Taylor & Francis
Taylor & Francis Group

Combustion Theory and Modelling

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tctm20

A dynamic load balancing model coupled with DAC and ISAT for a stochastic turbulent combustion model

Zhijie Huo, Matthew J. Cleary, Kun Wu, Assaad R. Masri & Xuejun Fan

To cite this article: Zhijie Huo, Matthew J. Cleary, Kun Wu, Assaad R. Masri & Xuejun Fan (2023) A dynamic load balancing model coupled with DAC and ISAT for a stochastic turbulent combustion model, Combustion Theory and Modelling, 27:3, 317-345, DOI: 10.1080/13647830.2023.2165967

To link to this article: https://doi.org/10.1080/13647830.2023.2165967

Combustion Theory and Modelling, 2023
Vol. 27, No. 3, 317–345, https://doi.org/10.1080/13647830.2023.2165967

A dynamic load balancing model coupled with DAC and ISAT for a stochastic turbulent combustion model

Zhijie Huo[a,b], Matthew J. Cleary [b,c], Kun Wu[a*], Assaad R. Masri[b] and Xuejun Fan[a,d]

[a]*State Key Laboratory of High Temperature Gas Dynamics, Institute of Mechanics, Chinese Academy of Sciences, Beijing, People's Republic of China;* [b]*School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Sydney, NSW, Australia;* [c]*ARC ITTC Data Analytics for Resources and Environments, The University of Sydney, Sydney NSW, Australia;* [d]*School of Engineering Science, University of Chinese Academy of Sciences, Beijing, People's Republic of China*

(*Received 14 August 2022; accepted 22 December 2022*)

Due to the composition-dependent stiffness of chemistry, simulations of reactive turbulent flows may present computational load imbalance among parallel processes when spatial decomposition is used for parallelisation, causing high CPU idle time and waste of computational resources. To increase computational efficiency, a dynamic load balancing (DLB) model is proposed to redistribute computational load among computing cores. The DLB model exploits a decomposition in the mixture fraction space with two dynamic adjusting decomposition strategies to realise load redistribution. The DLB model is suitable for the integration of chemistry on stochastic particles in hybrid Eulerian/Lagrangian turbulent combustion models in which the Eulerian field is conventionally decomposed statically in physical space in a way that balances the computational load for the solution of the Navier-Stokes equation but which does not generally lead to balanced load for the computation of the composition fields. Here it is tested using an OpenFOAM-based platform, *mmcFoam*, which is a comprehensive object-orientated C++ library for stochastic turbulent combustion modelling. Apart from direct integration (DI) for chemistry, the DLB model is also coupled with dynamic adaptive chemistry (DAC) and *in situ* adaptive tabulation (ISAT), which allows for extra speedup. The performance of the coupled models is validated and assessed for two laboratory flame conditions that exhibit different levels of computational load imbalance. Overall, the DLB model effectively balances the computational load distribution and increases the effective usage of computing power, shortening the simulation wall time required. Moreover, a strong scaling test is carried out using up to 512 cores. Although all approaches have sub-ideal scalability, the scalability of each with DLB is significantly better than without DLB. While DLB-ISAT has relatively poor scalability compared to the DI- and DAC-based methods, DLB-ISAT still ranks the fastest among the algorithms in all scaling trials.

Keywords: dynamic load balancing; DAC; ISAT; turbulent combustion; MMC-LES

1. Introduction

Turbulent reactive flows, which are common in practical combustion systems, can be strongly affected by nonlinear turbulence-chemistry interactions (TCI) which can disturb the performance of combustors. Turbulence involves a wide span of scales, and resolving

*Corresponding author. Email: wukun@imech.ac.cn

all the scales by direct numerical simulation (DNS) is too computationally expensive for practical use. Therefore, the temporal averaging and spatial filtering methods, i.e. the Reynolds-averaged Navier-Stokes (RANS) and large eddy simulation (LES), respectively, are often employed to achieve computationally affordable simulations. However, the TCI occurs in the unresolved sub-grid scales, and the highly nonlinear chemical source term becomes unclosed.

Probability density function (PDF) models [1–3] have a significant advantage over other turbulent combustion models, e.g. the flamelet [4] and conditional moment closure (CMC) [5] models, in that the nonlinear chemical source term in the transport equations appears in closed form without simplification, and represents the most general turbulent combustion model that is methodologically applicable to different combustion regimes. The stochastic multiple mapping conditioning (MMC) [6, 7] is a PDF-type model with an MMC mixing model enforcing localness in a combined physical and reference variable space. In the last decade, the MMC has evolved and resulted in the MMC-LES model being validated against a wide range of characteristics and physics [8–14]. To reduce the computational cost, these methods commonly replace the Eulerian PDF transport equations with the equivalent stochastic differential equations (SDEs), forming a hybrid Eulerian–Lagrangian approach in which the flow fields are solved by standard finite volume/difference methods while the reactive composition fields are represented by the notional particles governed by the SDE.

Solving the Lagrangian field involves a fractional reacting step that computes the non-equilibrium chemical source term on each notional particle. The chemistry is described by the coupled stiff ordinary differential equations (ODEs), and direct integration (DI) of the ODEs by stiff ODE solvers such as the Runge–Kutta or Rosenbrock methods [15] is computationally expensive. In practice, the computational cost of updating chemistry scales with respect to the size and stiffness of ODEs and accounts for a majority of computational cost for reactive flow simulations. This issue has motivated the development of expediting algorithms for chemistry such as *in situ* adaptive tabulation (ISAT) [16], which stores chemical calculation history and reuses it, and dynamic adaptive chemistry (DAC) [17], which locally reduces the size of chemical kinetics during runtime.

Parallel computing has become a routine technique, and it is desired to split a problem into parts with the same computational load. However, due to the inhomogeneity of composition fields and the composition-dependent stiffness of chemical ODEs, spatial partitioning of the computational domain with an equal number of cells and particles per processor may result in an unbalanced load distribution, which causes increased CPU idle time and wastes computational resources. Dynamic load balancing (DLB) methods for pure Eulerian solvers exist that estimate weights locally and then update partitioning boundaries based on the weights [18, 19]. However, domain decomposition is a non-trivial operation, and the performance may suffer if the load redistribution is applied too frequently. Moreover, it may encounter a conflict of load balancing between the Eulerian and Lagrangian modules in hybrid solvers since the computational cost scales differently for Eulerian and Lagrangian fields.

