added to it and (ii) the tree is cleared and the content of the MRU is added to binary tree.

Figure 3.23 shows the profiles of the average query time with respect to the maximum allowed dimension of the table using different values of the max size for the MRU. The first table cleaning strategy is adopted in this analysis.

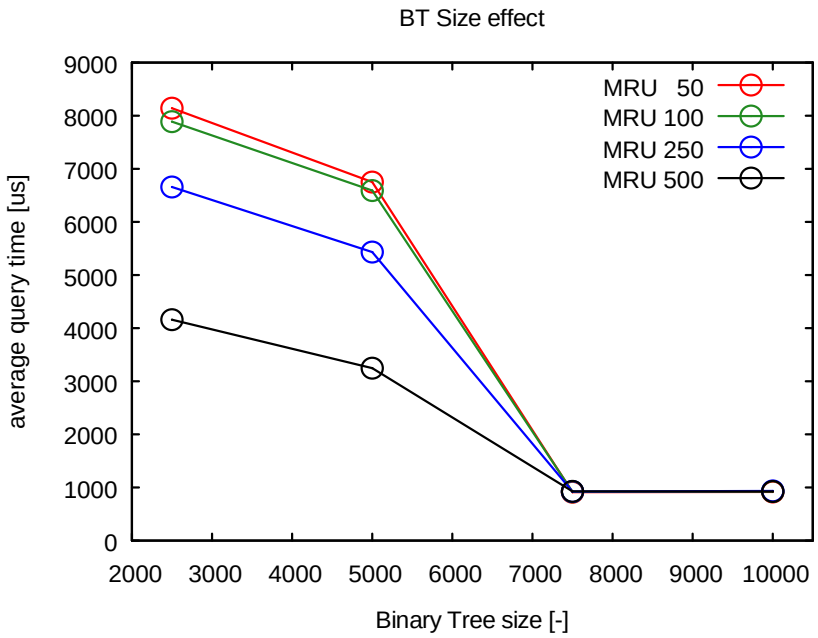


Figure 3.23 – Effect of the dimension of the table on the average query time

It is clear that the performances of the algorithm are strongly influenced by the maximum allowed dimension of the Binary Tree. A small size value leads to a large number of additions and growths because the table is not able to store a sufficient number of records to solve the queries. On the other hand, if the maximum number of entries is larger than the required number of leaves to satisfy all the queries, the performances are not affected anymore by this parameter.

It is important to choose a correct number of the dimension of the table. Basically, this value should be the maximum size that the RAM of the computer can host, in order to avoid or, at least, minimize the effect of a too tiny table on the performances. If the dimension of the table might be infinite, at a certain point the probability to have a retrieve becomes unitary obtaining the highest performance.

The dimension of the MRU list is another parameter that influences the performance of the algorithm when the size of the table is lower with respect to the total value of

necessary records. The decreasing of the MRU list dimension leads to general slowdown of the simulation, because the removed leaf is a record which can provide many retrieves in later queries. Moreover, the effect of the MRU list dimension decreases with the increasing of the binary tree dimension.

Finally, Figure 3.24 shows the effect of the different full table treatments on the algorithm performance. The analysis is carried out using the lowest value of the MRU size of Figure 3.23 and the results are compared with the average time required using the first full table treatment possibility.

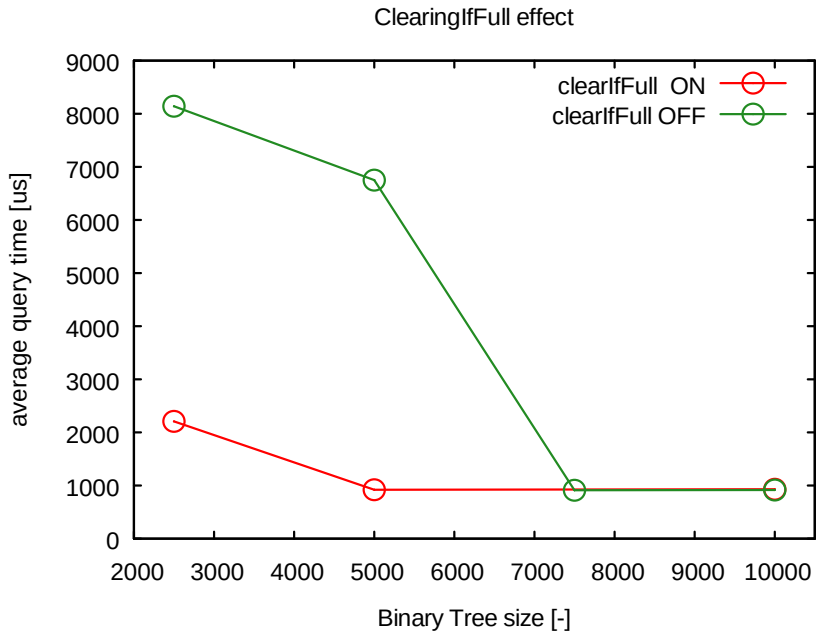


Figure 3.24 – Effect of the clearingIfFull option on the average query time

The performances of the second method (i.e. clear the tree and add the whole MRU list) are higher as the table maximum size is lower. The first method requires to remove a leaf that is deemed to be used for the last time quite far in the past and replace it with a new one. If the dimension of the MRU list is not large, the leaf selected to be removed is not the oldest leaf in the tree. In this way the old and unuseful leaves remains in the BT and a quite recent leaf is removed. The second method, instead, clears the whole table but it adds the leaves that is deemed to be more effective in the subsequent retrieve because they are the most recently used.

The results of the analysis allow to obtain some important guide lines on choosing the table dimension:

- The maximum table dimension should be set as the maximum value permitted by the RAM of the computer, in order to increase the possibility of effective retrieves.
- The size of the MRU should be set to large value which has to be lower to 500, in order to minimize the time spent in the addition and search operations inside it.
- In the case that the available RAM of the computer leads to a quite small maximum dimension is better to set the possibility of clearing the table if full.

3.3.2 Balance parameters

The effect of the balance parameters on the algorithm performance is investigated in this Section. As already mentioned, the ISAT algorithm provides four different parameters to carry out cleaning and balance of the Binary Tree. The analysis is performed only on the following main two:

- *maxHeigthCoeff*: number of times that the height of the tree has to be larger than the ideal table depth to perform the table reshape procedure;
- *maxTimeOldCoeff*: fraction of the simulation used to identify a leaf can be considered old and removed from the tree.

A set of simulations is performed using the same tolerance value to investigate the effect of these parameters upon the clock time required to manage a query. The effect of each parameter is deemed to be independent from the others, thus each analysis is performed using fixed values of the coefficients and modifying only the interested.

Figure 3.25 shows the behavior of the retrieve time with respect to the *maxHeightCoeff*: the profile shows a minimum for the average query times. The minimum value corresponds to a value of 5, which means that the height of the tree (i.e. the difference between the depth of the right and left branches) is 5 times larger than the ideal depth of the tree. A large value of the coefficient leads to larger computational time due to the unbalanced shape of the tree: if the right and left branches have a great difference in their depth the effectiveness of the search procedure is lower leading to a larger number of additions and growths.

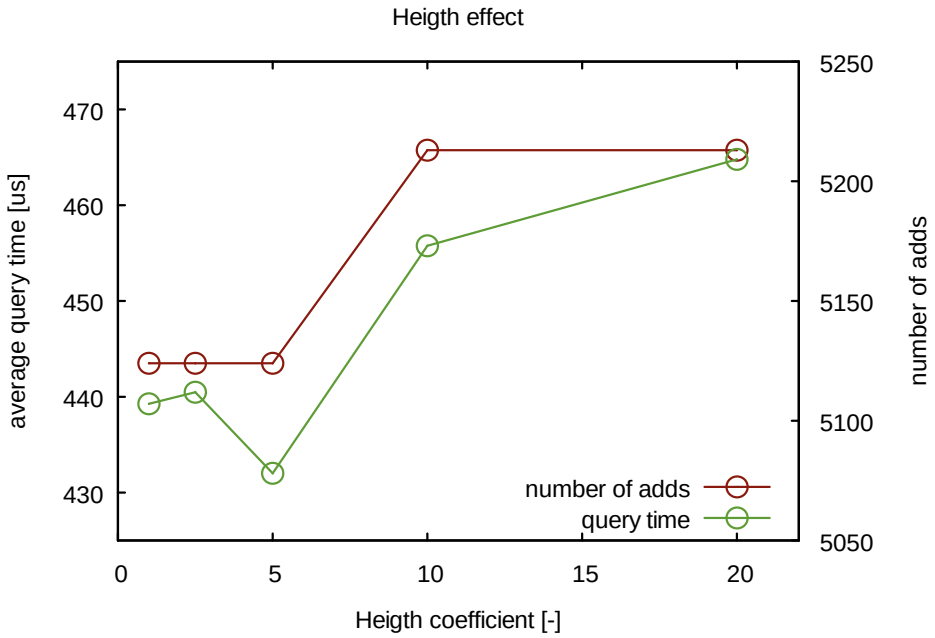


Figure 3.25 – Effect of the maxHeigthCoeff on query times

On the other hand if the coefficient is set to a too low value the slow reshape operation is performed to many times without any improvements, actually slowing down of the simulation.

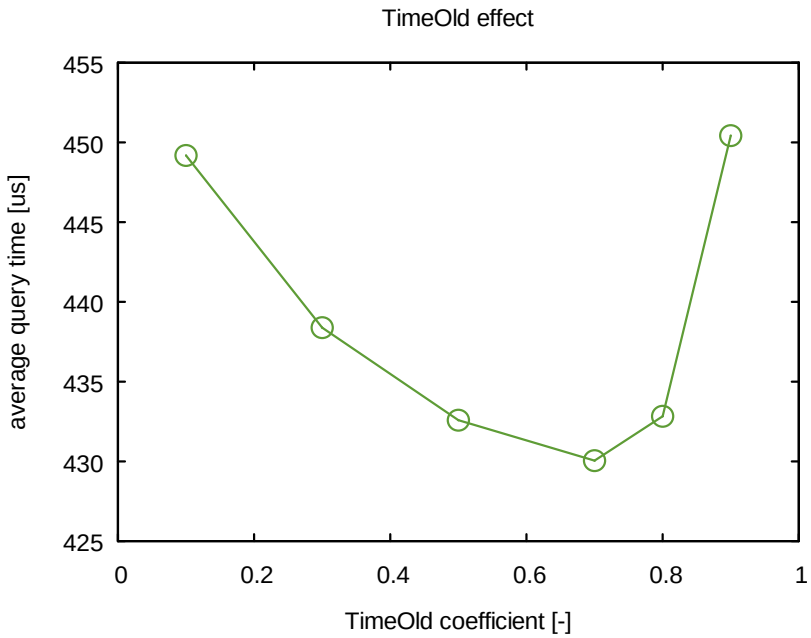


Figure 3.26 – Effect of the maxTimeOldCoeff on query times

Figure 3.26 shows the profiles of the average query time with respect to the *maxTimeOldCoeff*. The average query time presents a minimum with respect to the parameter to a value of 0.7, which means that, during each cleaning and balance procedure, the leaves used for the last time before the *maxTimeOldCoeff* multiplied per the total simulation time are removed.

The behavior is due to two different effects: first a low value determines that a lot of leaves are removed leading to the removal of useful leaves and, second, if the value is too high only a few leaves are removed and leaves that are not able to solve any further query remain in the tree.

3.4 Conclusions

The reliability of the *ISATLib* has been demonstrated: the performed simulations show a good overlap between the prediction of ISAT and direct integration. However, a correct value of the tolerance has to be set: as shown in the first test case, if its value is too high the overlap between the profiles is unacceptable despite the average error is below the fixed maximum value.

The algorithm proposed by Pope [3] always permits a good error control. A poor agreement between ISAT and direct integration is only due to a wrong tolerance value fixed by the user.

The time required to perform an adding procedure is more than the time necessary to perform a direct integration. It occurs because to perform an addition the direct integration is only the first step of the procedure followed by the evaluation of the mapping gradient and the initialization of region of accuracy. On the other hand, the time required to perform a retrieve is many magnitude order lower and it is modestly affected by the dimension of the composition vector.

The ISAT simulation, at the beginning, is slower than the one performed using direct integration due to the addition and growth, but it becomes faster when the table is settled and the retrieves begin to occur frequently.

The number of retrieves controls the speed-up of the simulations because each retrieve is at least one order of magnitude faster than a direct integration. A very low tolerance

value determines a large number of addition and growth that are slower than a direct integration, so it is possible that the ISAT simulation results slower than the same performed with direct integration. Thus, it is necessary to find out a correct value for the tolerance that permits to have both reliable results and speed-up of the simulation.

The best speed-up performance are observed using the smallest kinetic scheme, i.e. *Hydrogen UBI* [24]. The performances enlarging the dimension of the kinetic schemes become worse due to the higher number of additions that are required for that schemes. The number of additions is larger with these schemes due to the wider composition space that determines a lower probability of successful retrieves. Moreover, the algorithm evaluate the approximation error as the norm-2 of the difference between the linear approximation and the function value: a larger schemes determines a more probable large error value.

The analysis carried out on the most important balance parameters shows the importance of their good tuning. The performance in respect to the time required to manage a query (i.e. the clock time necessary to perform all the operations needed to return the function value given a query to *ISATLib*) can be reduced up to 5 %. The reasons of these improvements are:

- if the tree is balanced the retrieve procedure is more effective requiring a lower number of addition ad growing to solve the queries
- if the tree dimension is kept to a low value cleaning from it the old leaves, the retrieve procedure becomes faster due to the lower number of comparison required to reach a terminal leaf

The values for the balance parameters obtained in this analysis are used in the simulation performed in the next Section of this work.

4 *ISATLib* implementation in catalyticFOAM

In this chapter the catalyticFOAM *solvers are briefly described, then the* ISATLib *implementation in the solvers framework is illustrated and some simple validation cases with increasing kinetic schemes complexity are presented*

4.1 Overview

The catalyticFOAM solvers were developed in previous Thesis [1, 2] to investigate the interactions between fluid dynamics and chemistry in heterogeneous catalytic reactors through the "first principle approach". These solvers were developed using the structure and tools of OpenFOAM® (Open Field Operation and Manipulation) Toolbox, which is a free, open source CFD software package applied to solve several different systems, from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics [4]. In particular, two different catalyticFOAM solver versions were implemented and tested both on simple geometries and complex cases of industrial interest.

The first solver, named catalyticFOAM, is a CFD code able to simulate catalytic reactive systems in arbitrary complex multi-dimensional geometries using a detailed microkinetic scheme for the surface reactivity. The modelling of the solid catalyst is neglected, considering that the reactions occur only on the catalytic surface. This means that catalyticFOAM solver is able to describe the heterogeneous reactive system under chemical or external mass transfer regime, but not under internal diffusion regime.

For this reason, a second solver was developed to introduce the description of the inner part of the catalyst. This solver, named `catalyticFOAM-multiRegion`, allows to introduce a detailed description of the solid catalyst by means of a multi-region structure: the dynamic and the homogeneous reactions in fluid phase are described with the same approach used in `catalyticFOAM`, while the dynamic solution of the catalyst domain takes into account the transport phenomena within the solid phase and the heterogeneous reactions, which occurs not anymore only on the catalyst surface but in the whole solid volume.

The solvers implement the operator splitting technique, commonly used for the numerical computation of reactive flows. This numerical algorithm consists of splitting the equations into two different groups, which are integrated separately and sequentially. This method is particularly advantageous when the terms of the equations have a different rank of linearity and stiffness, because it allows to use the most efficient numerical technique to solve each term of the equations. The results of the integration at each sub-step are combined to approximate the final solution with high accuracy.

Considering the generic transport equation:

$$\frac{d\boldsymbol{\phi}}{dt} = \boldsymbol{M}(\boldsymbol{\phi}, t) + \boldsymbol{S}(\boldsymbol{\phi}) \tag{4.1}$$

where $\boldsymbol{\phi}$ is the entire set of the species composition and the temperature; $\boldsymbol{M}$ denotes the transport terms due to diffusion and convection, whereas $\boldsymbol{S}$ denotes the source term due to chemical reactions. The $\boldsymbol{M}$ term presents a low stiffness and a quasi-linearity; $\boldsymbol{S}$ is highly non-linear and stiff. The stiffness of the source term depends on the very broad timescales range of the reacting phenomena and requires the use of robust and time-consuming numerical techniques to ensure stability and accuracy of the solutions.

The splitting scheme applied in the solvers is based on staggered time step approach, as widely explained in previous Thesis [1].

This method separates the terms of the equation in two different systems of equations, a PDEs system given by the diffusion and convection terms and an ODEs system obtained from the reactive term.

$$\begin{cases} \frac{d\boldsymbol{\phi}_1}{dt} = \boldsymbol{M}(\boldsymbol{\phi}_1, t) \\ \frac{d\boldsymbol{\phi}_2}{dt} = \boldsymbol{S}(\boldsymbol{\phi}_2) \end{cases} \tag{4.2}$$

At the beginning of the time step, the composition $\boldsymbol{\phi}$ of a certain cell, i.e. the vector containing the mass fractions and the temperature, is the initial value of the PDEs system describing the transport and diffusion step – first equation of (4.2). The system is solved returning the value of $\boldsymbol{\phi}_1$. This value is used as the initial conditions of the chemical step, represented by the second equation of system (4.2). The solution of the chemical step provides $\boldsymbol{\phi}_2$ which is stored in the cell and represents the composition after the whole time step integration. This procedure is applied to all the cells of the computational domain at every time step.

The Jacobian matrix associated to the entire PDEs system is unstructured and sparse. Its dimension is $N_{cells} \times NS$, where N_{cells} is the total number of computational cells and NS is the total number of species and temperature. The Jacobian matrix associated to the reactive step becomes a diagonal blocks matrix. Indeed, the rate of production in each cell depends only on the conditions of the cell itself. This implies that the PDEs system can be turned into a group of decoupled Ordinary Differential Equations (ODEs) systems. The resulting numerical problem is constituted by a system of weakly non-linear PDEs and a group of non-linear ODEs systems.

The accuracy and the computational time of operator splitting technique depend on the time step Δt, which is defined on the basis of the Courant Number.

The resulting numerical problem requires high computational efforts:

- The dimension of the system is proportional to the number of species involved, so the level of detail of the kinetic scheme influences the required time
- The geometric domain has to be discretized and the resulting number of cells is proportional to the dimension of the problem
- The ODEs system representing the reaction step is very stiff and strongly non-linear

The solution of the transport and diffusion terms is performed without a large computational effort.

The chemical step is represented, as mentioned above, by the solution of a group of decoupled, independent and autonomous ODEs systems. Moreover, they are large and extremely stiff, making the task of solving these equations computationally expensive. To solve this problem, the evaluation of a multidimensional function is required, describing the mass fraction composition and temperature in each cell of the computational domain, through the integration of the associated ODEs system. Thus, the chemical step has the characteristics that allow the application of the ISAT algorithm. Moreover, the stiffness of these systems determines a large computational cost due to the *implicit* method that has to be used to solve the ODEs systems. For these reasons the solution of the chemical step is an ideal candidate for a successful ISAT implementation. The *in situ* adaptive tabulation algorithm has shown a significant capability in improving the time required to carry out the chemistry calculation of complex turbulent reactive flows, also described by a stiff ODEs system, up to a factor 1000 [3].

For these reasons, ISAT has been implemented in the `catalyticFOAM` framework in order to decrease the time required to solve the chemical step, obtaining a speed-up of the simulation without losing stability and accuracy of the solution. The benefits should be mainly two:

- reducing the time required to perform a simulation
- making possible to solve very large and complex problems in a smaller time

The algorithm has been implemented within `catalyticFOAM` framework with the purpose to improve the speed of the simulation both in the case of homogeneous and heterogeneous reactions. As mentioned before, many ISAT examples of homogeneous chemistry are available in literature, whereas the algorithm is not deeply exploited in heterogeneous chemistry so far [6].

4.2 Implementation in `catalyticFOAM`

In previous Thesis [1], it has been demonstrated that the most reliable solving procedure is the one proposed by Sportisse [26], shown in Figure 4.1. The transport equations of

species and energy are solved first providing the values of temperature and mass fractions. They are used as initial conditions for the solution of the source terms of the mass and energy to the final values of temperature and mass fractions. Finally, the momentum equation is solved by means the PISO algorithm [27] to obtain the velocity and pressure fields.

The ISAT algorithm is implemented in the second step of the process to reduce the time required to solve the chemistry.

Two different ISAT instances have to be created in order to manage the different dimensions of the ODEs system describing reaction steps in homogeneous and heterogeneous cells. The dimension of the problem concerning the homogeneous cells is equal to the number of gas phase species and temperature, whereas in the case of catalytic cells also the site coverage has to be taken into account to define the dimension of the numerical problem.

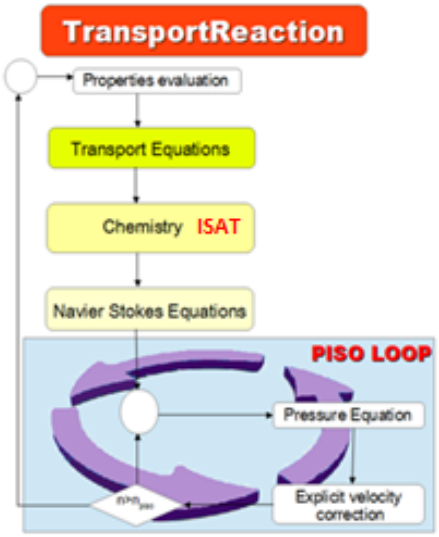


Figure 4.1 – Structure of the operator splitting method

The predictor values might be scaled before being passed to *ISATLib*; in particular, it is necessary to scale the temperature to obtain variations in the order of unity.