Instead of dynamic domain partitioning, a different DLB strategy employs parallel communication to redistribute the computational load. The method transfers Lagrangian workloads between CPU cores to reach a balanced load distribution and thus demands heavy data exchange. The performance of different DLB models is strongly influenced by the hardware architecture, i.e. shared or distributed memory. On the one hand, parallel communication via OpenMP presents trivial wall time in shared memory systems,

but it is limited by the hardware and not suitable for large-scale parallelisation. On the other hand, the distributed memory system is more prevalent in high-performance computing (HPC) infrastructure, but data transfer via Message Passing Interface (MPI) can be time-consuming.

Sitaraman and Grout [20] employed a redistributing approach to balance computational load via MPI and proposed an equalisation algorithm to reduce data exchange. A satisfactory speedup was achieved; however, their method focused on non-reactive flow and only balanced the number of particles. The shared memory architecture was explored by Amritkar et al. [21] who enforced an evenly distributed Lagrangian workload via OpenMP and reported appreciable improvements compared to a distributed memory system using MPI. Thari et al. [22] proposed asynchronous task-based parallelism for shared memory systems and achieved a reduction in computational time by 62%. Hybrid systems were exploited to take advantage of both shared and distributed memory communication. Yakubov et al. [23] employed a hybrid MPI-openMP method in which the computational load of Lagrangian fields is balanced via OpenMP. Houzeaux et al. [24] proposed a multicode approach with a similar hybrid MPI-openMP method, introducing an asynchronous model with load balancing. These openMP-based DLB strategies for Lagrangian fields significantly reduced the communication time between computing cores compared to MPI, but their applicability is constrained by HPC architecture, and they are usually less scalable than MPI implementation. Moreover, the past methods focus on balancing the computational load for particle convection, which is not a major concern in stochastic turbulent combustion modelling; therefore, a suitable DLB strategy for calculating chemistry is needed.

In the reacting fractional step, chemistry evolution can be fully described by the ODEs which are only dependent on the instantaneous composition fields regardless of physical coordinates, featuring a convenience for parallelisation. Tekgül et al. [25] and Muela et al. [26] employed a similar strategy in that the cell thermochemical states are packaged and shared between overloaded and underloaded processors, and then the updated thermochemical states are returned to the original cells via MPI. Inspiring improvements in computational efficiency were reported in their deterministic Eulerian solvers. In the load redistribution methods, the computational cost of solving chemistry needs to be predicted precisely so that the workload can be redistributed evenly. However, the computational cost is related to the stiffness of the ODEs and differs significantly in a nonlinear fashion. Therefore, predicting the computational cost of chemical reactions for stochastic particles is challenging. The questions about how many and which particles need to be transferred remain unanswered, not to mention the more complicated situation when the solver is coupled with DAC and ISAT.

In the present work, we propose a novel DLB model which employs conditioning and which is suitable for stochastic combustion models in the context of hybrid Eulerian/Lagrangian schemes. Two different strategies for dynamically adjusting the load balance are investigated and a hybrid of the two permits efficient coupling with ISAT to further speed up the calculations. The DLB models are examined by simulations of two laboratory turbulent flame configurations that have been the focus of many previous modelling publications, and detailed performance investigations are presented. The remainder of this paper is arranged as follows. Section 2 presents the models for turbulent combustion and the algorithms for DAC and ISAT. Section 3 introduces the details of the DLB model. In Section 4, the numerical implementation of the models is presented. Experimental setup

and numerical configurations of test cases, as well as results and discussion, are shown in Section 5. Lastly, conclusions are drawn in Section 6.

2. Numerical models

2.1. *MMC-LES*

MMC-LES is a hybrid model [7, 27] that solves the flow field by conventional Eulerian LES and the composition field by a Lagrangian scheme. The Eulerian module solves the filtered equations for mass, momentum, pressure and reference mixture fraction, $\widetilde{f}$, which is used for mixing closure. The composition field is represented probabilistically by the filtered density function (FDF) [2] and the evolution is governed by the stochastic differential equations (SDEs)

$$\mathrm{d}x_i^p = \left[\widetilde{u}_i + \frac{\partial}{\partial x_i}(D + D_t)\right]^p \mathrm{d}t + \left[\sqrt{2(D + D_t)}\right]^p \mathrm{d}w_i, \tag{1}$$

$$\mathrm{d}\phi_\alpha^p = \left(S_\alpha^p + W_\alpha^p\right)\mathrm{d}t, \tag{2}$$

where D and D_t are the molecular and turbulent diffusivity, respectively; superscript ^{p} represents quantities carried by particles or interpolated at particle locations; dw_i is an independent Wiener process; $W_\alpha^p = W_\alpha(\boldsymbol{\phi}^p)$ is the chemical source term, and S_α^p is the mixing operator that models molecular mixing. MMC enforces mixing to be local in the combined physical and reference space so that the conditional mean is conserved, i.e. $\langle S_\alpha | \boldsymbol{x}, \widetilde{f}\rangle = 0$ [6, 7]. Localness of mixing is achieved by pairing particles that are close in the $(\boldsymbol{x}, \widetilde{f})$-space, and then paired particles mix linearly as

$$\phi_\alpha^p(t + \Delta t) = \phi_\alpha^p(t) + \gamma_m(\overline{\phi}_\alpha^{p,q}(t) - \phi_\alpha^p(t)), \tag{3}$$

$$\phi_\alpha^q(t + \Delta t) = \phi_\alpha^q(t) + \gamma_m(\overline{\phi}_\alpha^{p,q}(t) - \phi_\alpha^q(t)), \tag{4}$$

where $\gamma_m = 1 - exp(-\Delta t/\tau_m^{p,q})$ is the mixing extent and $\tau_m^{p,q}$ is the mixing time scale modelled by the C&K model [7] or a-ISO model [28].

2.2. *Expediting algorithms*

2.2.1. *Dynamic adaptive chemistry (DAC)*

The DAC method [17] aims to locally (i.e. on each notional particle) and dynamically (i.e. temporally) reduce the number of chemically active species, thus reducing the size of coupled systems of ODEs and the computational cost of integrating them. It uses a reduction algorithm based on the direct relation graph with error propagation (DRGEP) concept [29].

For a chemical kinetics, species are coupled through reactions, and the strength of the kinetic relationships between species can be defined by a contribution matrix

$$r_{AB} = \frac{|\sum_{i=1}^{n_r} \nu_{i,A} W_i \delta_B^i|}{\max(P_A, C_A)}, \tag{5}$$

where

$$P_A = \sum_{i=1}^{n_r} \max(0, \nu_{i,A} W_i), \tag{6}$$

$$C_A = \sum_{i=1}^{n_r} \max(0, -\nu_{i,A} W_i). \tag{7}$$

In the above, n_r is the number of reactions, $\nu_{i,A}$ is the stoichiometric coefficient of species A in the ith reaction, W_i is the reaction rate and δ_B^i equals one if the ith reaction involves species B and otherwise equals to zero. The relationship between species A and B can be considered negligible if $r_{AB} < \varepsilon_{DAC}$, where ε_{DAC} is a specified DAC error tolerance. Species that are eliminated are chemically frozen but still participate in three-body reactions.

To take advantage of the possible additional dimension reduction, species that are far away from the target species are also trimmed. The dependency of a secondary species P from primary species P_0 is quantified by

$$R_{P_0}(P) = \max_{\Omega} \left(\prod r_{ij} \right) \tag{8}$$

where Ω is all possible paths from P_0 to P and r_{ij} is defined by Equation (5). Species with $R_{P_0}(P) < \varepsilon_{DAC}$ are also marked inactive with relevant reactions eliminated. In practice, multiple primary species, called the search initial set (SIS), can be specified by the user depending on the aim of the modelling.

2.2.2. *In-situ adaptive tabulation (ISAT)*

ISAT stores the chemical states and reaction mapping computed by direct integration (DI) and, when possible, extrapolates them to approximate later variations in the chemical state. A general introduction to ISAT is provided below and complete details can be found in [16, 30–32].

For an initial thermochemical state $\boldsymbol{\varphi}(t_0) = \boldsymbol{\varphi}^0$, the reacted thermochemical state for a fixed timestep is a unique function of $\boldsymbol{\varphi}^0$ expressed as $\boldsymbol{\varphi}(t_0 + \Delta t) = \boldsymbol{R}(\boldsymbol{\varphi}^0)$ which is called the reaction mapping. If a query point $\boldsymbol{\varphi}^q$ (i.e. a composition point whose reaction mapping is required) is similar to the $\boldsymbol{\varphi}^0$ that are stored on leaves, DI for $\boldsymbol{\varphi}^q(t_0 + \Delta t)$ is replaced by the linear approximation

$$\boldsymbol{R}(\boldsymbol{\varphi}^q) \approx \boldsymbol{R}^l(\boldsymbol{\varphi}^q), \tag{9}$$

and compositions satisfying

$$\varepsilon_{local} = |\boldsymbol{R}(\boldsymbol{\varphi}^q) - \boldsymbol{R}^l(\boldsymbol{\varphi}^q)| \leq \varepsilon_{ISAT} \tag{10}$$

defines an ellipsoid of accuracy (EOA) that controls the following operation at each timestep.

Retrieval: The leaf with the most similar composition to a query point is found. If the query point, $\boldsymbol{\varphi}^q$, is in the EOA, then the data is retrieved and the new composition, $\boldsymbol{R}(\boldsymbol{\varphi}^q)$, is obtained by Equation (9) instead of conducting DI. Move to the next query point.

Direct integration: If $\boldsymbol{\varphi}^q$ is not in the EOA, then $\boldsymbol{R}(\boldsymbol{\varphi}^q)$ is obtained by DI and ε_{local} is calculated. Depending on the local error, move to either the growth or add steps.

Growth: If $\varepsilon_{local} \leq \varepsilon_{ISAT}$, the current EOA is deemed as being too conservative and the leaf is grown according to the minimum volume rule [31] to include the current query point. Move to the next query point.

Add: If $\varepsilon_{local} > \varepsilon_{ISAT}$, a new leaf is added to the tree for possible use in later retrievals. Move to the next query point.

Removal: To reduce memory usage leaves are removed from the tree if they have not been accessed in the last N_d time steps, where N_d has a user-specified value and is set to 200 in the present work based on past experience with the flames studied here.

3. Dynamic load balancing (DLB) model

3.1. *Principles*

Parallelisation must be implemented explicitly in solvers to tackle communication between processors and must be designed to fit the computing hardware. The more prevalent distributed memory systems designate each CPU a private memory space that is uniquely accessible by itself, and computing cores need to be united via an interconnected network to enable inter-processor communication. The DLB model proposed in the present work is for computing infrastructures with distributed memory.

The composition fields are solved by notional particles that carry information about the thermochemical state. The computational cost of solving chemistry on a particle is associated with the stiffness of the chemical ODE, and it can differ significantly for small variations in composition. Therefore decomposing the domain with an equal number of particles fails to achieve a balanced load distribution. Although the computational cost of individual stochastic particles presents large fluctuations and is hard to predict, the overall distributions of cost in mixture fraction (z) space exhibit very similar scattering as shown in Figure 1(a). Moreover, as shown in Figure 1(b), the computational cost density functions $P_{cost}(z)$ for the two adjacent timesteps almost overlap. Here, $P_{cost}(z)$ is defined as the derivative of the cumulative cost distribution $F_{cost}(z)$

$$P_{cost}(z) = \frac{\mathrm{d}F_{cost}(z)}{\mathrm{d}z} = \frac{\mathrm{d}(\sum_p cost_p | z_p \leq z)}{\mathrm{d}z} \tag{11}$$

where $F_{cost}(z)$ is approximated by the gross cost of particles that have $z_p \leq z$. The good consistency of the computational cost density function over Δt can be used as a convenient measure to predict the computation cost based on the information collected from the past timestep.