In Figure 4.2 the interconnections between ISAT and the solver are showed: the library is initialized providing the tolerance for each table and the `scalingError` matrix, the significance of these variables have been explained in Section 2.1.3.

For each time step of the simulation, the ISAT library receives the scaled predictor values as a query; if a leaf covers the query, the linear approximation based on the information stored in the record is returned. On the other case the ODE solver, already present in `catalyticFOAM`, solves the system of equations and provides the function values, thus the library tries to perform a growth; if unsuccessful, the mapping gradient is evaluated and a new leaf is added to the tree.

The linear approximation of the query returned by the *`ISATLib`* in the case of successful retrieve may presents some negative values. The negative values occur when some elements of the composition vector has a value near zero. To explain this phenomenon it is easier to consider a one-dimensional case: in the linear approximation, the function is approximated with a straight line tangent to the function in the data stored in the leaf. If the species presents a very low concentration but, at the same time, a high reactivity, the tangent slope is steep. This can determine an interpolation to a negative value even if the data point and the query point are close.

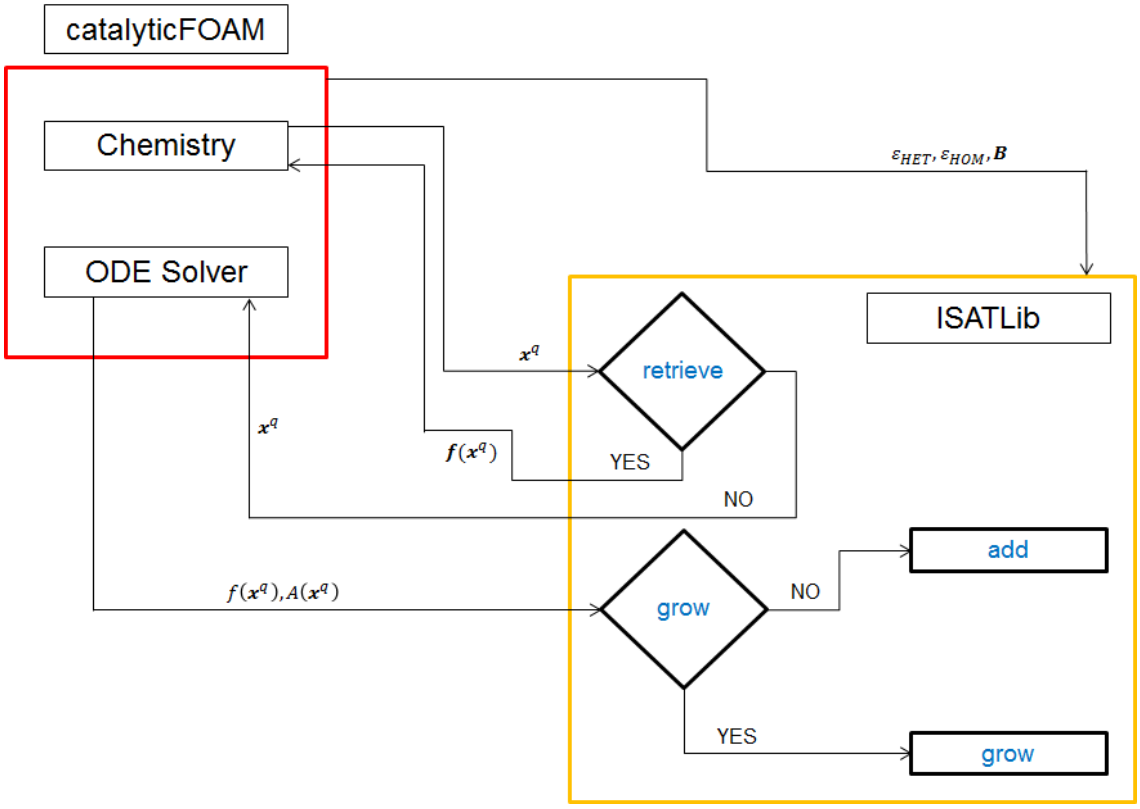


Figure 4.2 – Interconnection between `catalyticFOAM` and *`ISATLib`*

To solve this problem the vector returned by *ISATLib* is scanned and the negative values are set to zero. This is acceptable because the negative values are always very small, i.e. the absolute value is lower than 10^{-12}. Finally the composition vector, before being stored in the cells, is normalized in order to obtain unitary sum of mass fractions.

Every time a further addition is needed, the mapping gradient, described in Section 2.1.3, has to be evaluated. It is necessary to evaluate the Jacobian matrix of the ODEs system, due to the high complexity of the microkinetic schemes. A function able to evaluate the numerical Jacobian of the system has been implemented in `catalyticFOAM` framework.

The solver might operate either using a fixed or variable time step: the variable time step allows to fulfill the upper limit for the maximum Courant number, used to ensure stability to the simulation. The ISAT version has been tested in both situations obtaining reliable and accurate results, thus the variable time step method has been used for the simulations.

The `catalyticFOAM` has been developed, also, in a parallel version: the computational domain is split in a certain number of subdomains and allocated to separate processors for solution.

The parallel running uses the public domain *openMPI* implementation of the standard message passing interface (MPI).

The ISAT library has been implemented, also, in this version of the solver. The *openMPI* implementation creates a different instance of the solver for each subdomain. Each solver initializes its own ISAT instance and communications between the different tables are avoided. The performance, in this case, might be lower than the one achieved using an unique table for the whole system, after all a well-made decomposition of the domain, leading to a quite equal number of catalytic cells per processor, can lead to good fastening results.

Moreover, the absence of communications between the processors simplifies the Binary Tree management, because each tree interfaces only its domain as occurs in serial situations.

4.2.1 Case study: hydrogen combustion in adiabatic conditions over rhodium

The first test case used to validate the ISAT algorithm is the hydrogen oxidation over rhodium catalyst in a short-contact time annular reactor, as shown in Figure 4.3. The reactor consists of a catalyst-coated alumina tube coaxially inserted in a quartz-tube.

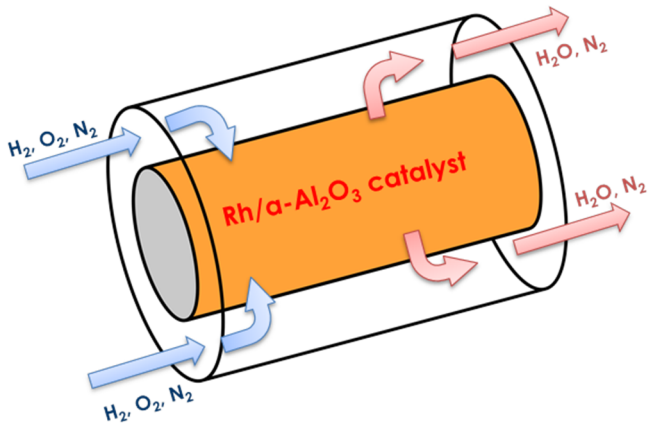


Figure 4.3 – Sketch of the annular reactor

The geometrical domain is axial symmetric and thus, it is possible to simplify its description considering a 2D system, i.e. one slice of the reactor. The mesh is a slice of a cylinder with a width of 5°.

The computational grid is refined with a specific grading, allowing a more accurate description of certain areas of the system. The cells close to the catalytic surface are interested by strong normal gradients and they require a high refining near the catalytic wall in radial direction. Moreover, a grading is introduced in the axial direction to provide a detailed description of the strong reactants consumption at reactor inlet, as shown in Figure 4.4.

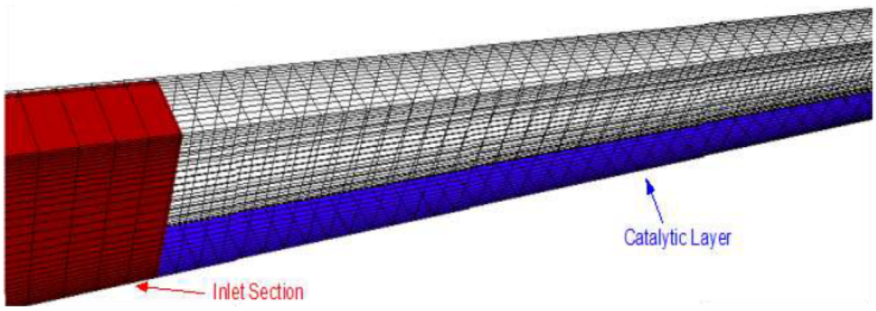


Figure 4.4 – Sketch of the annular reactor

A Courant number of 0.05 is adopted to estimate the time step of the temporal discretization and the homogeneous reactions are neglected due to the low temperature.

The mesh dimensions are listed in Table 4.1:

Catalytic cells	50
Total cells	1500

Table 4.1 – Mesh number of cells – H_2-UBI

The UBI microkinetic mechanism of hydrogen oxidation over rhodium catalyst, developed by Maestri et al. [24], is used to characterize the reactivity of the system. This microkinetic mechanism consist in 7 gas species, 5 solid species involved in 18 heterogeneous reactions.

The feed composition and the operative conditions are listed in Table 4.2:

OPERATING CONDITIONS

H_2 mass fraction	0.0031	**Inlet temperature**	473.15 K
O_2 mass fraction	0.0118	**Outlet Pressure**	1.01 bar
N_2 mass fraction	0.9851	**Flow velocity**	0.171 m/s

Table 4.2 – Operating conditions – hydrogen combustion – annular reactor

Several simulations are carried out with different tolerance values in adiabatic conditions and atmospheric pressure to exploit the influence of the tolerance on the simulation accuracy. The capability of the ISAT algorithm to control the local error has been demonstrated in Section 2.2, thus the aim of these tests is to find out an acceptable tolerance value to ensure accurate results and simulation speed-up. The results of these simulation are compared with the ones obtained by a direct integration simulation, used as a reference value.

The cup-mix, i.e. average composition computed as integral surface weighted on the mass flow rate, evolution profiles of hydrogen, oxygen and water are shown in Figure 4.5 for three different tolerance values – $8 \cdot 10^{-4}$, $5 \cdot 10^{-4}$, $1 \cdot 10^{-4}$ – and for three different time steps: one at the beginning of the simulation, one during the transitory and the last at the steady state.

The results of the simulations show a good agreement between ISAT and direct integration; only the larger tolerance leads to profiles which do not overlap the direct integration prediction. In particular, the hydrogen consumption is overestimated using the largest tolerance.

At the beginning of the simulation the overlap is quite good for each tolerance due to the high number of additions and growths, whereas the steady state solution is strongly affected by the tolerance value.

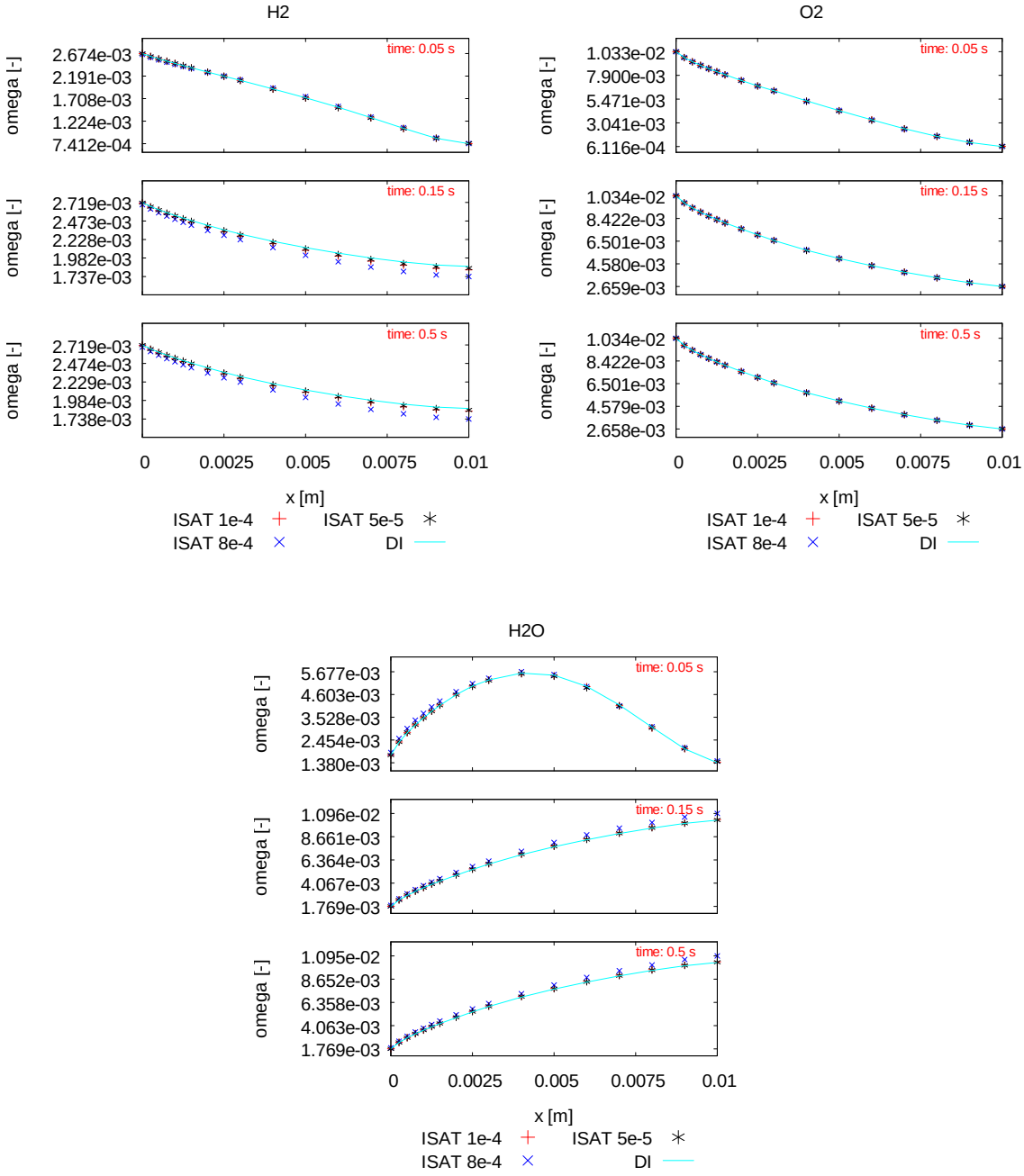


Figure 4.5 – Evolution profile of gas-phase main species – H_2-UBI

Figure 4.6 shows the axial profile for two site species with the different tolerance values. The agreement between ISAT and direct integration is good.

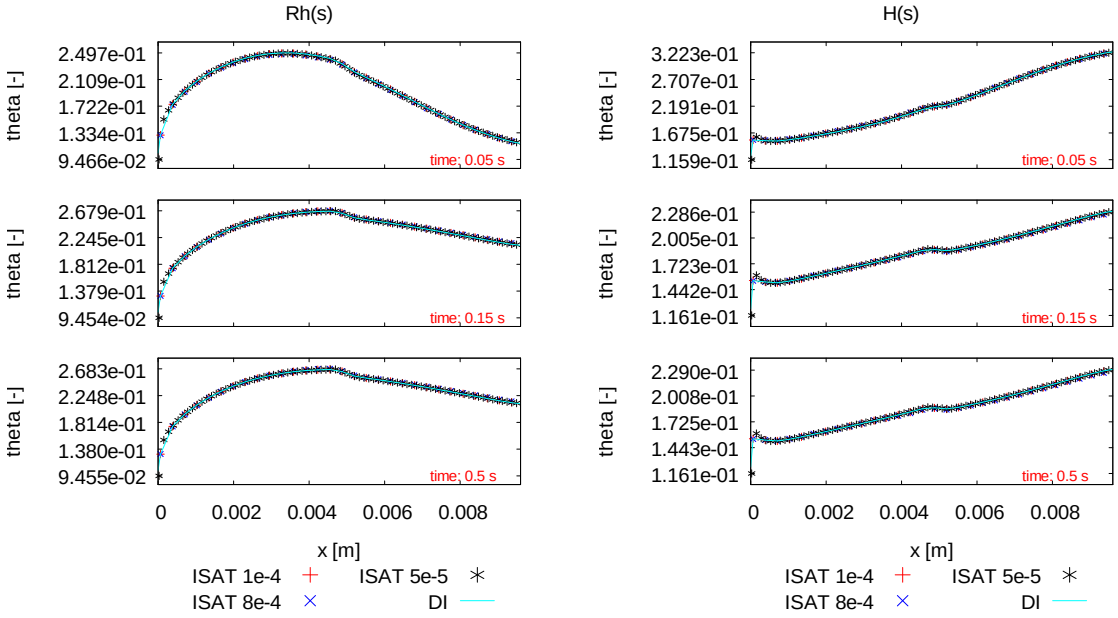


Figure 4.6 – Evolution profile of site coverage – H_2-UBI

The speed-up performances have been investigated evaluating the time required to perform a single time step and to carry out the whole simulation.

The computational time required to solve a single time step consists of two main tasks accordingly with the splitting method: the transport and the chemistry. The ratio between the time required to perform a time step using direct integration and ISAT is named speed-up factor:

$$SF = \frac{wall\ clock\ time\ per\ time\ step\ DI}{wall\ clock\ time\ per\ time\ step\ ISAT} \tag{4.3}$$

The time required to perform the former is quite independent from the latter which means that the speed-up of the simulation is due to the ISAT implementation on the chemistry step.

Reducing the computational time of the chemistry term means a significantly improvement of the computational time to carry out the whole simulation (see Table 4.3). As reported in Section 3.2.3, the speed-up decreases reducing the tolerance, moving from 4.5 to 2.7 for the highest and the lowest tolerances, respectively.

Simulation Time DI [s]	13199		
Tolerance	$8 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$5 \cdot 10^{-5}$
Simulation Time ISAT [s]	2941	3991	4809
Overall Speed-up factor	4.5	3.3	2.7

Table 4.3 – Simulation Time and Simulation speed-up factor – H_2-UBI

Figure 4.7 shows the speed-up factors and a unitary reference value for a time step every 500: at the beginning of the simulation, ISAT is slower than the direct integration due to the large number of additions and growths necessary to settle the record table. Then, the speed-up factor increases due to the higher number of retrieves, achieving a value of 4.5-5 during the final steady state.

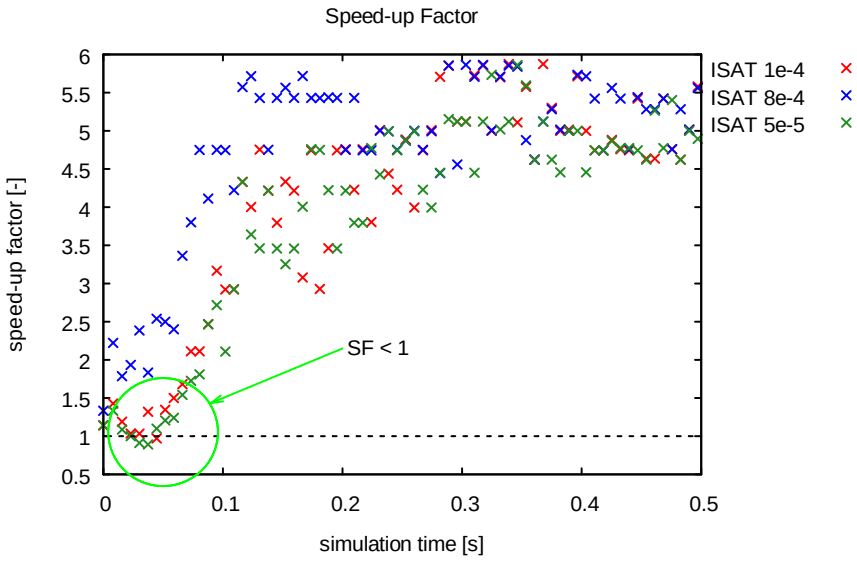


Figure 4.7 – Speed-up factor – H_2-UBI

During the initial phase, the ISAT simulation might become slower than the direct one, as shown in the light green circle in Figure 4.7.

The evolution profiles of the events are shown in Figure 4.8. At the beginning, the reactants have not reached the catalytic layer, thus only a few additions are necessary and the simulation proceeds using a large number of retrieves. Then, the chemistry becomes stronger and a high number of additions and growths is performed to settle the table.

The number of growths and additions depends on the tolerance value and in the case of the lowest one they could become larger than the number of retrieves.

After a certain number of events the table is sufficiently set up to fulfil every query, so the number of retrieves increases whereas the number of additions and growths becomes constant. The highest speed-up is achieved using the largest tolerance but it can lead to inaccurate simulation results.