In the current work, the new DLB model decomposes the mixture fraction space into N_{pc} partitions that correspond to an equal amount of computational load based on the $P_{cost}(z)$ evaluated from the last timestep. Here, N_{pc} is the number of processors. The N_{pc} continuous mixture fraction intervals covering $z \in [0\ 1]$ are dynamically generated and are assigned to different processors. Particles that fall within the same intervals are collected from the entire domain, solved by the same processor, and then returned to their owners via MPI. This methodologically simple and effective approach ensures that every processor can solve the assigned problems within similar wall time and thus achieve a load-balanced condition. Decomposition in mixture fraction space also enables processors to solve similar chemical problems, benefiting the DAC and ISAT models. Note that

the present methodology is not limited to decomposition in mixture fraction space only and can be extended to other conditioning dimensions depending on the complexity of the target flames. The success of the conditioning will in general depend on how strong the correlation is between the computational cost and the chosen conditioning variable. In non-premixed combustion the mixture fraction is a suitable choice and, although not tested here, we expect a progress variable to be suitable in premixed combustion. Departures from pure non-premixed combustion and pure premixed combustion lead to weaker correlations with the mixture fraction and progress variable, respectively, and at some point, two conditioning variables may be needed. This is case specific but, in principle, our load balancing method is also applicable to two conditioning variables although its performance remains to be tested

The DLB model includes two major components: (1) the dynamic decomposition strategy and (2) the data transmission module. These are discussed in turn below.

3.2. *Dynamic decomposition strategy*

The decomposition strategy is a critical component that decisively affects the extent of load balance. In the present work, two methods are proposed to dynamically adjust the decomposed boundary in mixture fraction space.

3.2.1. *Global adjusting*

The global adjusting method computes the mixture fraction boundaries by dividing the cost density function, $P_{cost}(z)$, c.f. Figure 1(b), into N_{pc} pieces so that their integrated areas are the same. Considering the average cost

$$cost_{av} = \frac{\int_0^1 P_{cost}(z)\mathrm{d}z}{N_{pc}}, \tag{12}$$

the z boundaries can be found starting from either end of the mixture fraction space. For instance, given $z_0 = 0$ and $P_{cost}(z)$ is evaluated from the previous timestep, z_1 can be uniquely determined via

$$\int_{z_0}^{z_1} P_{cost}(z)\mathrm{d}z = cost_{av}, \tag{13}$$

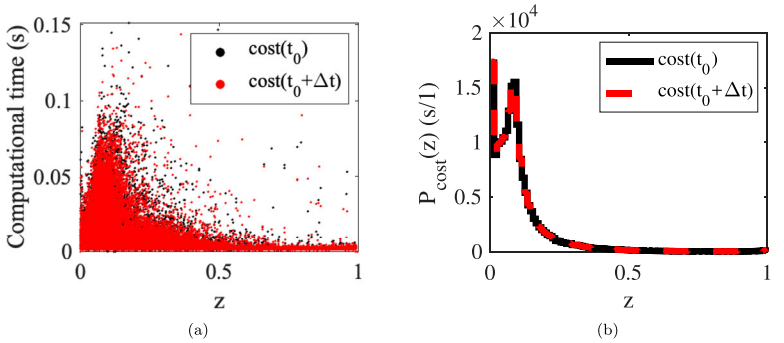


Figure 1. (a) Scatter plot of computational cost for chemistry versus mixture fraction at two adjacent numerical time steps. (b) Computational cost density function evaluated from the scatter data.

and subsequently for z_n, where $n = 2, \ldots, N_{pc}$, via

$$\int_{z_{n-1}}^{z_n} P_{cost}(z)\mathrm{d}z = cost_{av}. \tag{14}$$

Note that, $P_{cost}(z)$ is evaluated by binning particles based on mixture fraction and estimated by constant approximation within each bin. Dividing z space into more bins can increase the resolution of the evaluated $P_{cost}(z)$; however, it may result in a fewer number of particles in each bin, causing larger stochastic fluctuation, which scales by $\mathcal{O}(N_p)^{-1/2}$. Here, N_p represents the number of particles in each bin. We have compared $P_{cost}(z)$ evaluated for 1000, 10,000 and 50,000 bins which revealed that 1000 bins are insufficient to capture the peak in the cost density function whereas 10,000 equal-width bins produce nearly identical results to 50,000 bins without an excessive computational burden and are therefore deemed to be adequate.

3.2.2. *Diffusive adjusting*

The diffusive adjusting method is motivated by the significant fluctuating performance of the global adjusting method when coupled with ISAT (shown in Section 5). It is worth emphasising that the diffusive adjusting method is a balancing strategy that makes an analogy to diffusion but does not simulate any physical diffusion processes. The ISAT algorithm stores and tabulates DI solutions to approximate solutions of similar chemical states. For parallel computing, it is a standard implementation that ISAT builds tables in the private memory space of individual processors. The current DLB model with mixture fraction decomposition can methodologically decrease the accessed region to be tabulated in each processor and increase the efficiency of ISAT. However, the accessed region is highly dimensional in composition space, and it is computationally intractable to fully be tabulated. The stochastic evolution of the Lagrangian composition field may result in varying retrieve rates (or DI rates) and hence computational cost fluctuations. The global adjusting method may assign z-intervals to processors that are very different from the z-intervals, and thus tabulated regions, assigned at previous time steps, which nullifies the effort for the previous tabulation and intensifies the imbalance of the load distribution. Therefore a diffusive approach is proposed that gradually shares the computational load of a processor with its neighbours and reduces the timestep to timestep fluctuations of cost.

In principle, the diffusive adjusting method modifies the assigned mixture fraction boundaries progressively so that the wall time of computing chemistry relaxes toward the mean via a diffusive manner. A schematic illustration is shown in Figure 2. The computational load, indicated by the height, of the processor P is shared with the neighbouring processors K and N via modifying the left and right z-boundary by Δz_l and Δz_r, respectively.