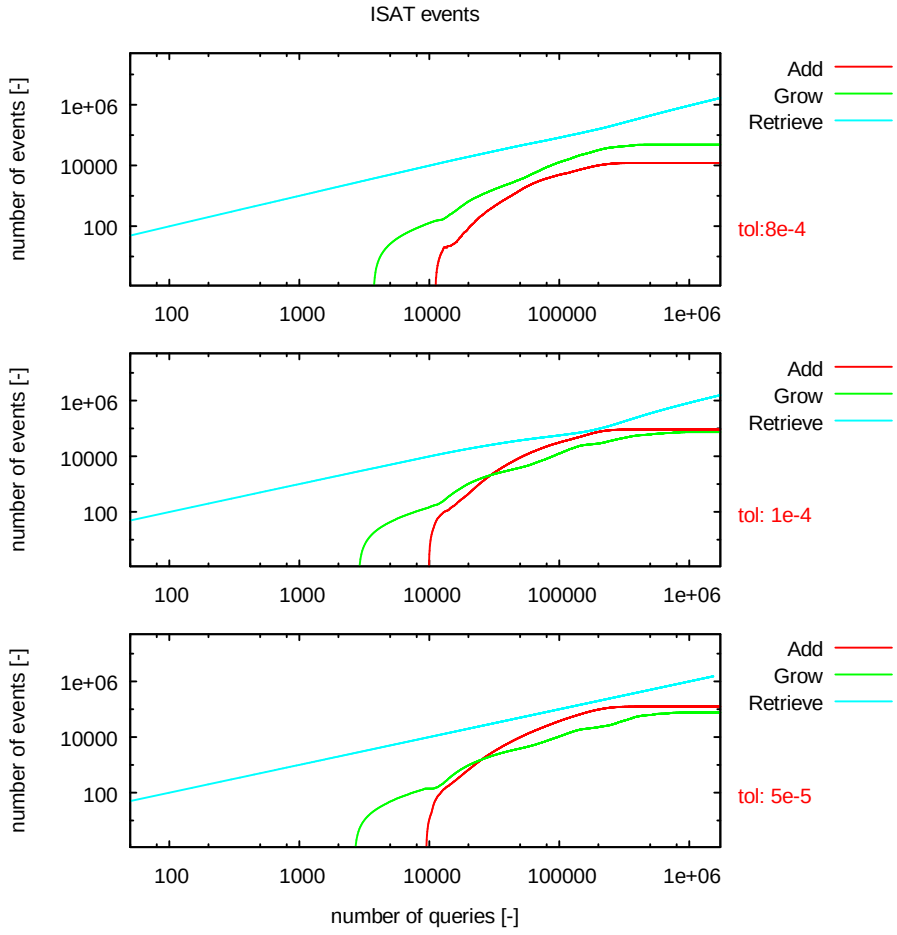


Figure 4.8 – Number of events vs number of queries – H_2-UBI

4.2.2 Case study: methane partial oxidation in adiabatic conditions

In order to stress the capability and suitability of the *ISATLib*, a second test case with a more complex kinetic mechanism is simulated in the same geometry described in Section 4.2.1. The partial oxidation of methane is investigated using a UBI microkinetic

mechanism [24], which consists of 16 gas species, 13 solid species involved in 80 heterogeneous reactions.

Several simulations have been carried out with different tolerance values in adiabatic conditions in order to exploit the effect of the tolerance on the overall accuracy and speed-up.

The feed composition and the operative conditions are listed in Table 4.4:

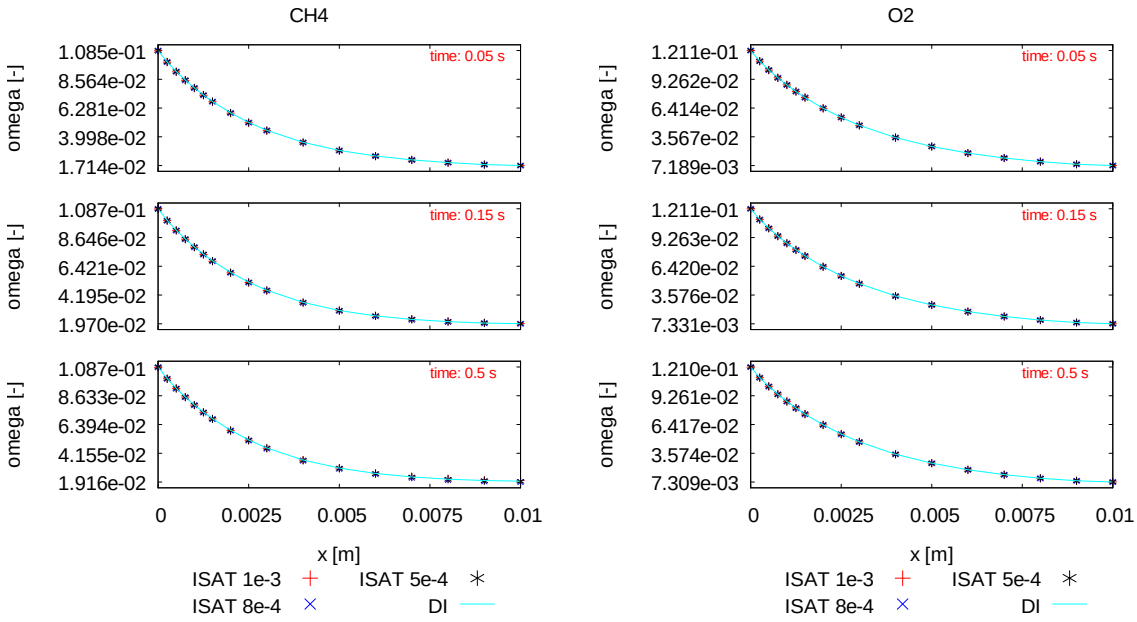
OPERATING CONDITIONS

CH_4 mass fraction	0.1565	**Inlet temperature**	873.15 K
O_2 mass fraction	0.1753	**Outlet Pressure**	1.01 bar
N_2 mass fraction	0.6682	**Flow velocity**	0.171 m/s

Table 4.4 – Operating conditions – methane partial oxidation – annular reactor

The simulations are performed using three different tolerance values – $1 \cdot 10^{-3}$, $8 \cdot 10^{-4}$, $5 \cdot 10^{-4}$ – and the evolution profiles are shown at three different time steps.

The overlap between the profiles for the gas species is good for each tolerance value, as shown in Figure 4.9. The agreement between ISAT and direct integration is good at every time step.



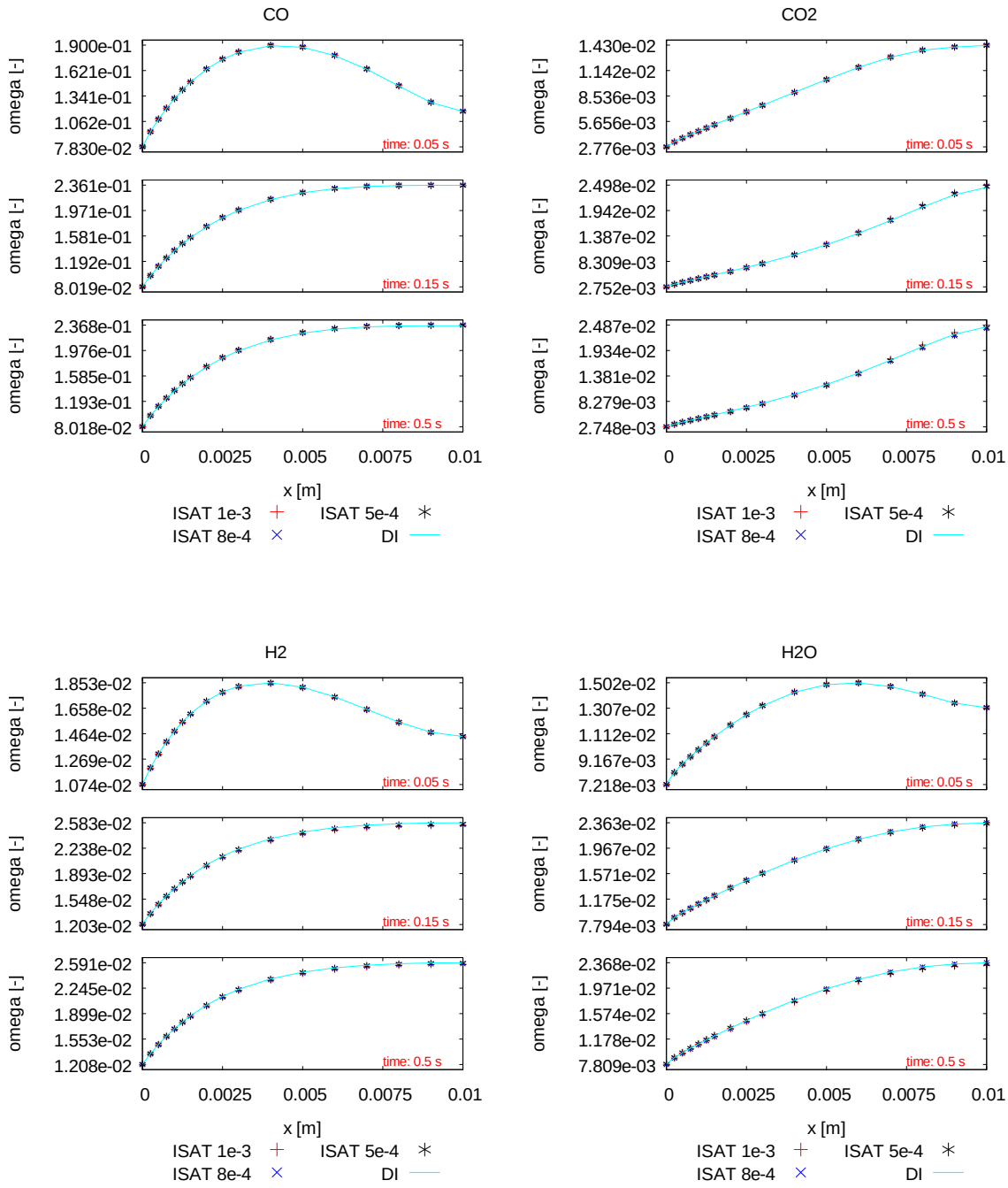


Figure 4.9 – Evolution profile of gas-phase main species – CH_4-UBI

Figure 4.10 shows the evolution profile of the sites coverage for adsorbed carbon monoxide and adsorbed hydrogen. The agreement is good for the lowest tolerance, whereas a not perfect overlapping is experienced for the two larger ones, especially for the adsorbed hydrogen.

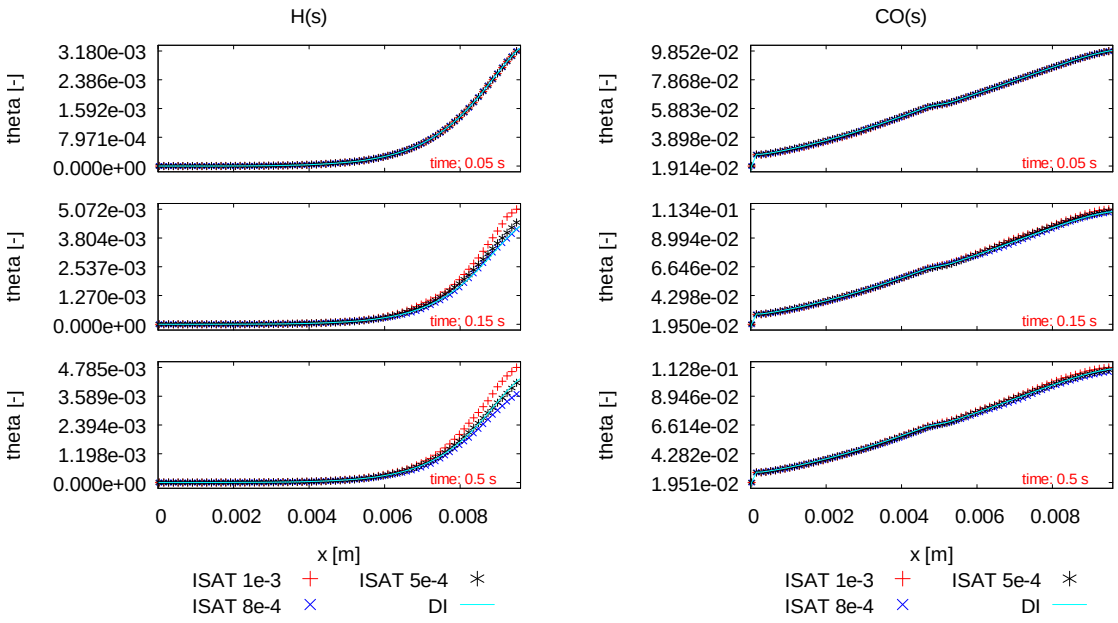


Figure 4.10 – Evolution profile of adsorbed main species – CH_4-UBI

The speed-up performance has been investigated considering the ratio between the time required to perform the whole simulation using ISAT and DI. The results, listed in Table 4.5, shows that the performance are function of the tolerance, but for each value the ISAT simulation is faster than the direct one up to 4 times.

Simulation Time DI [s]	274453		
Tolerance	$5 \cdot 10^{-3}$	$8 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
Simulation Time ISAT [s]	66188	72136	81673
Overall Speed-up factor	4.2	3.8	3.4

Table 4.5 – Simulation Time and Simulation speed-up factor –CH_4-UBI

Figure 4.11 shows the time step speed up factor in respect to the simulation time for a time step every 500. At the beginning of the simulation, the speed factor is about unitary due to the table settling.

Then, the speed-up factor grows to an average value between five and ten but it can reach at most about a value of twenty when the simulation approaches the steady state.

The profiles describing the number of events are shown in Figure 4.12. Initially, the number of retrieves is quite larger because the reactants have not reached the catalytic surface anymore.

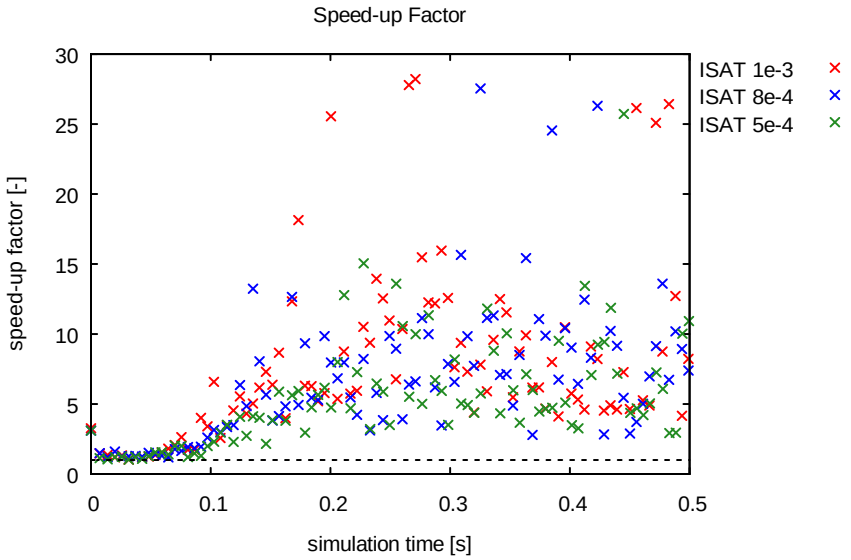


Figure 4.11 – Speed-up factor – CH_4-UBI

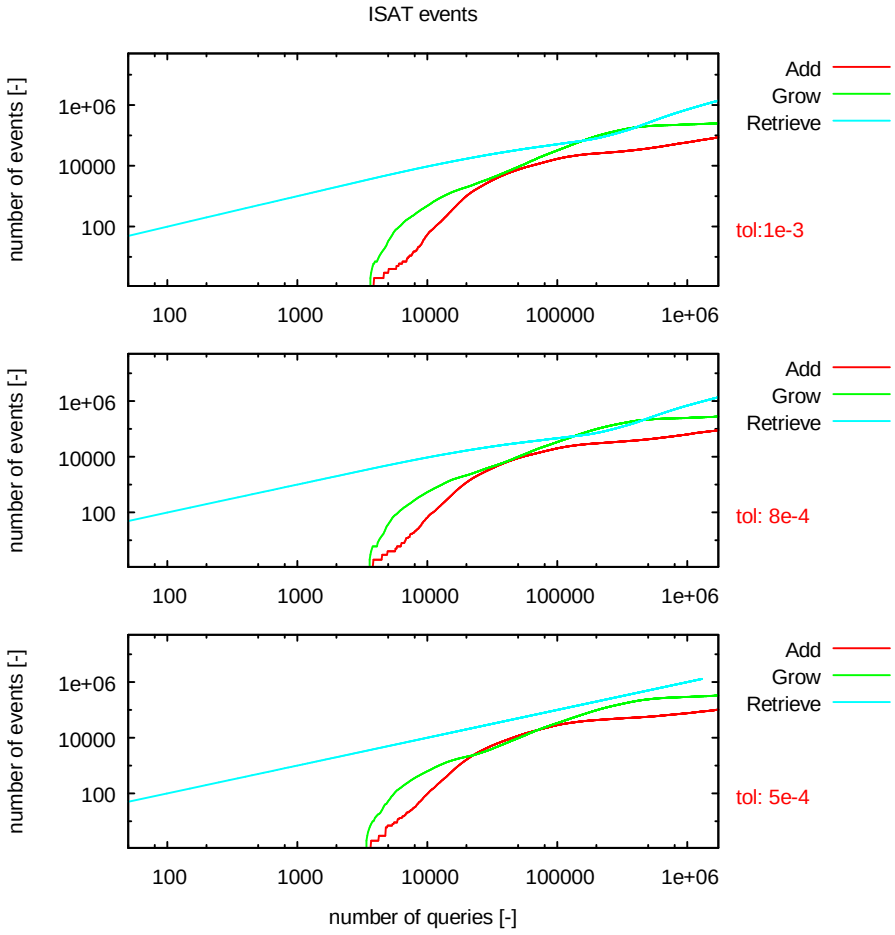


Figure 4.12 – Number of events vs number of queries – CH_4-UBI

Then, the number of additions and growths increases and, eventually, can overcome the number of retrieves especially using a low tolerance value. Finally, the table is settled and the ratio between the retrieves and the other events restarts to increase guaranteeing the speed-up of the simulations.

4.2.3 Case study: methane partial oxidation over a platinum gauze in adiabatic conditions

Finally, the performances of *ISATLib* are investigated analyzing the catalytic partial oxidation of methane over a platinum gauze with detailed kinetics schemes for gas-phase and surface chemistry. A detailed description of the kinetic schemes and geometry is reported in Deutschmann et al. [28]. It consists of 29 gas species, 11 solid species involved in 36 heterogeneous and 75 homogeneous reactions.

The gauze configuration is shown in Figure 4.13. It consists of two rows of six parallel platinum wires placed on top of each other. Due to symmetry considerations only a quarter of the catalyst gauze has to be taken into account. Thus, the computational domain requires a number of cells that is 4 times lower.

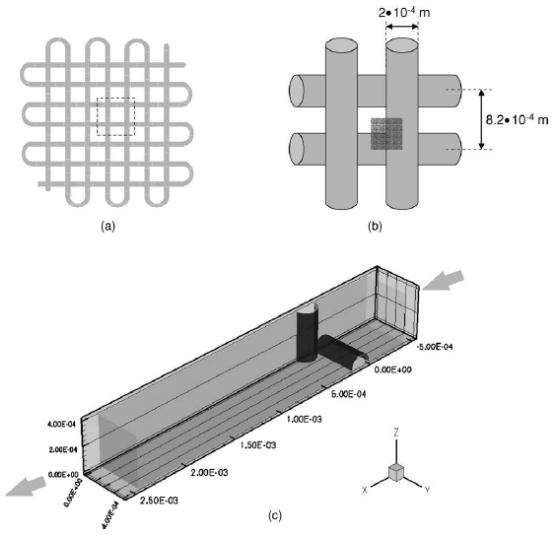
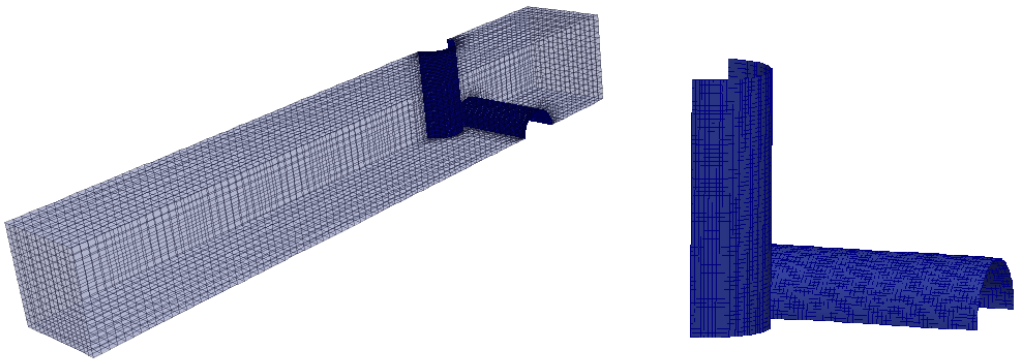


Figure 4.13 – Sketch of the platinum gauze [28]

The computational grid, as shown in Figure 4.14, is conveniently refined in the region near the platinum wires where the normal gradients are stronger. Moreover, the outlet of the reactor is graded in order to reduce the number of cells in a region where lower

chemical activity occurs. The computational domain includes a single contact point between two wires.



(a) (b)

Figure 4.14 – Sketch of the platinum gauze computation domain: (a) overall domain (b) wires detail

The dimension of the mesh is listed in Table 4.6.

Catalytic cells	3012
Total cells	19069

Table 4.6 – Mesh number of cells – platinum gauze

The reactive mixture consists of a methane and oxygen flow (volumetric ratio 2.5) diluted by 80% vol. of helium. The mass fraction of the species in feed composition and the operative conditions are listed in Table 4.7.

OPERATING CONDITIONS

H2 mass fraction	0.3125	**Inlet temperature**	600 K
O2 mass fraction	0.2500	**Outlet Pressure**	1.3 bar
He mass fraction	0.4375	**Flow velocity**	10 m/s

Table 4.7 - Operating conditions – methane partial oxidation – platinum gauze

In these simulations, the homogeneous reactions are introduced to test the performance of the ISAT method both in gas-phase and surface chemistry. The results are shown for a simulation settled with different tolerance values for homogeneous – $1 \cdot 10^{-6}$ – and heterogeneous – $1 \cdot 10^{-4}$, as suggested by the analysis reported in Section 3.2.3 and 4.2.2. The value used for the gas-phase chemistry is usually lower than the one required for the heterogeneous surface chemistry.

The results of the simulation show a good agreement between ISAT and direct integration, as shown in Figure 4.15. At the beginning of the simulation, the prediction of the CO mass fraction is underestimated but at the steady state the overlap between the DI and ISAT profiles is good.

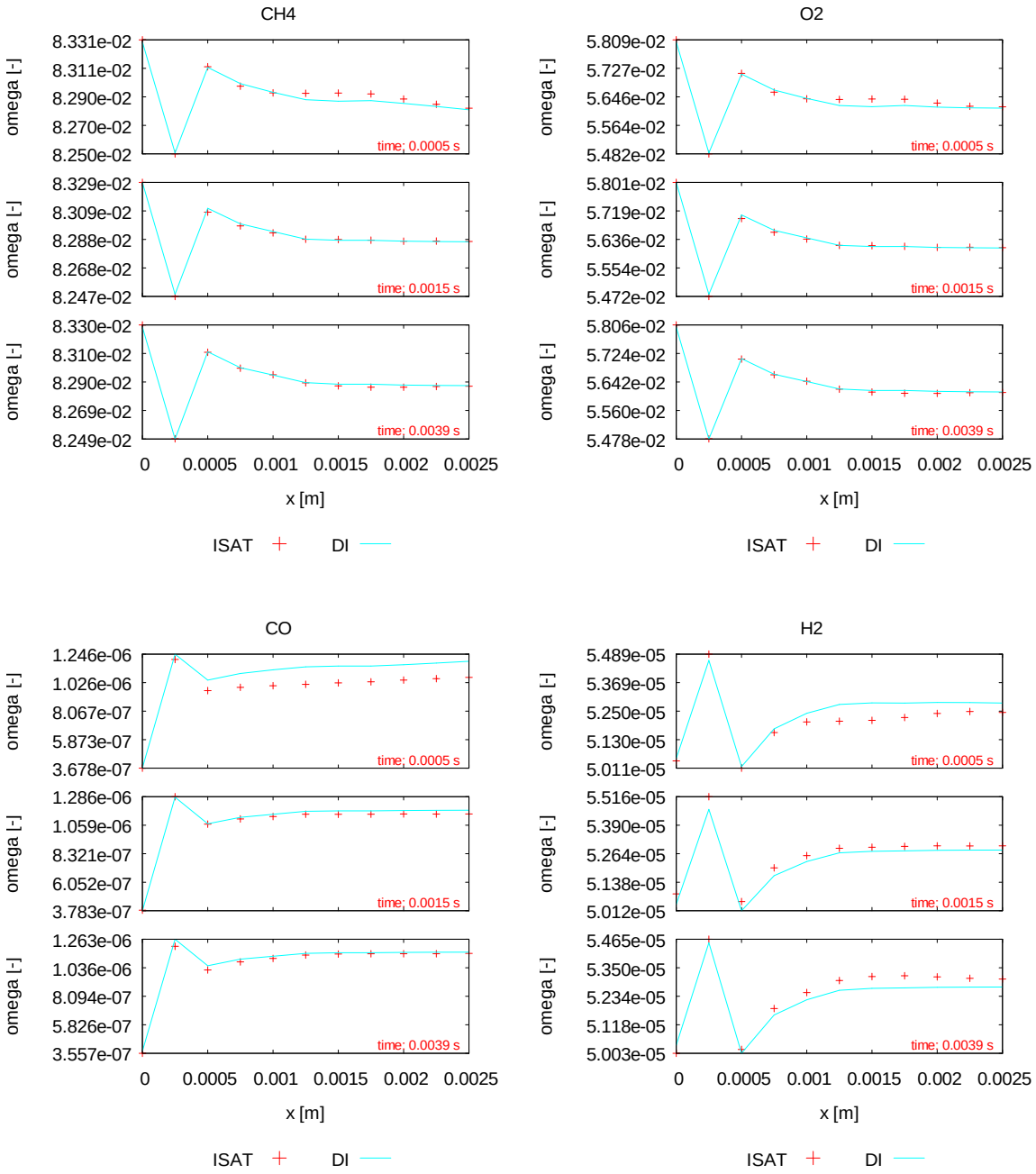


Figure 4.15 – Evolution profile of CO and CO2 – CH_4-CPO

The speed-up performance has been investigated comparing the time required to perform the whole simulation using direct integration ad ISAT, the results are listed in Table 4.8.

Simulation Time DI [s]	1012021
Simulation Time ISAT [s]	411198
Overall Speed-up factor	2.5

Table 4.8 – Simulation Time and Simulation speed-up factor – CH_4-CPO

Figure 4.16 shows the speed-up factors, i.e. the ratio between the time required to perform a single time step using both DI and ISAT, for a time step every 500.

At the beginning, the ISAT simulation is slower that the direct integration due to the large number of events needed to settle the table.

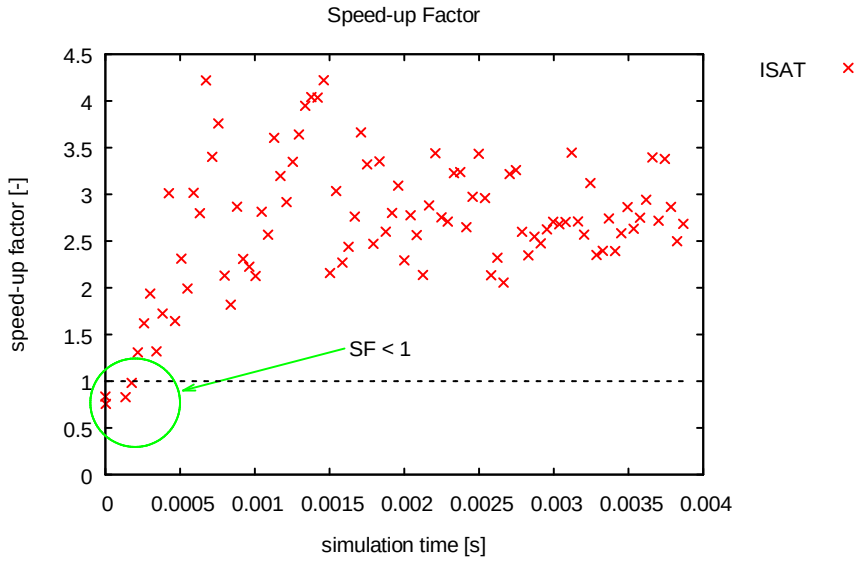


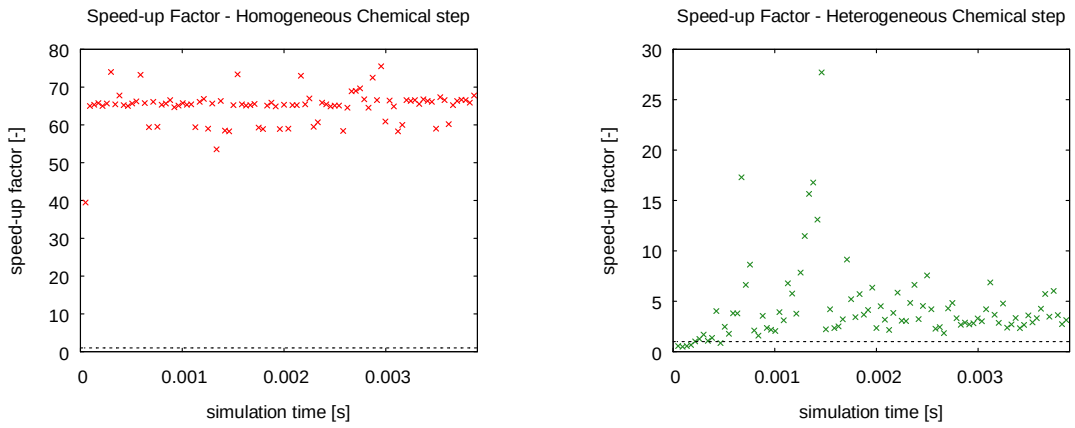
Figure 4.16 – Speed-up factor – CH_4-CPO

The performances of the *ISATLib* are investigated also considering the chemical speed-up factor, defined as follow:

$$SF_{chem} = \frac{wall\ clock\ time\ per\ time\ chemical\ step\ DI}{wall\ clock\ time\ per\ time\ chemical\ step\ ISAT} \tag{4.4}$$

Figure 4.17 a) shows SF_{chem} for homogeneous chemistry: the capability of ISAT to reduce the time required to perform the homogeneous chemical step is demonstrated. The time required is up to 70 times lower than the direct integration. This behavior is explained by two different reasons: first, the homogeneous reactivity is weak allowing a very large number of retrieves without a consistent table settling. Secondly, the high number of cells involved enhances the performances of the ISAT because the direct

integration has to solve a large number of time-consuming ODEs systems. The time spent carrying out the integration of the ODEs system is quite the same either the reactivity is weak or strong. In this case, due to the weak reactivity, the large of number of retrieves determines a fast linear approximation of the function value which strongly increases the speed of the process.



a) **b)**

Figure 4.17 –Speed-up factor a) homogeneous and b) heterogeneous chemical step – CH_4-CPO

Figure 4.17 b) shows SF_{chem} for heterogeneous chemistry. The *ISATLib* performances allow to achieve a speed-up in the surface chemistry description up to 3.5.This results is due to a low value of the tolerance used for the heterogeneous instance of the library. At the beginning of the simulation, the chemical speed-up factor is lower than 1 due to the table settling.

Moreover, the importance of the transport step in this system might not be neglected due to the large number of homogeneous cells in respect to the catalytic ones.

Figure 4.18 shows the number of events versus the number of queries for both gas-phase and surface chemistry. The surface chemistry requires a very large number of additions and growths in respect to the number of retrieves due to the low value for tolerance that has been set. On the contrary the homogeneous chemistry requires a lower number of additions and growths due to the lower importance of gas-phase reactions in these operating conditions.

The overall effect of the different behavior in the homogeneous and heterogeneous chemistry leads to the overall simulation speed-up of 2.5 times.

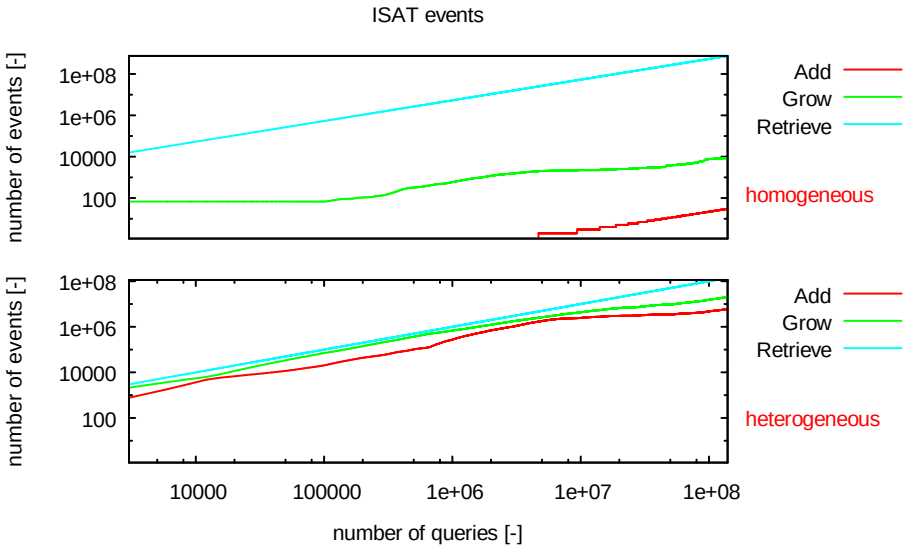


Figure 4.18 – Number of events vs number of queries – CH_4-CPO

4.3 Implementation in `catalyticFOAM-multiRegion`

The `catalyticFOAM-multiregion` [1, 2] solver is a numerical tool able to predict the behavior of heterogeneous catalytic systems, taking into account inter and intra-phase phenomena, as well as detailed kinetic schemes. The solver allows a complete description of all three regimes that can establish in a catalytic system: (i) chemical regime, (ii) intra-phase regime and (iii) external mass transfer regime.

The architecture of the solver is shown in Figure 4.19. The introduction of multiple regions requires an accurate characterization of the interface between the different regions and in particular of the communications among neighbor domains.

The iterative procedure begins with the solution of the characteristic equations for the fluid region; according to the operator splitting method, the reactive term is separated by the transport term following the so-called “Transport-Reaction-Momentum” order. First of all, the heat and mass transport equations are solved with the proper boundary

conditions. Then, the homogeneous reactions are solved. Finally, the PISO loop starts: the momentum and continuity equations are solved and the velocity field is corrected. Afterwards, the boundary conditions of the solid side are updated. The equations in the solid region are solved with the same approach: the mass and energy transport equations are solved and the side fluid interface values are updated. The iterative procedure, named PIMPLE loop widely described in previous work [1], continues until the interface convergence is reached.

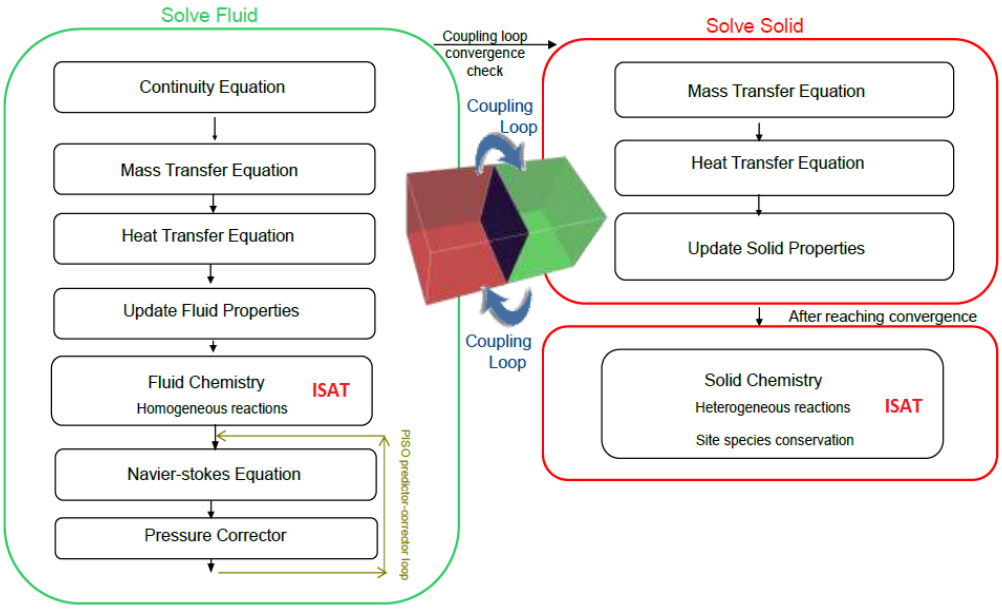


Figure 4.19 – catalyticFOAM-multiRegion architecture

Once the convergence criteria are satisfied, the ODE system of the heterogeneous reactions is solved. This methodology consists in decoupling the chemical steps from the interface convergence in order to better mimic what occurs in the physics of the system: the reactions take place only when the reactants reach the surface, thus when the convergence is achieved.

The `catalyticFOAM-multiRegion` has been developed, also, in a parallel version [1]: the computational domain is split in a certain number of subdomains and allocated to separate processors for solution. In order to avoid problems of synchronization of processors during the PIMPLE loop, the algorithm has been modified: the number of iterations is fixed to a sufficient value to ensure interface convergence.

The ISAT library has been implemented in the solver with the purpose of speeding-up the chemistry both in homogeneous and heterogeneous phases.

Two different ISAT tables are settled to manage the different reactive situations and the different dimensions of the composition space in heterogeneous and homogeneous chemistry. The solver provides, during the initialization, the tolerance and the `scalingError` matrix (described in Section 2.1.3) for both the instances.

The architecture of the connections between the solver and *`ISATLib`* are as shown in Figure 4.2. The CFD code interface passes the scaled composition vector to the library. If a leaf covers the query point, the linear approximation is returned. Otherwise, if the retrieve procedure is unsuccessful, the ODE solver of `catalyticFOAM` performs a direct integration and a grow attempt is carried out. If successful, the ellipsoid of the leaf is enlarged. On the other case the mapping gradient is evaluated and a new record is added to the table.

The vector returned by the *`ISATLib`* to the `catalyticFOAM-multiRegion` solver can contain some negative values. As proposed for the single region approach, the composition vector is scanned, the negative values are set to zero and, finally, the mass fractions are normalized to obtain a unitary sum.

4.3.1 Case study: hydrogen combustion in adiabatic conditions over rhodium

In order to assess the reliability of the solver with ISAT implemented, several simulations have been performed in an annular adiabatic reactor, as shown in Figure 4.20. The catalyst consists of Rh/α-Al_2O_3 and it is deposited on the inner wall of the reactor in order to form a uniform and thick layer.

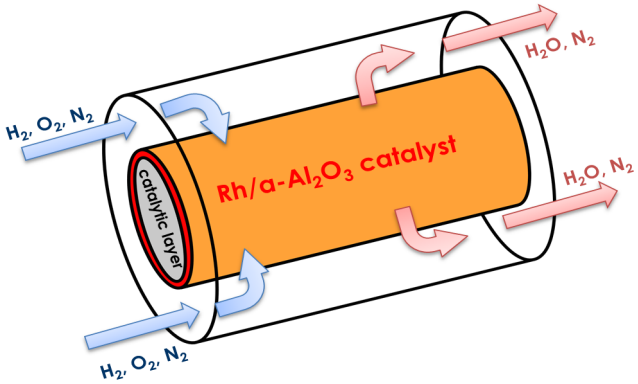


Figure 4.20 – Sketch of the annular catalytic reactor - multiRegion

The computational domain is different from the one used in 4.2.1, it takes into account also a solid phase with a thickness of 50 μm, as show in 4.21.

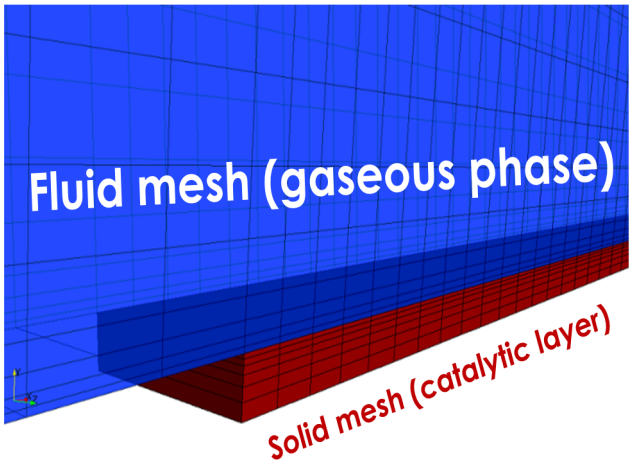


Figure 4.21 – Short computational domain

The simulations are performed using a detailed microkinetic model of hydrogen combustion, the scheme consists of 18 reactions and 5 adsorbed species [24]; the reactions in the homogeneous phase are neglected.

The dimension of the mesh is listed in Table 4.9.

Fluid cells	1800
Solid cells	390

Table 4.9 – Mesh number of cells – platinum gauze

The operating conditions are listed in Table 4.10:

OPERATING CONDITIONS

H2 mass fraction	0.002985	**Inlet temperature**	473 K
O2 mass fraction	0.011845	**Outlet Pressure**	1.01 bar
N2 mass fraction	0.98517	**Flow velocity**	0.171 m/s

Table 4.10 – Operating conditions – hydrogen combustion - multiRegion

The simulations are carried out using different values of the tolerance to find out an acceptable one and to exploit the ISAT speeding-up performances. The results of the simulations have been compared with the ones provided by a simulation carried out using direct integration of the chemical step.

The overlap between the profiles for the gas species is good for each tolerance value, as shown in Figure 4.22.

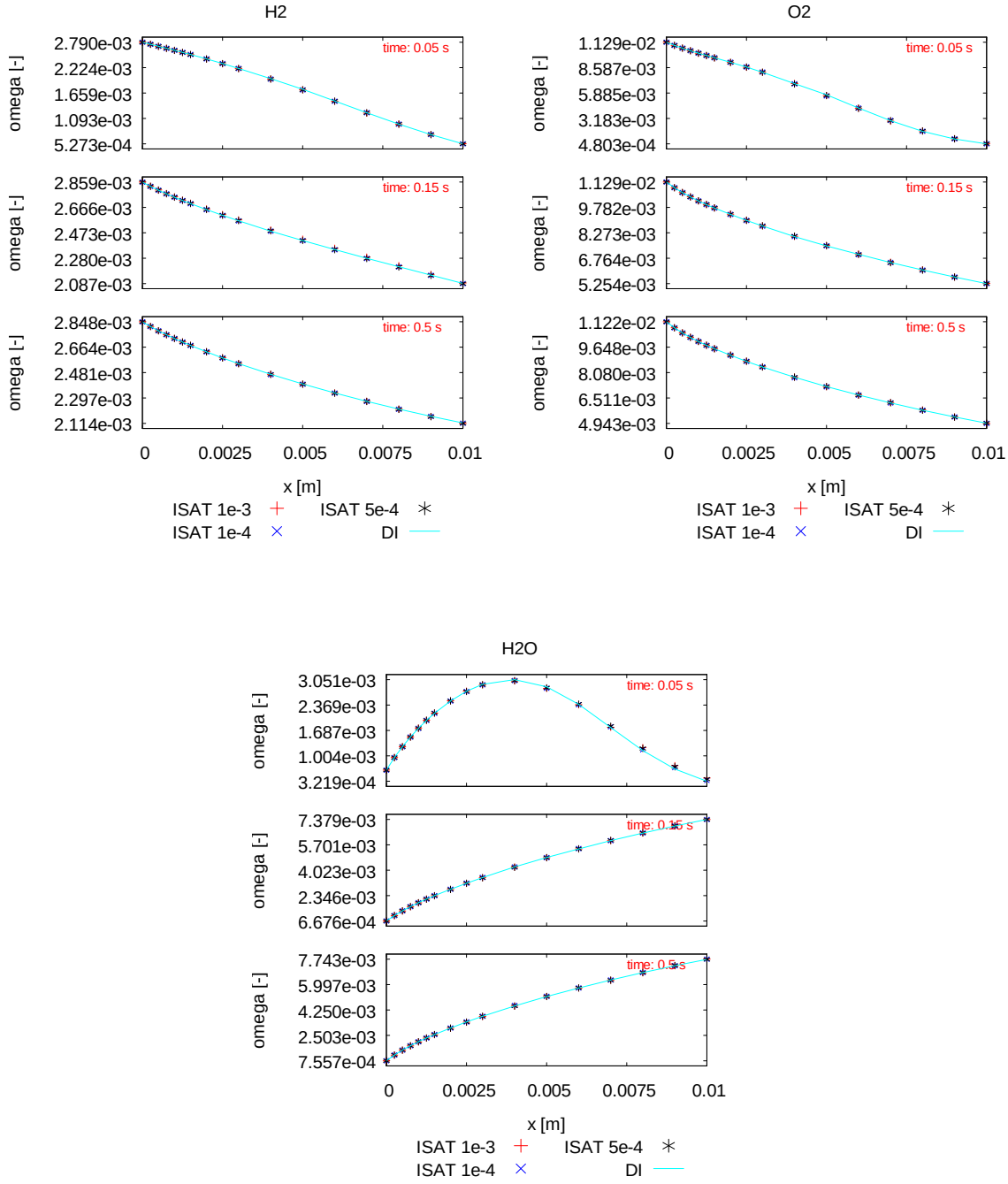


Figure 4.22 – Evolution profile of gas-phase main species – H_2-UBI – multiRegion

Figure 4.23 shows the evolution profile of the sites coverage for adsorbed oxygen and rhodium. The agreement is good for the lowest tolerance, whereas a not perfect overlapping is experienced for the two larger ones especially for rhodium.