The relationship between the computational load distribution L and Δz of a processor is formulated as follows. We firstly derive the change of computational load ΔL if it follows a kind of diffusing behaviour. A diffusion equation in z space has the form

$$\frac{\partial L}{\partial t} = D_c \frac{\partial^2 L}{\partial z^2} \tag{15}$$

where D_c represents the diffusion coefficient of the computational load. Here, the equation can be extended to multidimensional space by replacing z with a characteristic space $\boldsymbol{\zeta} =$

$(\zeta_1, \zeta_2, \ldots, \zeta_n)$, but this is not considered in the current work. Applying standard finite volume techniques and Gauss's divergence theorem leads to the semi-discrete form

$$\frac{\partial L}{\partial t} = \frac{D_c}{V}\left[\left(-S\frac{\mathrm{d}L}{\mathrm{d}z}\right)_l + \left(S\frac{\mathrm{d}L}{\mathrm{d}z}\right)_r\right] \tag{16}$$

where $\cdot_l$ and $\cdot_r$ represent quantities on the left and right faces of a cell respectively, and S is the face area. For a cell P and its neighbouring cells K and N, as shown in Figure 3, $V = (z_r - z_l) \cdot 1 \cdot 1$ and $S_f = 1 \cdot 1$. Here, the cells basically represent processors with L_i being the load on rank i, where $i = K, P, N$ in this example, and z_l and z_r being the assigned z boundaries for process rank P. Using the explicit forward scheme in time and central approximation in space, a discrete form of Equation (16) can be obtained

$$L_P^n = L_P^o + \underbrace{\left(-\frac{\mathrm{d}t \cdot D_c}{z_r - z_l}\frac{L_K - L_P}{z_K - z_P}\right)^o}_{\Delta L_{P,l}} + \underbrace{\left(\frac{\mathrm{d}t \cdot D_c}{z_r - z_l}\frac{L_P - L_N}{z_P - z_N}\right)^o}_{\Delta L_{P,r}} \tag{17}$$

where superscript $\cdot^n$ is the value at the new timestep and $\cdot^o$ is the value at the old timestep. The second and third terms on the rhs represent the load changes, $\Delta L_{P,l}$ and $\Delta L_{P,r}$, caused by the modification of Δz_l and Δz_r, respectively. For the left boundary, they can be linked via $\Delta z_l = -\Delta L_{P,l}/\rho_{z_l}$, where ρ_{z_l} represents the computational cost per unit length in z space at z_l and is approximated by $\frac{1}{2}(L_K/\Delta z_K + L_P/\Delta z_P)$. Therefore, the increment of the left z-boundary for the cell P is

$$\Delta z_l = \left[\frac{\mathrm{d}t \cdot D_c}{z_r - z_l}\frac{L_K - L_P}{z_K - z_P}\right] \Big/ \left[\frac{1}{2}\left(\frac{L_K}{\Delta z_K} + \frac{L_P}{\Delta z_P}\right)\right], \tag{18}$$

and similarly, for the right boundary

$$\Delta z_r = \left[\frac{\mathrm{d}t \cdot D_c}{z_r - z_l}\frac{L_P - L_N}{z_P - z_N}\right] \Big/ \left[\frac{1}{2}\left(\frac{L_P}{\Delta z_P} + \frac{L_N}{\Delta z_N}\right)\right]. \tag{19}$$

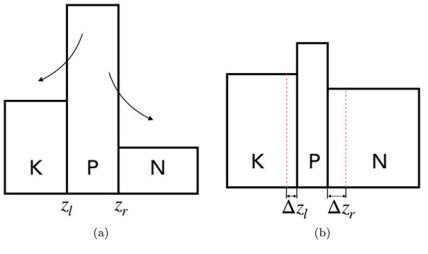


Figure 2. (a) Computational load distribution before adjusting. (b) Computational load distribution after adjusting z boundaries by Δz_l and Δz_r.

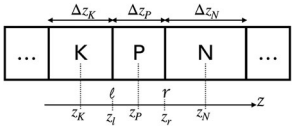


Figure 3. Discretisation in z-space.

Equations (18) and (19) represent a model for the required increments on the z boundaries to enforce load redistribution in a diffusive manner. The values are computed based on the computational load L_i and the z decomposition in the past timestep. The general stability condition for a diffusion equation is

$$\frac{D_c \Delta t}{\Delta z^2} < \frac{1}{2} \tag{20}$$

where Δt is the timestep size and Δz is the width of the decomposed z-intervals. Therefore, to keep the adjusting algorithm stable, i.e. no overshooting,

$$D_c = \frac{1}{2} K_c \frac{(\Delta z)^2_{min}}{\Delta t} \tag{21}$$

where $(\Delta z)_{min}$ is the smallest z intervals, $0 < K_c < 1$ is a coefficient to adjust stability and balancing performance, and the present study uses $K_c = 0.5$.

3.2.3. *Coupling with DI, DAC and ISAT*

Generally, the global adjusting method has the advantage of numerical simplicity and having a fast response to an evident load imbalance but may overshoot the adjustments if cost estimation is not precise, causing a performance drop. The diffusive adjusting method takes time to balance the load, but it can achieve more stable and predictable performance. In the tests shown later in Section 5, the global adjusting method is adequate to balance the load for DI and DAC. Therefore the global adjusting method is used solely for DI and DAC, and the z decomposition is adjusted every timestep. The coupled methods are denoted by *DLB-DI* and *DLB-DAC*.