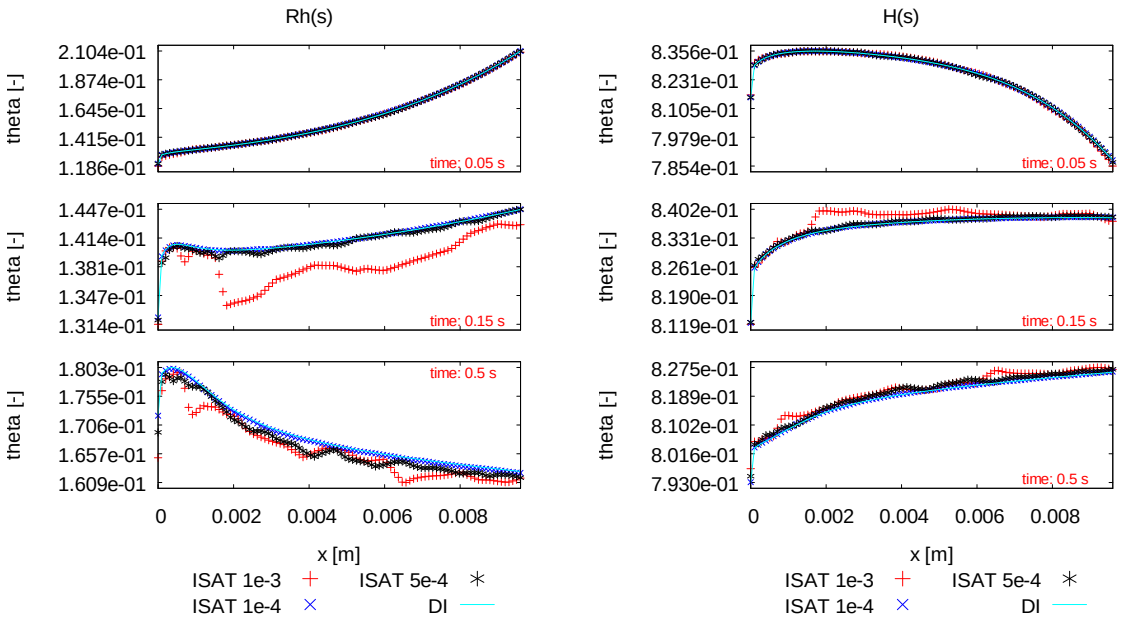


Figure 4.23 – Evolution profile of adsorbed main species – H_2-UBI – multiRegion

Figure 4.24 shows the ratio between the time required to perform a single time step using DI and ones using ISAT, in respect to the simulation time for a time step every 500. The simulation speed-up factor is, from the beginning, larger than 1 and it reaches an average value larger than 4 after 0.15 s of simulation. Thus, the simulation is speeded-up also in the initial phase of the table settling. The speed-up factor depends on the tolerance value: a larger value leads to a larger speed-up, but also to inaccurate results.

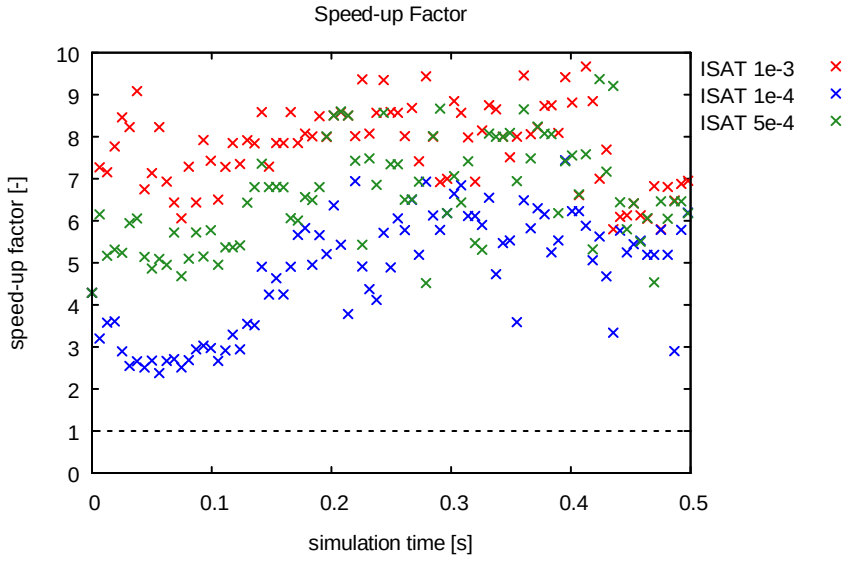


Figure 4.24 – Speed-up factor – H_2-UBI – multiRegion

The speed-up performance has been investigated first considering the ratio between the time required to perform the whole DI simulation and the ones required to perform the whole ISAT simulation.

The results are listed in Table 4.11:

Simulation Time DI [s]	44054		
Tolerance	$8 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
Simulation Time ISAT [s]	5779	6609	10519
Overall Speed-up factor	7.6	6.7	4.2

Table 4.11 – Simulation Time and Simulation speed-up factor – H2-UBI – multiRegion

The whole simulation performed with ISAT is faster than the direct integration up to 4 times without any loss in accuracy using a tolerance value of $1 \cdot 10^{-4}$ or lower.

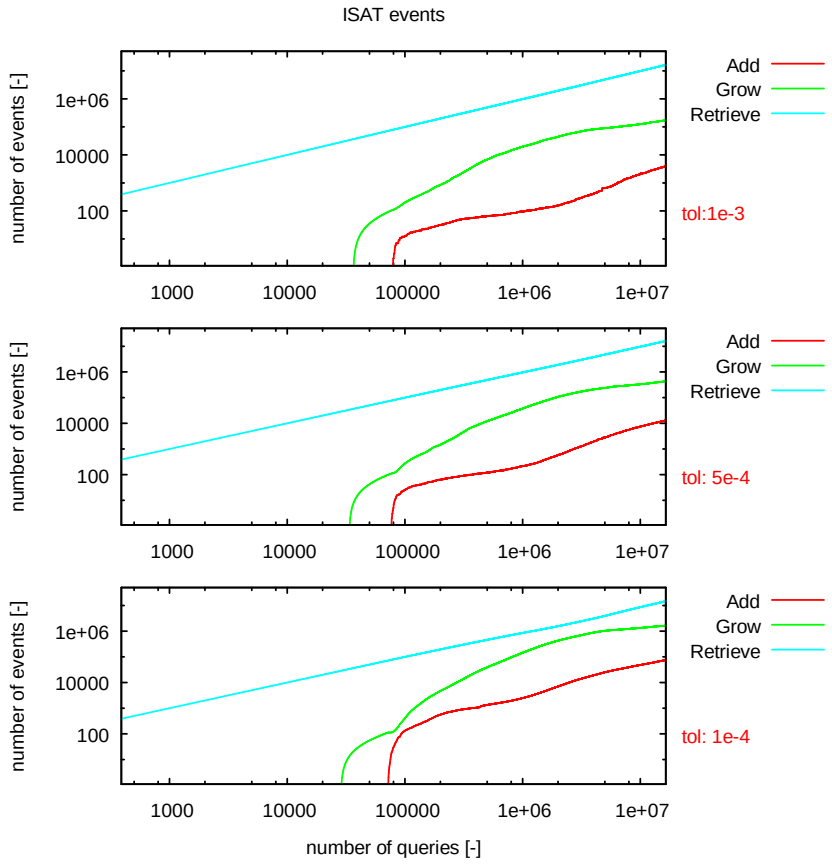


Figure 4.25 – Number of events vs number of queries – H2-UBI - multiRegion

The profiles describing the number of events are shown in Figure 4.25. Initially, the number of retrieves is quite larger because the reactants have not reached the catalytic surface anymore and a single leaf can solve every query. Then, the number of additions and growths increases due to the table settling. Finally, the algorithm continues to add a small number of leaves but the retrieve becomes the more common event and the speed-up of the simulation is ensured.

4.3.2 Case study: methane partial oxidation in an annular reactor in adiabatic conditions

In order to stress the features of the *ISATLib*, a second test case is carried out in the same geometry described in Section 4.3.1 using a more complex kinetic scheme. The catalytic partial oxidation of methane is investigated using a UBI microkinetic model [24], which consists of 16 gas species, 13 solid species involved in 80 heterogeneous reactions, to describe the surface chemistry.

Several simulations have been performed with different tolerance values in adiabatic conditions in order to exploit the effect of the tolerance on the overall accuracy and speed-up.

The feed composition and the operative conditions are listed in Table 4.12:

OPERATING CONDITIONS

CH_4 mass fraction	0.1565	**Inlet temperature**	873.15 K
O_2 mass fraction	0.1753	**Outlet Pressure**	1.01 bar
N_2 mass fraction	0.6682	**Flow velocity**	0.171 m/s

Table 4.12– Operating conditions methane partial oxidation - multiRegion

The simulations are performed using three different tolerance values – $5 \cdot 10^{-3}$, $2.5 \cdot 10^{-3}$, $8 \cdot 10^{-4}$.

Figure 4.26 shows the evolution profiles of the mass fraction for the main gas-phase species: the agreement between ISAT and direct integration is good only for the stricter tolerance values. The two larger tolerances determine an unacceptable forecast on the mass fraction of the species due to the large admitted error.

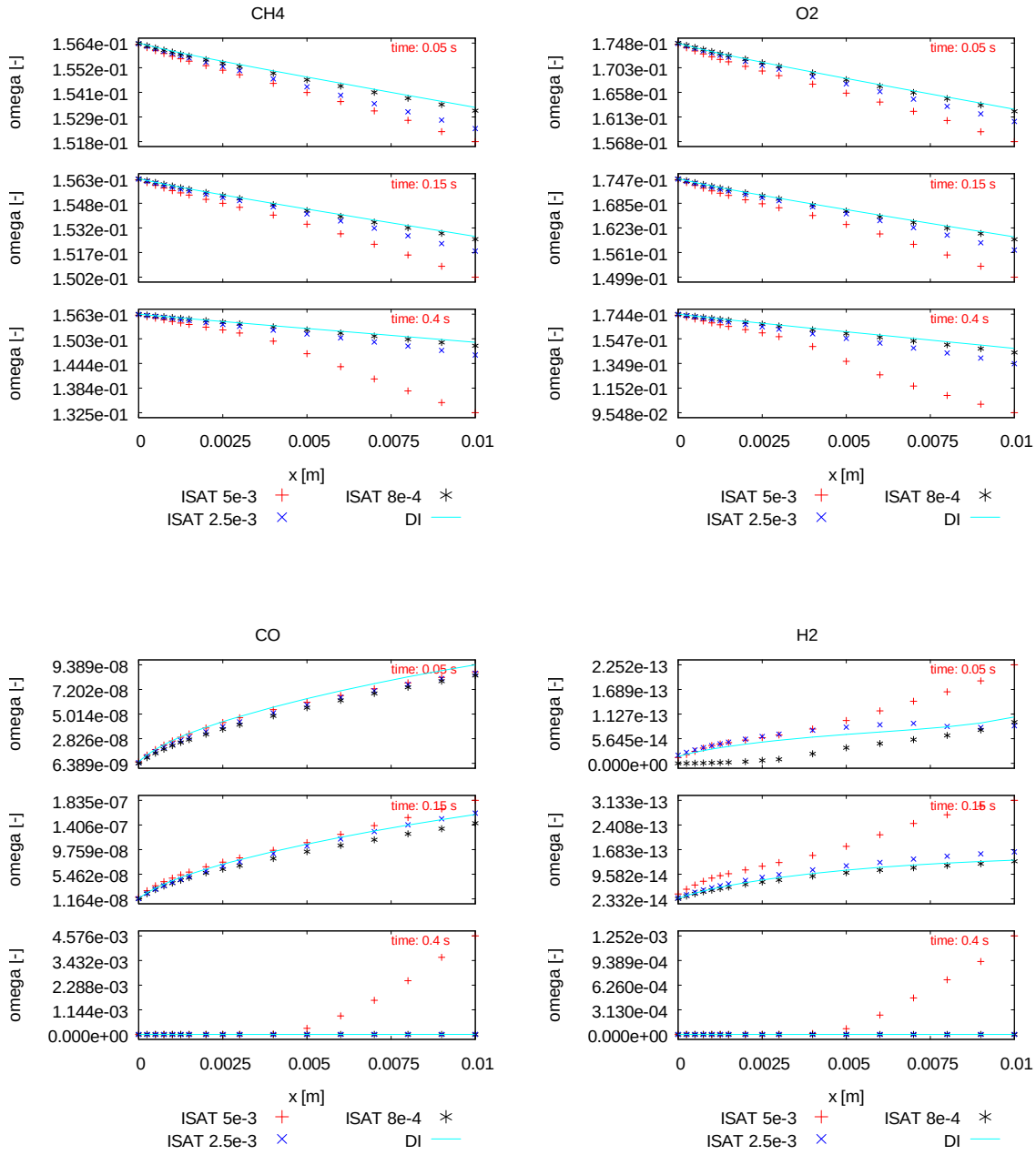


Figure 4.26 – Evolution profile of gas-phase main species – CH_4-UBI – multiRegion

The same behavior occurs in the profiles of adsorbed species, as shown in Fig. 4.27. The profiles are obtained plotting the site coverage of some species in the middle of the catalytic layer. The two larger tolerance values provide unacceptable profiles which do not agree with the ones obtained by direct integration.

The smallest tolerance value allows a good agreement between the prediction of the *ISATLib* modified solver and direct integration.

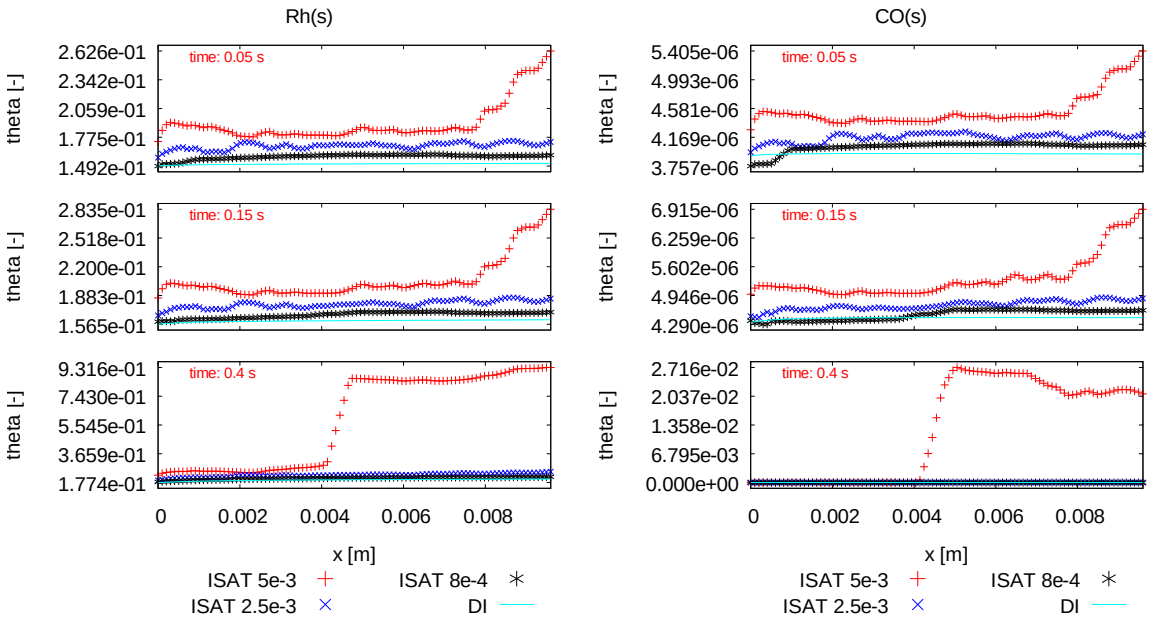


Figure 4.27 – Evolution profile of adsorbed main species – CH_4-UBI – multiRegion

Figure 4.28 shows the time step speed up factor SF in respect to the simulation time for a time step every 1000. The simulation speed-up factor is, from the beginning, higher than 1 and it reaches an average value larger than 20. At the end of the simulation, the speed-up factor decreases due to further additions and growths that are necessary to satisfy next queries.

The simulations have been stopped after 0.4 s of simulation due to large computational time required by the direct integration simulation. For this reason the simulations have not reached the steady state, thus some additional table inserting operations are required to satisfy the queries, explaining the decreasing of the speed-up at the end of the simulation. On the basis of the previous test-cases, it is possible to speculate that the speed-up will increase as the solution will reach the steady state conditions.

The speed-up performance has been investigated first considering the ratio between the time required to perform the whole DI simulation and the one necessary to carry out the whole ISAT simulation.

The *ISATLib* shows its potentiality in this case study: the speed-up of the overall simulation is up to 9.5 ensuring, at the same time, a good overlap between the profiles. The speed-up achieved with the two larger tolerance are greater but lead to unacceptable approximation error.

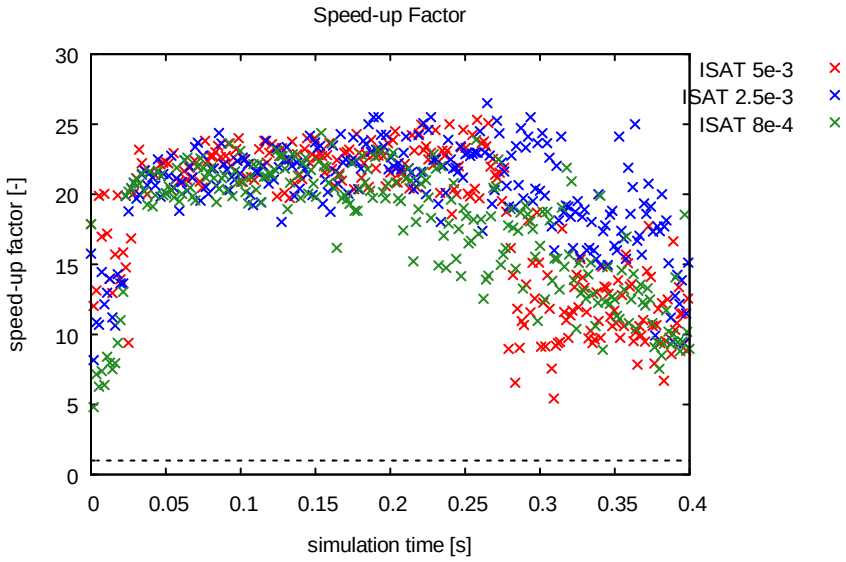


Figure 4.28 – Speed-up factor – CH_4-UBI – multiRegion

The results are listed in Table 4.12:

Simulation Time DI [s]	1924495		
Tolerance	$5 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$	$8 \cdot 10^{-4}$
Simulation Time ISAT [s]	149920 s	161854 s	201980 s
Overall Speed-up factor	12.8	11.8	9.5

Table 4.13 – Simulation Time and Simulation Speed-up factor – CH_4-UBI – multiRegion

In this simulation, the time necessary to evaluate the chemical step with ISAT can be measured directly and it is possible to compare it to the time required to perform the same step with direct integration. Unfortunately, this analysis cannot be carried out in all the other simulations because the ISAT chemistry is often too fast and it is difficult to measure the time required to solve the chemical step with *ISATLib*.

Figure 4.29 shows the chemical speed-up factor in respect to the simulation time for a time step every 1000, for the simulation with the tolerance value of $8 \cdot 10^{-4}$. The speed-up achieved in the chemical step is up to 500 ensuring the capability of the *ISATLib* to speed-up the solution of the chemical step. The speed-up factor is near 1 during the table settling phase due to the addition and growing procedure.

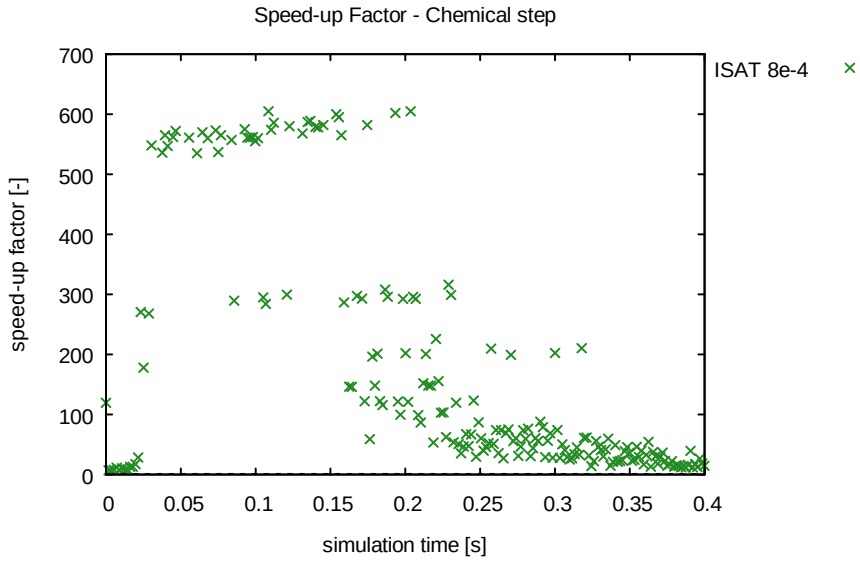


Figure 4.29 – Chemical Speed-up factor –CH_4-UBI – multiRegion

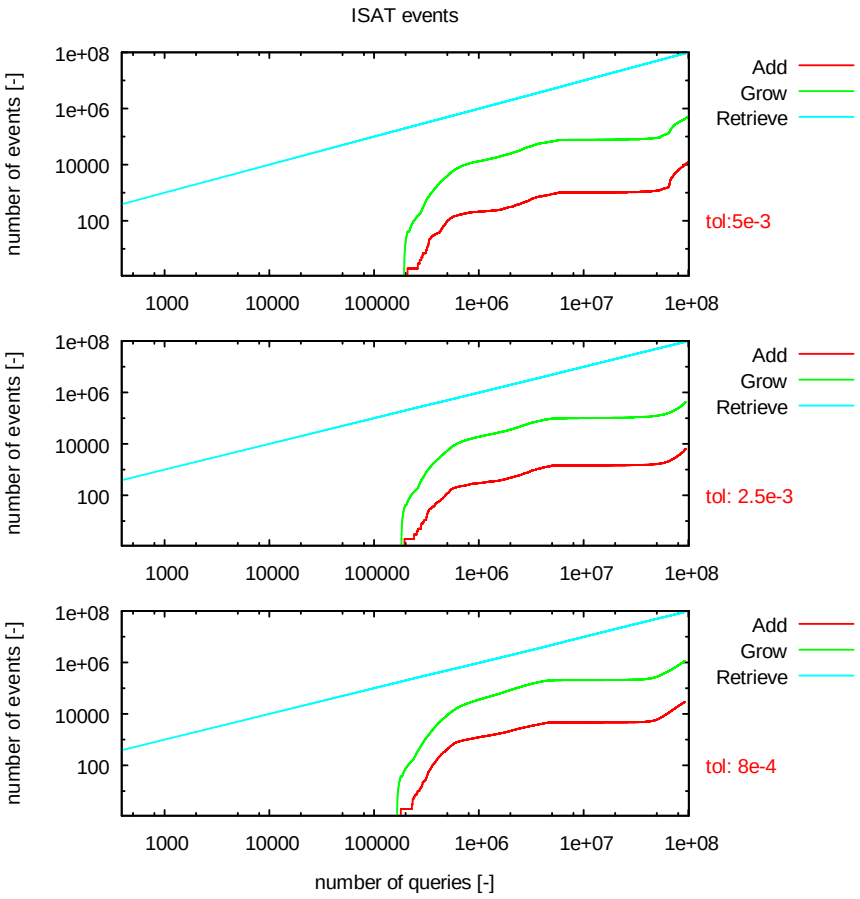


Figure 4.30 – Number of events vs number of queries – CH_4-UBI - multiRegion

After the table settling, the number of retrieves becomes dominant leading the chemical step to become two orders of magnitude faster than the direct integration. Unfortunately, the overall simulation speed-up is up to ten due to the time spent in the transport phase which is independent from the methodology used to solve the chemical step.

The profiles describing the number of events are shown in Figure 4.30. Initially, the number of retrieves is quite larger because the reactants have not reached the catalytic surface anymore and a single leaf can solve every query. After about 100,000 events, the number of additions and growths increases due to the table settling. Then, the table contains a sufficient number of leaves that satisfies the next query and the number of additions and growths becomes constant.

4.4 Conclusions

The *ISATLib* has been implemented both in `catalyticFOAM` and `catalyticFOAM-multiRegion` to reduce the time spent performing the ODE integration step.

The reliability of the modified solvers has been demonstrated comparing the results of the ISAT simulations with reference cases obtained by direct integration. The agreement between the profiles is good for both solvers if the tolerance value is set to a reasonable value. The large number of simulations performed allows to assess that a tolerance value lower than $5 \cdot 10^{-4}$ ensures a satisfactory agreement between ISAT and direct integration. Sometimes larger values are possible but they have to be validated with an available reference case.

The speed-up of the overall simulation is directly proportional to the tolerance value used: the larger the tolerance, the higher the speed-up. The achieved speed-up is up to 4.5 using a tolerance value resulting in good simulation results.

The performances of the algorithm improve with the dimension of the kinetic scheme: the simulation of the hydrogen and methane oxidations, in similar stiffness conditions, shows a higher speed-up for the latter due to the larger ODE system that has to be solved by direct integration. The time required to perform the fast interpolation

procedure is quite linear with the kinetic scheme dimension, but it is not affected by the number of species that are considered. The time required to perform the direct integration of the ODEs system is strongly affected by the dimension of the composition space and by the number of species involved. For this reason, the performances of the *ISATLib* grow increasing the dimension of the kinetic scheme.

The *ISATLib* decreases the time spent in the chemical step of the operator splitting up to two orders of magnitude and, at least, by ten times. For this reason the best performances are achieved when the time spent in the transport step is lower than the time required to solve the chemistry. The larger is the difference between the time spent in solving the two terms, the larger will be the possible speed-up of the simulation.

The implementation of the *ISATLib* within the `catalyticFOAM` framework has decreased the time required to carry out the simulation without losing in accuracy and stability. The capability of the algorithm to speed-up the simulation depends also on the time spent in solving the transport step.

5 Industrial showcase

In this Chapter several packed beds with different pellet shapes have been investigated in order to assess the capability of the `catalyticFOAM-multiRegion`*, improved with* `ISATLib`*, to describe their behavior. These simulations have been carried out during an internship period at the CFD Department of BASF production plant in Ludwigshafen (Germany).*

5.1 Overview

Catalysis and catalytic processes are central to many aspect of Chemical Engineering. The most part of the actual catalytic processes is carried out in packed bed reactors.

Employing CFD, the catalytic reactors can be described with their real geometry, motion flow field, inner-pellet transport limitation and detailed heterogeneous surface reactivity, providing an useful and powerful instrument for both the design phase and the control strategies.

One of the actual limit of the reactor description by means of CFD tools is the unfeasible mesh dimensions of the whole industrial reactor: only a very small part of the reactor can be described in both feasible simulation time and computational effort. However, the description of a small section of the reactor allows to investigate the effect of different shapes and dimensions of the pellet, of different supports and catalyst composition providing an alternative way in respect to the laboratory experiments.

Therefore, the performances of the `ISATLib` on relevant industrial cases has to be investigated to exploit its features in order to reduce the gap between the largest feasible CFD simulation of packed bed and the industrial systems. Moreover, the reduction of the simulation computational time can improve the feature of the tools allowing a wider field of application.

5.2 Packing generation and parallel solving

The methodology used to set the industrial packed bed is explained step by step: first the *DEM* technique, able to randomly generate packed bed structures, is described. Then the mesh generation is analyzed in detail focusing on the problems observed. Finally the description of the mesh decomposition and the parallel simulation are explained.

5.1.1 *DEM* methodology

The mesh generation of the packed bed reactor is performed using a mathematical model, based on *DEM* (Discrete Element Method). This method describes the mechanical behavior of assembling the different shapes of pellets through an explicit numerical scheme, in which the interaction of the particles is monitored contact by contact and the motion of the particles is modeled particle by particle.

A computational tool based on these methods has been used in order to generate random packed beds characterized by different pellet shapes. These beds will be used as the computational geometry for the simulation of steam reforming. Usually, these operations are carried out in multi-tubular reactors, and for reasons of computational time only a part of a single tube is considered for the simulations with `catalyticFOAM` solver.

The Discrete Element Method is a family of numerical methods for computing the motion of a large number of particles. Though *DEM* is close to molecular dynamics, the method is generally distinguishable by the inclusion of rotational degrees of freedom as well as state full contact and, often, complex geometries.

DEM code solves the second Newton law of motion and it is able to provide random spatial orientations of the particles filling the reactor.

The following forces have to be taken into account:

- Friction: when two particles touch each other
- Contact plasticity: when to particles collide
- Gravity: attraction between particles due to their weight
- Attractive potentials: such as cohesion, adhesion, electrostatic attraction

All the forces are added up to determine the resulting forces acting on each particle. A certain integration method is employed to compute the change, both in the position and velocity, of each particle during a certain time step from Newton's law of motion. Then, the new positions are used to compute the forces during the next step and the loop is repeated until the simulation ends.

The equations of motion of particles are solved using the second-order Adams-Bashforth method [29], which estimates the values of contact forces in the next calculation time by linear extrapolation, or by multi-step methods such as the predictor-corrector method [30].

Commercial and open-source softwares can be exploited to setup these beds:

- *SDEC*: Spherical Discrete Element [31]
- *Yade*: Yet Another Dynamic Engine, a modular toolkit of *DEM* algorithms written in C++, running in parallel with OpenMP [32]
- *LAMMPS*: fast parallel open-source molecular dynamic package with GPU support allowing *DEM* simulation [33]
- *LIGGGTHS*: is a code based on *LAMMPS* with more *DEM* features such as wall import from cad, moving mesh and granular heat transfer [34]

The *DEM* model adopted considers that a particle has two types of motion: translational and rotational, which can be described by Newton's law of motion [35]. These equations, based on the forces and torques originated from its interaction with neighboring particles, are given by:

$$\begin{aligned} m_i \frac{d\boldsymbol{v}_i}{dt} &= \sum_j \boldsymbol{f}_{ij} + m_i \boldsymbol{g} \\ I_i \frac{d\boldsymbol{\omega}_i}{dt} &= \sum_j \boldsymbol{t}_{ij} \end{aligned} \tag{5.1}$$

where m_i, I_i, $\boldsymbol{v}_i$, $\boldsymbol{\omega}_i$ are the mass, moment of inertia, translational velocity and angular velocity of the particles *i*, respectively. The involved forces are the gravitational force and the interaction force $\boldsymbol{f}_{ij}$ between particles *i* and *j*, which depends on the deformation of particle *i*. The torque acting on particle *i* by particle *j*, $\boldsymbol{t}_{ij}$, stems from the contact force and causes particle *i* to rotate or slow down their relative rotation. Wall is treated as a particle of infinite size.

The components of the forces and torques acting on a particle *i* are:

- Normal elastic forces

- Normal damping forces
- Tangential forces
- Columbian friction forces
- Torque by tangential forces
- Rolling friction torques

Therefore, the trajectories, velocities and angular velocities of all particles can be traced solving Equations (5.1).

The first step in a *DEM* simulation is the definition of the basic structure of the bed, i.e. the pellet shape. Usually, this operation is performed using a CAD software able to produce a STL file. This format, named *Standard Tessellation Language*, describes a raw unstructured triangular surface by the unit normal vertices, ordered by the right-hand rule, of the triangles using a three dimensional Cartesian coordinate system.

Then, the number of pellets in the bed or its length has to be chosen to set up the simulation environment.

Next step, the geometry of the tube, where the particles have to fall down and randomly pack, has to be generated.

The structure has an enlargement in the upper part in order to allow to the unit structure to enter inside the reactor in an easier way (see Figure 5.1). The *DEM* code solves balances of forces in order to calculate the position of equilibrium of each unit that will establish the final bed. The algorithm proceeds iteratively until the velocity of each particle becomes lower than a threshold value fixed by the user.

The frame of Figure 5.1 shows the results of the *DEM* simulation.

Then, a *DEM* packing geometry is exported into a text file containing the information regarding dimensions and positions of all the particles in the bed.

Usually a *DEM* code is able to treat only spherical particles. If a different shape is required, it has to be built using assembling spheres. This structure is generated by placing spheres on the surface of the pellet shape which is defined by the diameter and number of particles on the surface. Since the spheres are staggered on the surface, irregularities can be found. However, the smoothness of the surface can be adjusted by the relative positions and diameters of the spheres. A smoother surface can be achieved

using more spheres with smaller diameters, increasing the numerical complexity of the simulation.

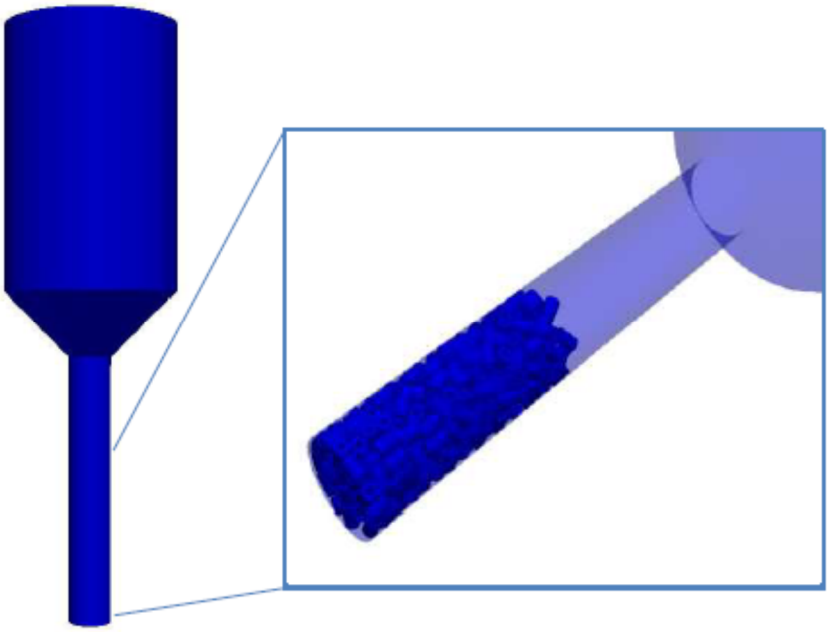


Figure 5.1 – Results of a DEM simulation

5.1.2 *DEM* simulation results

In this work, two different geometries were investigated, as shown in Figure 5.2:

- Rings
  - Large-Hole or First type
  - Small-hole or Second type (pellet shape for steam reforming)
- "7 Holes" [36] (patented pellet shape for steam reforming)

Ring – Large-Hole Ring – Small-Hole "7 Holes"

Figure 5.2 – Pellet shapes – Steam Reforming

The dimensions of the pellets are shown in Figure 5.3

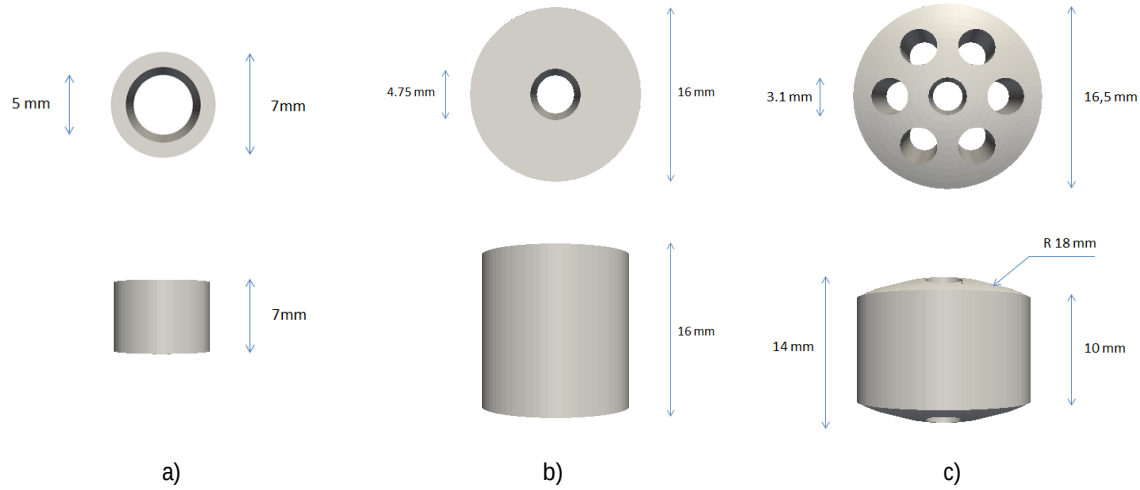


Figure 5.3 – Pellet dimension – a) Ring (Large-Hole), b) ring (Small-Hole), c) "7 Holes"

The number of pellets in each catalytic bed has been calculated to have the same catalytic mass. The *DEM* permits to set up catalytic beds with thousands of particles. However the number of particles that can be modeled in CFD is significantly less and is limited by available computational resources. Only a small section of the entire packed bed is considered to fit the available computational resources.

An example of a packed bed obtained through the *DEM* simulation is presented in Figure 5.4

Figure 5.4 – Example of DEM packed bed

One of the aim of this work is to investigate the behavior of these packed beds in different conditions. Considering the same available catalytic mass, the results of the `catalyticFOAM` simulations will be dependent on the fluid dynamics around the pellets and the capability of these geometries to transfer mass and energy both inter and intra-phase.

5.1.3 Meshing phase

The result of the *DEM* simulation is a STL file that has to be used to obtain the proper mesh for both the tube and the packed beds. The OpenFOAM framework [4] provides a meshing tool named *SnappyHexMesh* able to build up a mesh starting from an STL file.

The first step of the meshing procedure is to set up a background mesh. Then, the meshing tool will reshape the background mesh in order to introduce the geometries in the STL file.

The most important parameter of the meshing phase is the level of refinement: the higher it is, the more accurate and smoother are the surfaces of the pellets. Moreover, the points of contact between the particles and the wall are better described increasing the level of refinement. At these points of contact, mesh cells become highly skewed, which results in a poor-quality mesh and often can cause convergence and stability problems in CFD simulation.

Figure 5.5 shows some typical problems due to a low level of refinement: surface aberrations, which determines an irregular surface, and merged pellets, which causes difficulties in the analysis of intra-particles transport.

In order to overcome these problems, it is possible to selectively increase the level of refinement only in these problematic zones.

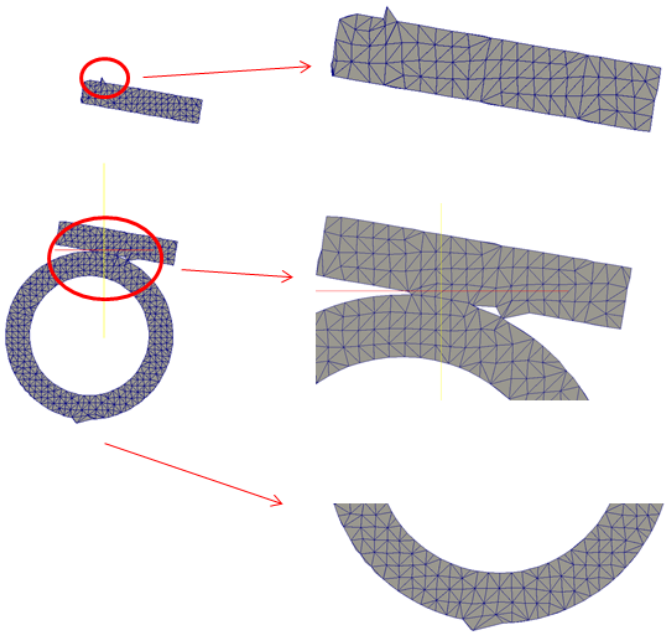


Figure 5.5 – Meshing problems due to low level of refinement

Ideally, the level of refinement should be the highest possible in order to describe the particles and the tube in the best way. Unfortunately, the number of cells required in case of very high level of refinement leads to an unfeasible dimension of the mesh for the available computational resources. For this reason, the high level of refinement is set only in the region of the reactor tube where the packed bed is present, permitting a low level of refinement in the reactor inlet and outlet. In this way, the number of cells is kept to an acceptable value without any loss in accuracy in the bed description, as shown in Figure 5.6.

The inlet and the outlet of the reactor show a coarser mesh and an axial grading in order to reduce the number of cells where the chemistry is not effective. Despite, the mesh is finer in the pellet section allowing an accurate description of the chemical and transport phenomena ensuring, at the same time, stability to the system.

Another purpose of this work is to exploit the effect of the intra-phase transport resistance influence on the system activity and, for this reason, the particles are meshed following a multi-region approach. It is necessary to define the level of refinement both for the fluid region and the solid one. It is important to point out that the two meshes have to be conformal, which means that the number of interface cells between the two regions has to be the same. It is possible to achieve this aim only using the same level of refinement for the two regions.

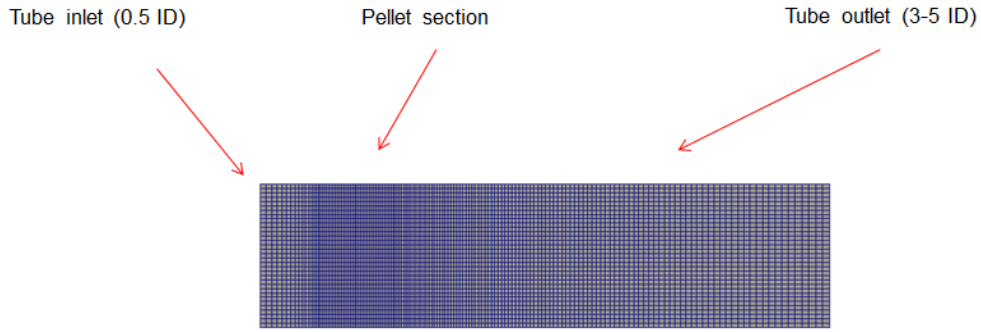


Figure 5.6 – Background mesh example

The resulting mesh obtained from `snappyHexMesh`, is shown in Figure 5.7. The description of the pellet zone is accurate thanks to the high refinement level used, while the reactor region farther from the reacting zone presents large cell dimension. The

inner part of the pellets is described with a high level of refinement to allow an accurate description both of the intra-phase diffusion phenomena and heterogeneous reactions.