However, coupling the global adjusting method solely with ISAT, denoted as *DLB-ISAT (Global)*, leads to a strongly oscillating load distribution and deteriorates model performance, as shown in Section 5. In the present work, a combination of the global and diffusive methods is proposed. In the combined adjusting method, diffusive adjusting is executed every two timesteps to minimise the influence on the cost estimation due to adding and growing operations for the newly updated z-intervals, and the global adjusting method is activated when the extent of load imbalance is greater than a user-defined threshold. The extent of load imbalance can be quantified by the potential improvement (PI) defined as

$$PI = \frac{Cost_{max} - Cost_{av}}{Cost_{max}} \tag{22}$$

where $Cost_{max} = max(L_i)$ and $Cost_{av} = \sum_{i=1}^{N_{pc}} L_i / N_{pc}, i = 1, \ldots, N_{pc}$. The PI indicates how far away the current condition is from the ideally balanced condition. $PI = 0$ implies a completely balanced load distribution, whereas $PI = 1$ indicates the most extremely imbalanced condition. A user-defined parameter ε_{PI} is used as a threshold, and the global adjusting method is activated when $PI_{min} > \varepsilon_{PI}$, where PI_{min} is the minimum PI in the past t_r timesteps. This condition is examined every t_g timesteps. t_r and t_g should take values that are sufficiently large such that ISAT has adequate data to build a table leading to stable computational cost predictions with which the load can be effectively balanced. However, if the values are too large the subsequent adjustment towards a well-balanced load condition is slow and computational time is sub-optimal. In the present work, $t_r = 5$ and $t_g = 25$. The chosen values in the current work are a compromise of those two effects.

3.3. *Data transmission module*

The data transmission module is responsible for workload redistribution among processors. In practice, notional particles carry many supporting data and objects that are not required for chemical calculation. Therefore, only the necessary thermochemical scalars, i.e. chemical species concentration, temperature, pressure and required supporting data, are enveloped and transferred in the DLB model. The data define complete chemical ODEs which can be solved by the receiving processors. The enveloping operation on each processor groups data according to their mixture fraction and the dynamically adjusted mixture fraction boundaries $\boldsymbol{z} = \{z_0, z_1, \ldots, z_n\}$ and then sends the data to buffers for non-blocking MPI communications.

A flowchart of the scheme is shown in Figure 4 illustrating the following operations of DLB model.

(i) At the start of an iteration, computational cost information of the last iteration is collected and used to update mixture fraction division $\boldsymbol{z}$ on the master node. The updated division is then broadcasted to all processors.
(ii) Each processor groups chemical problems based on their mixture fraction and the updated mixture fraction boundaries $\{z_0, z_1, \ldots, z_n\}$, and sends the enveloped data of particles in the n-th z-intervals to the n-th processor.
(iii) Processors solve the received problems. While solving chemistry, the cost of each ODE and the corresponding mixture fraction are recorded and packaged with the solutions. If tabulation is invoked, private ISAT trees are built based on the problems solved in each processor.
(iv) Once the calculation is finished, the packaged solutions are returned to the memory space of the original owners, and the thermochemical states of the notional particles are updated.
(v) Lastly, the computational cost information is summarised to update mixture fraction division $\{z_0, z_1, \ldots, z_n\}$ in the next iteration.

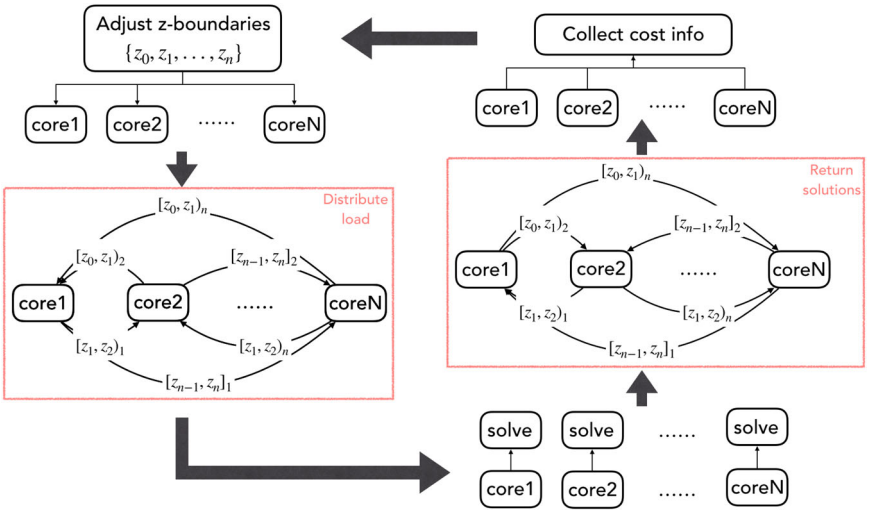


Figure 4. A schematic of data flow between processors in the DLB model.

4. Numerical implementation

4.1. *mmcFoam*

The MMC-LES has been implemented in the object-orientated C++ open-source platform, mmcFoam, and its numerical performance has been widely tested [33]. It is a hybrid Eulerian–Lagrangian solver for turbulent combustion. There is a two-way coupling between the Eulerian and Lagrangian fields. The forward coupling interpolates the filtered quantities from the Eulerian fields to particle locations to integrate the SDEs, while the backward coupling maintains mass consistency between the two fields and feedbacks density to LES. The backward coupling employs a conditional form [7] of the equivalent enthalpy method [34]. Several transport equations for major species are solved, and the source terms for each cell are modelled to relax toward the conditional mean evaluated by the flamelet regression method [7] or the kernel estimation method [33]. Complete details about mmcFoam can be found in [33].

4.2. *DLB model*

The DLB model is a supplementary model that is only activated for chemistry calculations and is independent of the other components of the solver, as illustrated in Figure 5. Compared to the previous version of mmcFoam [33], the DLB implementation in the submodels is an additional layer between the base class of finite rate chemistry and the calculation schemes, e.g. DI, DAC and ISAT. Therefore, different methods can be flexibly selected and coupled with the DLB model. When DLB is combined with ISAT, the model tabulates all computed chemical solutions locally, i.e. in the memory spaces where they are solved, and does not return any ISAT-related objects to the original processors. Such implementation allows processors to tabulate only a narrow range of mixture fractions, which corresponds to a much smaller region in composition space than the whole realisable region. When the

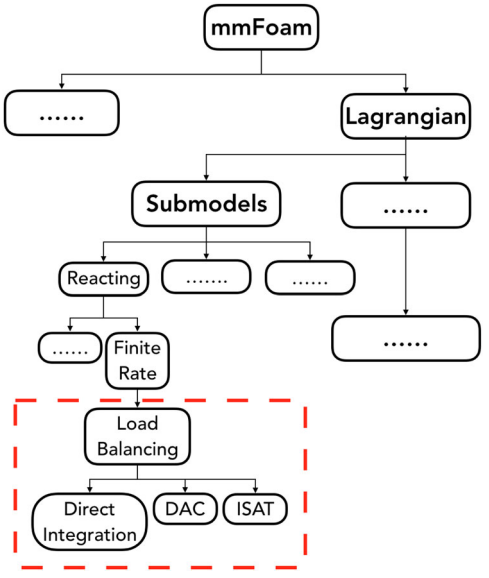


Figure 5. A schematic of code blocks for mmcFoam and the newly implemented DLB model. This block diagram should be read in conjunction with the block diagram shown in Figure 1 in [33].

processors solve chemical problems within the same, or similar, mixture fraction range, the possibility of successful retrieval from the table is indeed greater, resulting in higher computational efficiency of ISAT.