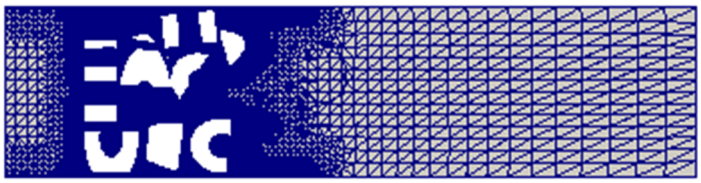


Figure 5.7 – Longitudinal mesh section – fluid region

Due to the large dimension of the computational domain, the simulations have been performed using the parallel version of the solver on several CPUs, up to 12.

The mesh is decomposed using the OpenFOAM `decomposePar` utility in order to break-up the domain. The geometry and fields are broken up according to a set of parameters specified in a dictionary named `decomposeParDict`, that must be located in the system directory of the case of interest.

The user has a choice of four methods of decomposition, specified by the method keyword as described below:

1. `simple`: simple geometric decomposition in which the domain is split into pieces by direction, e.g. 2 pieces in the x-direction, 1 in the y-direction, 1 in the z-direction
2. `hierarchical`: hierarchical geometrical decomposition which is the same as the `simple` except that the user specifies the order in which the directional split is done, e.g. first in the y-direction, then in the x-direction
3. `scotch`: decomposition performed with the Scotch code [37],which requires no geometric inputs from the user and attempts to minimize the number of processor boundaries. The user can specify a weighting for the decomposition between processors, through an optimal `processorWeigths` keyword which can be useful on machines with different performances between processors
4. `manual`: manual decomposition where the user directly specifies the allocation of each cell to a particular processor.

A set of subdirectories has been created by the decomposition utility, one for each processor, inside the case folder. The directories are named *processor* – where $N = 0,1,2,\ldots$ – they represent a processor number and contain a time directory, including the decomposed solid and fluid field description and the decomposed mesh description for both regions.

5.2 *ISATLib* in parallel `catalyticFOAM`

The simulation runs in parallel on separate subdomains and the communication between the processors occurs through the MPI protocol. Each subdomain is linked to a `catalyticFOAM` solver instance which creates an *ISATLib* instance.

In the current `catalyticFOAM`-*ISATLib* implementation, the *purely local processing* (*PLP*) [7] strategy has been adopted. This means that the *PLP*-*ISATLib* uses only the local ISAT table on each processor.

The main features of *PLP*-*ISATLib* are the following:

- No messages passing: the local ISAT table depends on the local query distribution
- Load imbalance is possible due to non-uniform intensity of chemical reactions or non-uniform distribution of computational particles.

One reason that could lead to a load imbalance is the inhomogeneous distribution of the catalytic cells along the reactor. This because particles are primarily assigned to processors based on their position in the physical coordinate space, rather than any chemical properties. Therefore, it is advantageous to define the best decomposition strategy to guarantee that each processor works with the same chemical activity, avoiding load imbalance. This procedure is not easy at all, because of two principal reasons: the first one is the transient behavior of the catalytic bed which determines that a region can strongly modify its chemical properties during the simulation. The second reason is the random generation of the packed bed, which causes a difficult splitting of the mesh in subdomains owing a similar number of cells and with the same chemical activity.

In general, for homogeneous architectures, i.e. identical processors (as used for all parallel computations performed with `catalyticFOAM`), an effective load balancing

can be achieved by assigning to each processor a similar number of cells. This is certainly true for what regards the transport step, but for the chemical one only an equal subdivision of the cells is not sufficient to guarantee load balance.

For example, considering the simple annular reactor described in Section 4.3.1, the decomposition may be performed subdividing the mesh in equal sections along the axial coordinate. In this situation the load of each processor is quite similar in respect to the number of computational cells, thus the load balance in the transport phase is ensured. The chemistry of the system, as shown in Section 4.3.2, is concentrated in the first part of the reactor, thus a load imbalance for the chemical step is possible: the first processor has to front a strong chemistry and the others a lower one. In this situation, the ISAT table of the first processor will show earlier a larger number of additions and growths with respect to the other ones due to the stronger chemistry, thus the overall performance of the system will decrease. Figure 5.8 shows the profiles of additions for `simple` decomposition method in the annular reactor for a hydrogen combustion. The computational domain is split in 4 subdomains. As expected the number of additions of the first CPU is the largest determining a light load imbalance.

Better processor loading performances are achieved using the *scotch* decomposition, because it minimizes the processors boundaries. Thus, the *scotch* decomposition will be used in the simulation of industrial packed bed reactors.

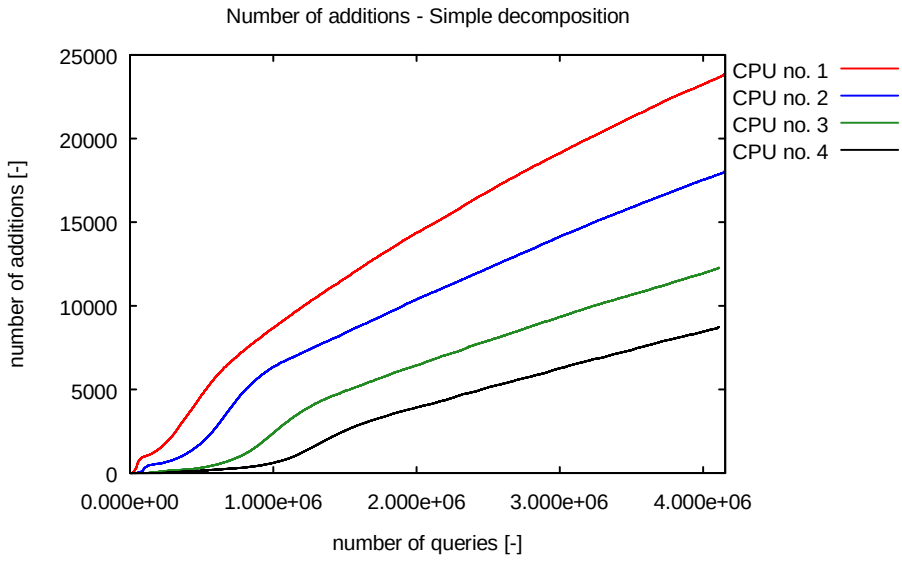


Figure 5.8 – Number of addition respect to CPU number

Further improvements of the *ISATLib* code might be the implementation of a redistribution of the query in order to use not only the ISAT table on a certain processor but also the records stored on the other processors. As suggested by Pope et al. [7], one possible solution is a parallel strategy which adopts a multi-stage strategy: first the query is attempted to be solved on the local ISAT table; the particles unresolved are randomly distributed uniformly among all the processors and resolved either by retrieves or by function evaluations. This improvements might be matter of future works of Thesis.

5.3 Showcase simulations: Steam Reforming

Syngas, i.e. a mixture of hydrogen and carbon monoxide, can be produced by steam treatment of natural gas or light petroleum fractions as naphtha. Syngas is used in industrial synthesis of ammonia and methanol. At high temperatures (700-1100 °C), in presence of a Nickel-based catalyst, methane reacts with steam to yield syngas. The reaction is strongly endothermic and requires external heating provided in cathedral furnaces where the tubes are inserted.

The main reaction is:

$$CH_4 + H_2O \rightarrow CO + 3H_2$$

Additional hydrogen can be recovered by water-gas shift reaction:

$$CO + H_2O \rightarrow CO_2 + H_2$$

The syngas obtained presents a H_2/CO ratio of 3:1 which is required by ammonia synthesis. If necessary some carbon dioxide can be added to the initial charge in order to modify that ratio satisfying further production requirements, as occurs in methanol synthesis, where the ideal ratio is 2:1.

The reactor works under a pressure of 14-40 bar and with a high flow velocity. The initial charge presents a ratio of 5 between steam and methane, in order to reduce the coke formation and favoring its gasification. The tube length is, usually, 8 m and the tube internal diameter is 0.8 m.

The simulations are performed using a detailed microkinetics mechanism developed for steam reforming by Deutchmann et al. [38]. It consists of 42 reactions among 7 gas phase and 13 surface species.

The reaction mechanism suggests that adsorbed carbon species $CH_x(s)$ (x = 0,1,2,3), formed from activated methane, react with adsorbed oxygen, formed from the adsorption of oxygen or from the decomposition of water and CO_2, and produce carbon oxide.

The total reforming process on Ni catalyst is described as follows:

1. Adsorption/desorption of reactants and products
2. Activation of methane without oxygen:

$$CH_4(s) + (4-x)Ni(s) \rightarrow CH_x(s) + (4-x)H(s)$$

$$CH_x(s) + xNi(s) \rightarrow C(s) + xH(s)$$

and with adsorbed oxygen:

$$CH_4(s) + (4-x)O(s) \rightarrow CH_x(s) + (4-x)OH(s)$$

$$CH_x(s) + xO(s) \rightarrow C(s) + xOH(s)$$

3. Decomposition of water:

$$H_2O(s) + 3Ni(s) \rightarrow O(s) + 2H(s)$$

and carbon dioxide:

$$CO_2(s) + 2Ni(s) \rightarrow C(s) + 2O(s)$$

4. Reaction of adsorbed species and production of CO and H2:

$$CH_x(s) + O(s) + (x-1)Ni \rightarrow CO(s) + xH(s)$$

The reaction mechanism includes the reactions of partial oxidation and steam reforming of methane and it is based on adsorbed atomic oxygen $O(s)$, that is the most common and key intermediate for these reactions.

The simulations are performed with `catalyticFOAM-multiRegion` with *`ISATLib`*, for different purposes:

- Investigate the *`ISATLib`* speed-up performance
- Analyze the effect of pellet shape on conversion

- Investigate the effect of different effective diffusivities inside the pellet

The simulations are performed in adiabatic conditions. The temperature of the reactor tube is fixed and equal to the inlet reactor value.

5.3.1 Simulation results: *ISATLib* speed-up performance

The reliability of the `catalyticFOAM-multiRegion-`*ISATLib* solver has been demonstrated in Chapter 4.

The performances of the algorithm are investigated using a complex geometry showcase, representative of a section of an industrial steam reforming reactor tube.

The beds are built-up using 18 rings – Large-Hole or first type – and 20 pellets "7 Holes".

The mesh with rings is built-up using a high level of refinement in order to obtain a high quality pellet description, whereas the "7 Holes" mesh uses a medium level of refinement.

Table 5.1 presents the dimensions of the grids

	Number of cells	
	Rings	**7 Holes**
Fluid	2,482,268	622,329
Solid	1,406,152	296,741
Total	3,888,420	919,070

Table 5.1 - Mesh number of cells

The simulations are performed in adiabatic conditions and in parallel using 12 CPUs. The initial charge is the typical feed for steam reforming operations (see Table 5.2).

OPERATING CONDITIONS

CH_4 mass fraction	0.121	**Inlet temperature**	973 K
H_2O mass fraction	0.829	**Outlet Pressure**	14 bar
N_2 mass fraction	0.009	**Flow velocity**	2.5 m/s
H_2 mass fraction	0.002	**ISAT tolerance**	$1 \cdot 10^{-4}$

Table 5.2 – Operating conditions – Steam reforming

The tube internal diameter is 0.015 m for the ring simulation and 0.040 m for the "7 Holes" simulations. The resulting bed length are 0.030 m and 0.080 m, respectively. The simulation of the whole reactor tube is unfeasible due to the enormous dimension of the mesh necessary to describe that system.

Figure 5.9 and 5.10 shows a comparison between the results of the ISAT and direct integration for the "7 Holes" pellets. As expected the maps obtained with ISAT overlap the direct ones both in the fluid and in the solid phase.

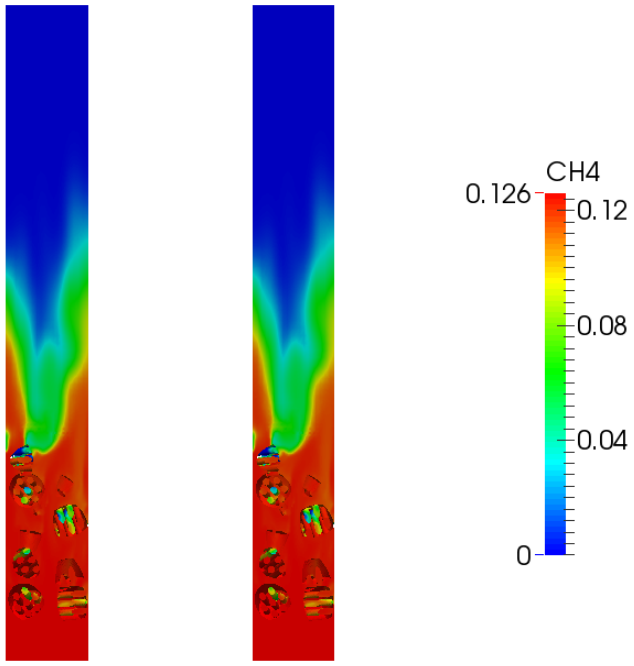
The accurate good agreement between the profiles ensures the reliability of the `catalyticFOAM-ISATLib` solvers in these complex geometries. The tolerance value used in these simulations, i.e. $1 \cdot 10^{-4}$, has been set in order to ensure numerical reliability and it might be slightly increased in order to achieve better speed-up performances without losing in accuracy.

The speed-up performances of the system are listed in Table 5.3. In these simulations is also possible to compare the speed-up of the chemistry with the overall speed-up of the simulation. The overall speed-up is evaluated computing the ratio between the average time required to perform a time step using ISAT and direct integration. The chemistry speed-up is the ratio between the average time spent to integrate the ODE system using direct integration and ISAT.

The performances in terms of speed-up the chemical step are good for both the beds: `ISATLib` reduces the time required to carry out the chemical step. However, they strongly depends on the number of cells loaded on each processor: assuming that the *scotch* decomposition allows to load a homogeneous distribution of the computational particles on each processor, the best results are achieved in the ring simulation where the number of solid cells on each processor is much higher.

	Rings	**7 Holes**
Overall Speed-up factor	3.5	1.5
Chemistry Speed-up factor	28	14
Catalytic Cell [%]	36.2	31.9

Table 5.3 – Speed-up factor



a) b)

Figure 5.9 – Comparison between ISAT a) and DI b) mass fraction profile – fluid region – 7Holes

a) b)

Figure 5.10 – Comparison between ISAT a) and DI b) site coverage profile – fluid region – 7Holes

Both simulations are performed using the same kinetic scheme, therefore the different Chemistry Speed-up factors achieved are due to the different numbers of catalytic cells loaded on each processors. The better performances are reached in the Rings simulation where the processor load is 4.7 times higher. This analysis points out that the *ISATLib* performances improve increasing the number of cells loaded on the processors. Thus, once the number of processors is fixed, *ISATLib* allows to work with higher-dimension or more accurate meshes, spending the same or less computational time. Moreover, once the mesh dimension is fixed, the ISAT implemented solver can work with a lower number of CPUs then the direct integration without losses foe what concerns the performances.

On the other hand, the performances on the overall speed-up of the simulations are not so satisfactory, as shown in Table 5.3. The main reason is that the clock time required to perform the chemical step, in these simulations, is of the same order or lower than the time required to carry out the transport step. The best ISAT performances are achieved when the chemistry is the bottleneck of the process or when a large part of the computational time is spent in that step. The percentage of catalytic cells is much higher in the Rings simulation (see Table 5.3) determining a higher simulation speed-up, because of the larger relative importance of the time required to perform the chemical step in respect to the transport step.

The overall speed-up factor has been defined as follows:

$$SF = \frac{\tau_T^{DI} + \tau_C^{DI}}{\tau_T^{ISAT} + \tau_C^{ISAT}} \tag{5.2}$$

where τ_T^{DI}, τ_C^{DI}, τ_T^{ISAT}, τ_C^{ISAT} are the times required to compute transport and chemical steps in direct integration and ISAT, respectively. Assuming that the time required to perform the transport step is the same both in direct integration and ISAT:

$$\tau_T^{DI} \cong \tau_T^{ISAT} = \tau_T \tag{5.3}$$

Substituting (5.3) in (5.2):

$$SF = \frac{\tau_T + \tau_C^{DI}}{\tau_T + \tau_C^{ISAT}} \tag{5.4}$$

Defining the ratio between the time of the chemistry step in direct integration ad ISAT:

$$SF_{chem} = \frac{\tau_C^{DI}}{\tau_C^{ISAT}} \tag{5.5}$$

It is possible to evaluate the ratio between the time spent to compute the transport and the ISAT chemistry:

$$\frac{\tau_T}{\tau_C^{ISAT}} = \frac{SF_{chem} - SF}{SF - 1} \tag{5.6}$$

In order to validate the proposed formula for the speed-factor the ratio obtained using (5.6) is compared with the one observed during the simulation. The ratio evaluated is 9.8, for the rings, and 25 for the "7 Holes".

This analysis shows that the performance of the *ISATLib* determines that the bottleneck of the simulation is not anymore the chemistry but might become the transport term. This requires, to achieve higher simulation speed-up, an improvement of the algorithm used in the transport step of the splitting method, because the chemistry it is not always the slowest step of the simulation.

For example, assuming that the time required to carry out the transport step is unitary and it is the same of the one required to solve the chemistry by direct integration, the speed-up factor, function of the ISAT chemistry time, is the following:

$$SF = \frac{2}{1 + \tau_C^{ISAT}} \tag{5.7}$$

Even if the time required to solve the chemistry with ISAT tends to zero the maximum achievable speed-up is lower than 2.

$$\lim_{\tau_C^{ISAT} \to 0} SF = \lim_{\tau_C^{ISAT} \to 0} \frac{2}{1 + \tau_C^{ISAT}} = 2 \tag{5.8}$$

Thus, the ratio between the time required to perform the transport step and the direct evaluation of the chemistry step is a key parameter to understand the possible performance of *ISATLib* solvers.

5.3.2 Simulation results: effect of pellet shape upon conversion

The effect of the pellet shape upon the reactants conversion has been investigated for a packed bed made by rings – Small-Hole or second type – and "7 Holes".

The beds are prepared in order to have the same catalyst mass: it is assumed that the two different pellets have the same density, thus the same catalyst mass is obtained using the same volume of catalyst. The number of particles in each bed and the catalyst mass loaded in the reactor are listed in Table 5.4.

	Rings	**7 Holes**
Number of pellets	14	20
Catalyst mass	$3.8 \cdot 10^{-2}$ kg	

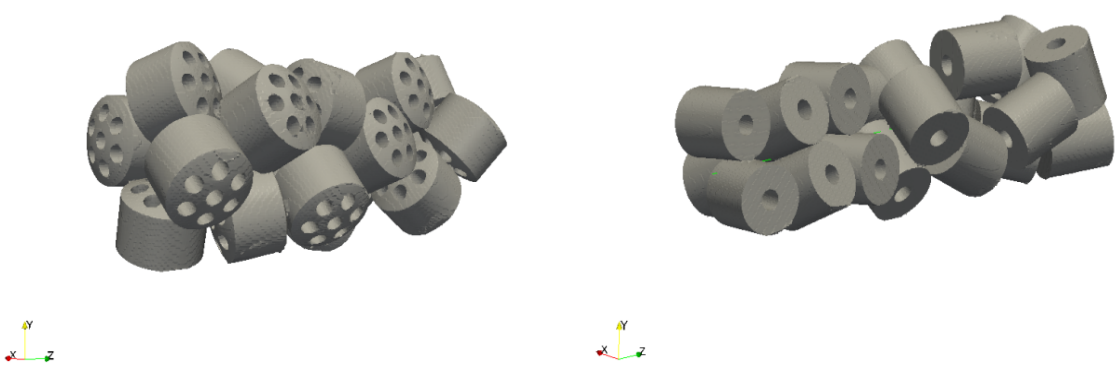
Table 5.4 – Bed dimension and catalyst mass

The operating conditions are the same listed in Table 5.2 and the simulations are carried out in adiabatic conditions. The reactor diameter is 0.04 m and it is the same for the packed beds. The packed bed length is 0.08 m. The unique difference between the catalytic beds is the pellet shape, thus it is possible to investigate the effect of the shape on the conversion.

	Number of cells	
	Rings	**7 Holes**
Fluid	518,895	622,329
Solid	320,757	296,741
Total	839,652	919,070

Table 5.5 – Mesh dimension

The dimensions of the two meshes are listed in 5.5 and they are shown Figure 5.11.



a) b)

Figure 5.11 – Mesh of the catalytic bed – a) "7 Holes", b) Rings

Figure 5.12 and 5.13 show the results of the simulation, for a longitudinal section of the reactor, for the two different pellet shapes.

The reactors work in diffusion regime with strong intra-phase transport limitation, which means that only a superficial layer of the pellet is interested in the reactions and the most part does not take part to the heterogeneous chemistry, determining a low pellet effectiveness. For this reason, the most efficient pellet shape is "7 Holes", because of its larger superficial catalytic area.

The computed conversions for the limiting reactant, i.e. methane, are listed in Table 5.6.

	Rings	**7 Holes**
Conversion	2.01 %	2.78 %

Table 5.6 – Bed conversion

The conversion of the "7 Holes" bed is larger than the one of the ring bed due to the higher superficial area, key parameter in diffusion regime. The conversion is quite low due to the shortness of the beds.

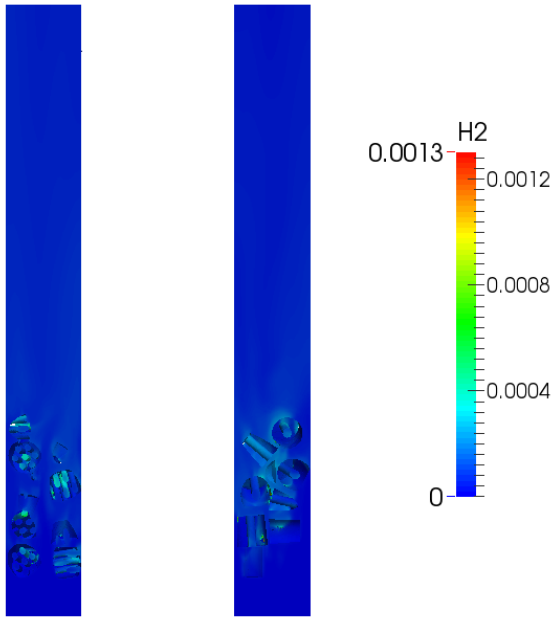


Figure 5.12 – Comparison between "7 Holes" a) and Rings b) mass fraction profile – fluid region

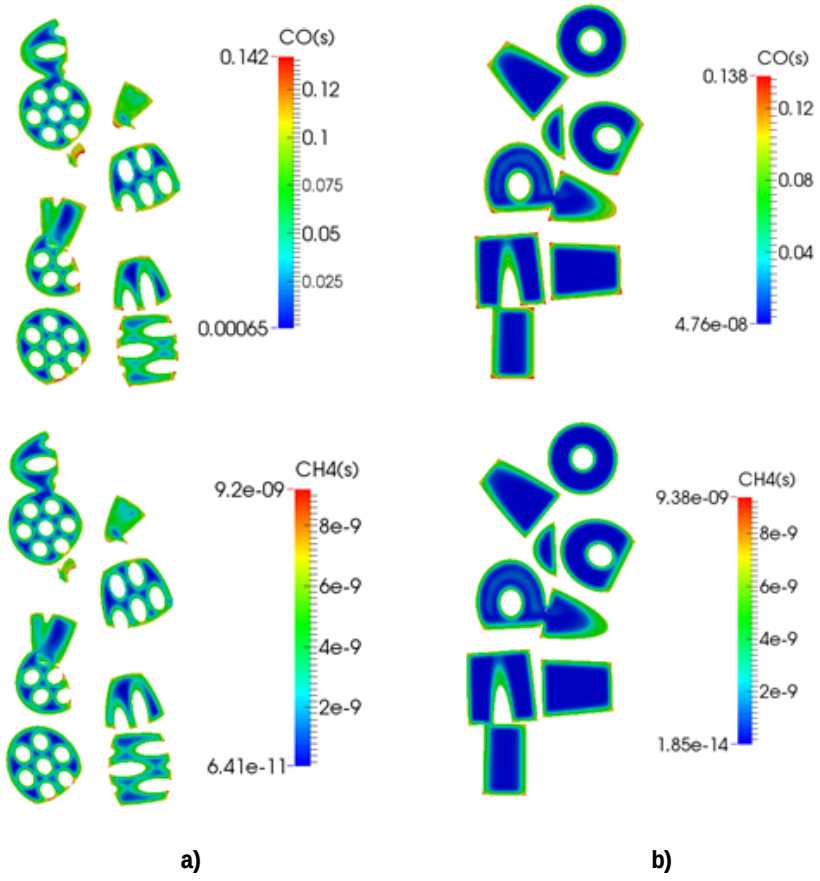


Figure 5.13 – Comparison between "7 Holes" a) and Rings b) site coverage profile – solid region

5.3.3 Simulation results: effective diffusivity effect

The effect of different values of effective diffusivity is investigated in the packed beds made by "7 Holes" pellets. The purpose of this analysis is to investigate the different behaviors of the reactor using different effective diffusivity values. The values for the effective diffusivity are selected to test the performance of the solver and not to represent a real industrial simulation.

The `catalyticFOAM-multiRegion` permits to select among three different intra-phase diffusion models [1]:

- *Reduction Coefficient*: the gas-phase diffusivity of each species is scaled by this reduction coefficient, simply multiplying the coefficient times the diffusivity. The effective diffusivity is given by the following equation:

$$\mathfrak{D}_{A,c} = CR \cdot \mathfrak{D}_{A,M} \tag{5.9}$$

- *Parallel Pore Model*, developed by Wheeler [39] : the porous material is considered to be a bundle of capillaries through the species diffuse. The volume available for diffusion is the pore volume. An additional factor, the tortuosity, is introduced to account the non-straightness of the pores which follow a tortuous path. The effective diffusivity is given by the following equation:

$$\mathfrak{D}_{A,c} = \frac{\varepsilon}{\tau} \cdot \frac{\mathfrak{D}_{A,M} \cdot \mathfrak{D}_{A,K}}{\mathfrak{D}_{A,M} + \mathfrak{D}_{A,K}} \tag{5.10}$$

  where $\mathfrak{D}_{A,M}$ is the molecular diffusivity m^2/s, $\mathfrak{D}_{A,K}$ is the Knudsen diffusivity m^2/s, ε is the void fraction of the pellet and τ is the tortuosity. Because of the difficulty in obtaining experimental measurements of tortuosity, this parameter has been considered as adaptive. The parallel pore model works better for a catalyst with an unimodal pore size distribution that has a narrow range of pore diameters

- *Random Pore Model*, developed by Wakao-Smith [40]: this model is able to take into account the micro-morphology features such as pore size, pore orientation, interconnections and dead ends. Wakao and Smith proposed this model for predicting diffusivity at constant pressure in bidisperse porous media. It represents the diffusion flux as being the sum of that through macropores and

micropores. The model gives the effective diffusivity for isobaric diffusion with the following expression :

$$\mathfrak{D}_{A,c} = \varepsilon_M^2 \cdot \mathfrak{D}_{A,M} + \frac{\varepsilon_u^2 \cdot (1 + 3\varepsilon_M)}{1 - \varepsilon_M} \cdot \mathfrak{D}_{A,u} \tag{5.11}$$

where ε_M is the porosity of the macropores and ε_u is the porosity of the micropores. The predictions of this model are usually higher than those evaluated experimentally.

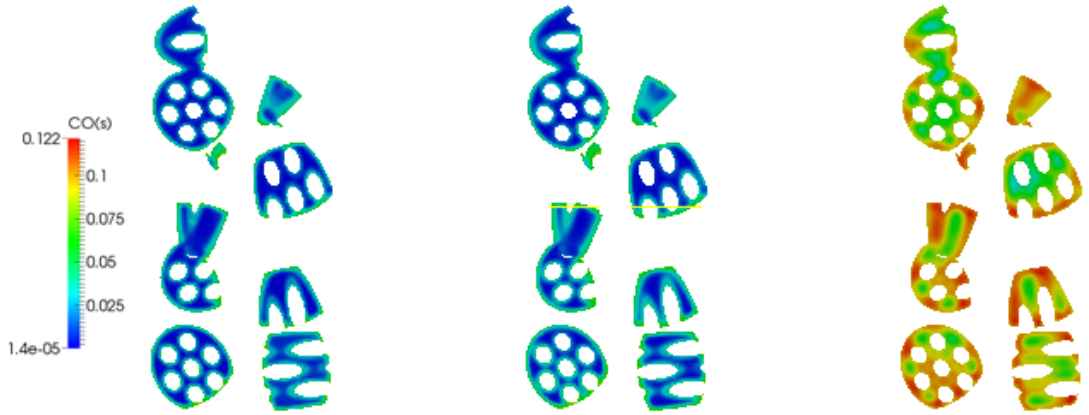
The simulation are carried out using the *Parallel Pore Model*, using the values listed in Table 5.7 for void fraction of the pellet and tortuosity. The same computational domain and the same operating conditions investigated in Section 4.3.2 have been employed in these simulations.

	Low Diff	**Med Diff**	**High Diff**
Void Fraction	0.55		
Tortuosity	10	3	1

Table 5.7 – Parallel Pore Model parameter

Figure 5.14 shows the results of the simulations: the different values of the effective diffusivity determines a different usage of the inner part of the pellet. The simulation performed with the lower value of the diffusivity shows high catalyst effectiveness due to the faster intra-phase diffusion: the reaction does not occur only on the surface but also in the inner part of the pellets. This behavior confirms that “7 Holes” catalyst work under internal mass transfer regime.

The different diffusion inside the pellets leads to a different conversion of the beds: the bed characterized by the higher diffusivity value converts much more reactants than the other two.



a) b) c)

Figure 5.14 – Comparison among Low Diff a), Med Diff b), High Diff site coverage profile – solid region

5.4 Conclusions

The reliability of the `catalyticFOAM-multiRegion` using direct integration dealing with these complex systems has been proved in previous works [1]. The capability of the `catalyticFOAM` solver with *`ISATLib`* implemented to solve large dimension meshes with complex geometries and detailed kinetic schemes has been demonstrated, confirming the good agreement between ISAT and direct integration simulation results.

One of the main issue has been the refinement of the mesh, the grid has to be strongly refined near the contact points between pellets and with the wall of the tube to improve the simulation accuracy and stability. In particular, the description of the fluid region near the pellets surface has to have a very high level of detail in order to describe the mass and heat transfer between the fluid and the solid region. The description on the solid phase requires a finer mesh in the region near the catalytic surface. For this reason the number of cells required to describe the "7 Holes" bed is higher than the one required to represent the ring bed, using the same level of refinement.

As demonstrated in Section 5.3.1, if the transport step requires a clock time comparable to the one of the chemical step, the speed-up achieved by ISAT in the chemistry step has

not a strong influence on the total computational time. Moreover, the ratio between the time required the transport and the chemical step is a key parameter to forecast the performance of the *ISATLib* solvers. The performances are lower than the ones experienced in Section 3, due to the higher number of cells in the fluid region, that determines a considerable time spent performing the transport step.

Although, the capability of speeding-up the chemical step has been demonstrated. The Chemical Speed-factor values are higher than the one obtained in Chapter 3, due to the large number of cells loaded on each processor.

Therefore, the accurate set-up of the mesh and its decomposition are two issues of utmost importance to reach the best *ISATLib* performance.

In conclusion, this analysis shows the capability of the *ISATLib* solver to deal with industrial relevant cases determining an overall simulation speed-up which allows to use larger packed beds, reaching the purpose of pushing the frontier of feasible simulations with the `catalyticFOAM` framework.

6 Conclusions

In this work for the first time, ISAT has been introduced in a CFD simulation able to perform the dynamic simulation of heterogeneous catalytic reactors. In particular, an improvement of the `catalyticFOAM` solvers performances has been performed to reduce the computational time and to allow its successful and efficient application to complex geometries, large dimension industrial cases with detailed kinetic mechanism. The main purpose is to decrease the computational effort required in the solution of the chemical step, highlighted in previous works as the bottle-neck of the whole simulation process.

A library, named *`ISATLib`*, has been developed in order to implement the *in situ* adaptive tabulation algorithm. The architecture of the library requires the management of a storage and retrieval structure; this feature is provided through a Binary Tree data structure. Moreover, some improvements to the original algorithm are implemented in the library, as the linked search lists, in order to increase the capability of successful retrieves within the storage table. A procedure to control the efficiency of the Binary Tree has been implemented to improve the overall performance.

The *`ISATLib`* has been implemented in both `catalyticFOAM` and `catalyticFOAM-multiRegion` solvers.

Moreover, the library has been left intentionally general to be implemented in a wide field of applications. Considering the application in the `catalyticFOAM` framework, the user can select the ODEs solver that best suit the problem without any modification in the library.

The implementation in the solvers framework has fronted three main issues:

1. The implementation of the library in the solvers requires the initialization of two different instances: the first for the homogeneous chemistry and the second for the heterogeneous one. Each instance sets up an empty Binary Tree which will be filled with a certain number of records only if necessary. It is not possible to

use a single table to treat both reactive systems due to the different dimensions of the problems.

2. The methods necessary to direct evaluate the function, already inside in the solvers, are used to obtain the information needed to the ISAT algorithm to settle up the table. The evaluation of the mapping gradient matrix requires the Jacobian of the chemistry ODEs system. A function has been inserted in the solver framework in order to carry out its the numerical evaluation.
3. The composition vector returned by *ISATLib* can contain some negative values due to the interpolation procedure of near zero value of highly reactive species. The unfeasible negative mass fractions are set to zero and the vector is normalized to obtain a unitary sum. This process does not affect the accuracy of the method because the absolute value of the negative elements of the composition vector is very small.

The reliability and the predictive capabilities of the modified solvers have been tested by comparing the numerical results of ISAT simulation to the results obtained by direct integration in different operating conditions (composition, temperature, pressure), in different configurations (annular reactor, catalytic gauze, packed bed) and with different heterogeneous and homogeneous kinetic mechanisms (H_2/O_2 on Rh/Al_2O_3, CH_4/O_2 on Rh/Al_2O_3, CH_4/O_2 on Pt, CH_4/H_2O on Ni). In all investigated conditions, the results obtained with *ISATLib* have shown an excellent agreement with the one provided by the direct integration.

The performances of the `catalyticFOAM-`*ISATLib* solvers have been considered as the capability of speeding-up the whole simulation. The results have shown an overall speed up from 1.5 to 9.5 times depending on the dimension of both the mesh and the kinetic schemes: the larger is the number of the heterogeneous cells in respect to the fluid transport, the better performances are achieved. This occurs because in these situations the chemistry is the real bottle-neck of the simulation. Furthermore, the time required to perform the direct integration is strongly affected by the number of both species and reactions involved. On the other hand the time required to carry out an interpolation depends weakly on the dimension of the scheme ensuring higher speed-up in more complex cases.

The performances of the `catalyticFOAM-`*`ISATLib`* solvers have been evaluated also by means of the chemistry speed-up factor. This analysis has shown a very satisfactory capability of the *`ISATLib`* to improve the computational time of the chemistry step, also 5-20 times lower than the computational time required by direct integration to solve the chemistry (at steady state conditions).

A key parameter which strongly affects the performances and the reliability is the tolerance: a correct value allows accurate results in a smaller computational time. The speed-up of the simulation increase enlarging the tolerance: the region of accuracy of each leaf is larger and the probability of a successful retrieve, leading to fast interpolation procedure, are higher. On the other hand, the accuracy of the method follows the opposite trend: a low tolerance value allows better agreement due to the lower acceptable approximation error.

The correct tolerance value has to be set as a compromise between these opposite behavior and, depending on the problem, the user can give more importance to the accuracy or to the speed-up.

Finally, the `catalyticFOAM-`*`ISATLib`*`-multiRegion` solver has been used to test relevant industrial cases. Several catalytic packed bed reactors for the methane steam reforming have been simulated, through the parallel version of the solver, during an internship period in BASF. The catalytic beds have been set up using a computational tool based on *DEM* methodology. The results of the simulation highlighted the capability of the *`ISATLib`* solvers to reduce the computational time required to perform the whole simulation. The performances, for the multiRegion parallel approach, strongly depend on the ratio between the number of solid and the number of fluid cells . The lower is this ratio, the more important is the transport term. As consequence of this fact, the resolution of the transport term can become the limiting factor of the computational time leading to a decrease of the potential speed-up factor.

Future developments will be focused on the extension of the range of applicability of the solver. In particular, the following objective can be highlighted:

- implementations of strategies to improve the speed-up of the transport step, in particular in the parallel version of the multiRegion approach, because it has

been identified, in some simulations, as the limiting step of the computational time;

- development of full steady-state solver in order to neglect the transient behavior directly reaching the simulation steady state. The steady state requires a good initial point in order to reach a solution. This initial value can be obtained by a dynamic simulation speeded-up by *ISATLib*.

List of figures

List of Tables

Bibliography

[1] Gentile, G., Manelli, F., *An efficient computational framework for the advanced modeling and design of industrial catalytic reactors*. 2013.

[2] Goisis, S., Osio, A., *Computational fluid dynamics of gas-solid catalytic reactors based on microkinetic description of surface chemistry*. 2011.

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

[4] OpenFOAM, *The Open Source CFD Toolbox - User Guide*. 2013.

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

[6] Kumar, A. and S. Mazumder, *Adaptation and application of the in situ Adaptive Tabulation (ISAT) procedure to reacting flow calculations with complex surface chemistry.* Computers & Chemical Engineering, 2011. **35**(7): p. 1317-1327.

[7] Lu, L., et al., *Computationally efficient implementation of combustion chemistry in parallel PDF calculations.* Journal of Computational Physics, 2009. **228**(15): p. 5490-5525.

[8] Lu, L. and S.B. Pope, *An improved algorithm for in situ adaptive tabulation.* Journal of Computational Physics, 2009. **228**(2): p. 361-386.

[9] Pope, S.B. *ISAT-CK7: Fortran 90 libraries for the efficient implementation of combustion chemistry*. Available from: https://tcg.mae.cornell.edu/ISATCK7/.

[10] Xu, J., Pope, S.B., *PDF Calculations of Turbulent Nonpremixed Flames with Local Extinction.* Combustion and Flame, 2000. **123**: p. 281–307.

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

[12] Kapoor, R., Lentati, A.,Menon, S., *Simulations of Methane–Air Flames using ISAT and ANN*, in *37th AIAA/ASME/SAW/ASEE Joint Propulsion Conference*, AIAA, Editor. 2001. p. 2001–3847.

[13] Heyea, C., Ramana, V., Masrib, A. R., *Influence of spray/combustion interactions on auto-ignition of methanol spray flames.* Proceedings of the Combustion Institute.

[14] Irannejad, A., Banaeizadeh, A., Jaberi, F., *Large eddy simulation of turbulent spray combustion.* Proceedings of the Combustion Institute.

[15] Tang, Q., *Computational modelling of turbulent combustion with detailed chemistry*. 2003, Cornell University.

[16] Wang, L., Fox R. O., *Application of in situ adaptive tabulation to CFD simulation of nano-particle formation by reactive precipitation.* Chemical Engineering Science, 2003. **58**(19): p. 4387-4401.

[17] Singer, M.A. and S.B. Pope, *Exploiting ISAT to solve the reaction-diffusion equation.* Combustion Theory and Modelling, 2004. **8**(2): p. 361-383.

[18] Singer, M.A., S.B. Pope, and H.N. Najm, *Operator-splitting with ISAT to model reacting flow with detailed chemistry.* Combustion Theory and Modelling, 2006. **10**(2): p. 199-217.

[19] Hedengren, J.D. and T.F. Edgar, *Approximate nonlinear model predictive control with in situ adaptive tabulation.* Computers & Chemical Engineering, 2008. **32**(4-5): p. 706-714.

[20] Hedengren, J.D. and T.F. Edgar, *Order reduction of large scale DAE models.* Computers & Chemical Engineering, 2005. **29**(10): p. 2069-2077.

[21] Pope, S.B., *Algorithms for Ellipsoids*, in *FDA 08-01*. 2008, Cornell University.

[22] Contino, F., et al., *Coupling of in situ adaptive tabulation and dynamic adaptive chemistry: An effective method for solving combustion in engine simulations.* Proceedings of the Combustion Institute, 2011. **33**: p. 3057-3064.

[23] Eigen. *Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.* Available from: http://eigen.tuxfamily.org/index.php?title=Main_Page.

[24] Maestri, M., et al., *A C-1 Microkinetic Model for Methane Conversion to Syngas on Rh/Al2O3.* Aiche Journal, 2009. **55**(4): p. 993-1008.

[25] Kazakov, A. and M. Frenklach. *Reduced Reaction Sets Based on GRI-Mech 1.2*. Available from: http://www.me.berkeley.edu/drm/.

[26] Sportisse, B., G. Bencteux, and P. Plion, *Method of Lines versus Operator Splitting for reaction-diffusiuon systems with fast chemistry.* Environmental Modelling & Software, 2000. **15**(6-7): p. 673-679.

[27] Issa, R.I., *Solution of the implicitly discretized fluid-flow equations by operator-splitting.* Journal of Computational Physics, 1986. **62**(1): p. 40-65.

[28] Quiceno, R., et al., *Modeling the high-temperature catalytic partial oxidation of methane over platinum gauze: Detailed gas-phase and surface chemistries coupled with 3D flow field simulations.* Applied Catalysis a-General, 2006. **303**(2): p. 166-176.

[29] Kruggel-Emden, H., et al., *Selection of an appropriate time integration scheme for the discrete element method (DEM).* Computers & Chemical Engineering, 2008. **32**(10): p. 2263-2279.

[30] Diethelm, K., N. Ford, and A. Freed, *A Predictor-Corrector Approach for the Numerical Solution of Fractional Differential Equations.* Nonlinear Dynamics, 2002. **29**(1-4): p. 3-22.

[31] SDEC. *An open–source software using a discrete element method to simulate granular material*. Available from: http://geo.hmg.inpg.fr/frederic/Research_project_Discrete_Element_Software.html.

[32] Yade. *An extensible open-source framework for discrete numerical models, focused on Discrete Element Method*. Available from: https://www.yade-dem.org/doc/.

[33] LAMMPS. *Molecular Dynamics Simulator*. Available from: http://lammps.sandia.gov/.

[34] LIGGGTHS. *Open Source Discrete Element Method Particle Simulation Code*. Available from: http://cfdem.dcs-computing.com/?q=OpenSourceDEM.

[35] Zhu, H.P., et al., *Effect of particle properties on particle percolation behaviour in a packed bed.* Minerals Engineering, 2009. **22**(11): p. 961-969.

[36] *Matros Technologies. Inc*. Available from: http://www.matrostech.com.

[37] Pellegrini, F. and J. Roman, *Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs*, in *High-Performance Computing and Networking*, H. Liddell, et al., Editors. 1996, Springer Berlin Heidelberg. p. 493-498.

[38] Maier, L., et al., *Steam Reforming of Methane Over Nickel: Development of a Multi-Step Surface Reaction Mechanism.* Topics in Catalysis, 2011. **54**(13-15): p. 845-858.

[39] Wheeler, A., Catalysis Vol. II, 1955.

[40] Wakao. W, S., J.M., Chem. Eng. Sci., 1962. **17**.