5. Performance evaluation

The performance of the DLB models is tested under two laboratory non-premixed flame conditions. In both cases, the spatial domains are partitioned by a conventional domain decomposition method in the axial direction with an equal number of particles per processor and the purpose, therefore, is to assess the benefits of the DLB methods. The first case is a Sydney piloted flame on a burner with adjustable inlet mixture homogeneity [35, 36]. Here, the case FJ200-5GP-Lr300-59, which has a homogeneous mixture (CH_4/Air) at the burner exit, is considered. This case was simulated previously by Galindo et al. using mmcFoam [10] and represents a condition where computational load distribution is relatively balanced by conventional domain decomposition due to the relatively constant size of the near-stoichiometric reactive fluid volume along the flame axis. Because the case is less computationally demanding, it is used to validate the correctness of code implementation. The second case is the piloted non-premixed methane flame, Delft III flame [37, 38], which has a highly variable reactive region volume along the flame axis and represents a test case for which conventional domain decomposition does not lead to good load balancing and for which the new DLB method is expected to provide a significant computational performance. The effectiveness of the DLB model is assessed by comparing the computational efficiency with and without load balancing in this turbulent reactive flow. All flame simulations are partitioned by the standard domain decomposition method along the axial direction with an equal number of particles per processor.

Simulations for both flame cases are carried out using a GRI-3.0 chemical mechanism [39] which has 53 species and 325 reactions. The DLB overhead is mainly induced by data transmission and is associated with the number of species. Although only one mechanism is tested in this work, the DLB overhead for other chemical kinetics is expected to scale linearly with respect to the number of species. The base cases use ISAT tolerances of $\varepsilon_{ISAT} = 10^{-4}$ and DAC tolerance of $\varepsilon_{DAC} = 10^{-2}$ with CH_4 being the SIS. These tolerances are found to be reasonably accurate in the tests shown later.

5.1. *Sydney piloted flame with homogeneous fuel mixture*

5.1.1. *Experimental setup and numerical configuration*

A schematic diagram of the burner is shown in Figure 6 along with the temperature contour of the flame produced by MMC-LES. A retractable inner pipe with a diameter of 4 mm is concentric with a $D_{fj} = 7.5$-mm-diameter outer tube that is surrounded by a pilot annulus of pilot with a diameter of 18 mm stabilised on a perforated plate located at 4mm upstream of the burner's exit plane. The retraction length, Lr, controls the mixing extent between the fuel and air issuing respectively from the inner and outer tubes and thus the homogeneity of the fuel mixture at the burner exit. For the case studied here, Lr is at its maximum value, producing a non-premixed flame with a homogeneous CH_4/Air mixture with a bulk velocity of 59 m/s. The entire burner is centred in a wind tunnel where a coflow stream of 15 m/s is provided. In the present study, the same numerical configuration of the previous work [10] is used, i.e. the same grid, notional particle distribution and boundary conditions.

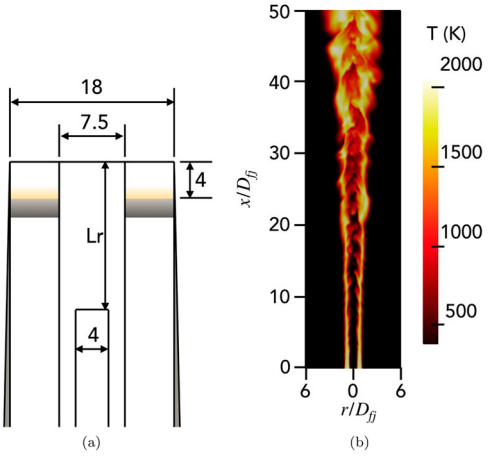


Figure 6. (a) A schematic of the Sydney piloted burner with retractable inner pipe. (b) Instantaneous Eulerian equivalent temperature field from simulation.

The flamelet regression method is used for density coupling. The time-averaged results are obtained for the same period of 6 ms starting from the same initial condition. The efficiency of the DLB model is tested on an eight-core desktop workstation equipped with Intel(R) Xeon(R) Gold 5115 CPU @ 2.40GHz.

5.1.2. *Results*

We firstly examine the correctness of the code implementation by comparison of the time-averaged results produced by DI, DLB-DI, DLB-DAC and DLB-ISAT. Here, the DLB-ISAT uses a combined adjusting method with $\varepsilon_{PI} = 0.25$. The radial profiles of temperature are shown in Figure 7. The overlapping of the results for DI and DLB-DI presents a validation of the DLB code implementation. As for ISAT and DAC, while $\varepsilon_{ISAT} = 10^{-3}$ and $\varepsilon_{DAC} = 10^{-1}$ are too coarse leading to appreciable deviations from the DI results, $\varepsilon_{ISAT} = 10^{-4}$ and $\varepsilon_{DAC} = 10^{-2}$ produce an excellent agreement with the DI results, indicating that these tolerances, which are used in the remainder of this work, are adequate.

The temporal evolutions of computational cost and *PI* using DI and DLB-DI are shown in Figure 8. Note that the computational cost is measured by the wall time taken to solve chemistry and includes all related overhead. By inspecting the results of *PI*, the original case setup without the DLB model shows a fairly balanced load distribution in that *PI* fluctuates around the mean value of 0.071. Even though this condition leaves a very small margin for improvement, the DLB achieves a lower mean *PI* at around 0.037. However, although the DLB model achieves around 50% lower *PI*, it leads to a rather minor enhancement in the computational speed.

Figure 9 presents the temporal evolution of computational cost and *PI* using DAC with and without the DLB model. When DLB is coupled with DAC, the computational cost accounts for DI and all DAC operations. Overall, the DLB-DAC model presents a slightly lower *PI* in the simulations than the standalone DAC model, indicating an improved load distribution. DAC dynamically reduces the chemical kinetics and shortens the computational time by around 28% compared to DI; however, applying DLB to DAC shows

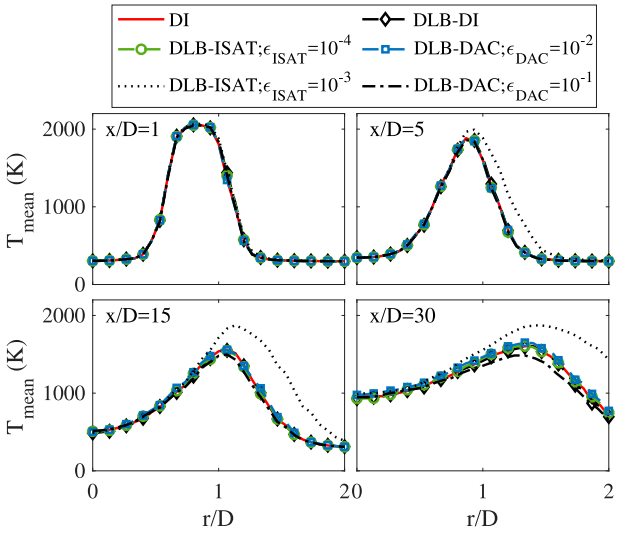


Figure 7. Radial profiles of mean temperature using different methods.

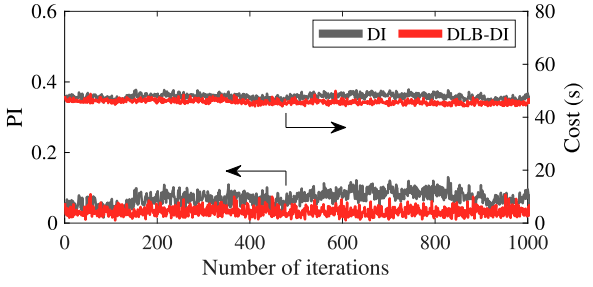


Figure 8. Temporal evolutions of potential improvement (*PI*) and computational cost for simulations of the Sydney flame with homogeneous compositional inlet using DI and DLB-DI.

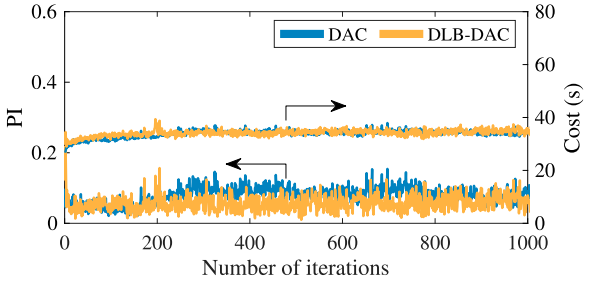


Figure 9. Temporal evolutions of computational cost and potential improvement (*PI*) for simulations of the Sydney flame with homogeneous compositional inlet using DAC and DLB-DAC.

marginal improvement in the simulation since the original computational load distribution is reasonably balanced.

As can be seen in Figure 10(a), ISAT breaks the original balanced load distribution. The number of successful retrievals from the table varies due to the stochastic evolution of

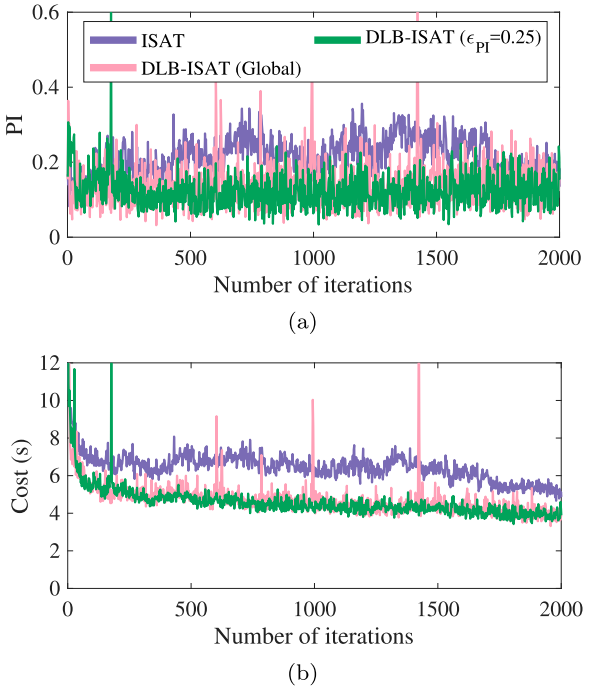


Figure 10. Temporal evolutions of (a) potential improvement (*PI*) and (b) computational cost for simulations of the Sydney flame with homogeneous compositional inlet using ISAT and DLB-ISAT.

particles in the composition space, and consequently, the performance of ISAT fluctuates. The deteriorated load distribution using ISAT is in fact attributed to the different speedup effects on different processors.

The influence of the dynamic adjusting method on the performance of DLB-ISAT methods is shown in Figure 10(a). The methods using the global and combined adjusting methods are denoted as DLB-ISAT (Global) and DLB-ISAT ($\varepsilon_{PI} = 0.25$), respectively. Here, the DLB-ISAT (Global) applies the global adjusting method continuously, and the combined method specifies an allowed extent of imbalance, $\varepsilon_{PI} = 0.25$, below which load distribution is only tailored by the diffusive adjusting approach. Both DLB-ISAT methods present improved load distributions relative to straight ISAT. However, DLB-ISAT (Global) tends to cause larger fluctuations and produces an overall larger *PI* than DLB-ISAT ($\varepsilon_{PI} = 0.25$). Overshooting of load adjustment by DLB-ISAT (Global) can be seen from the spikes in the time series and is caused by abrupt changes to the mixture fraction divisions and the resultant need to grow the ISAT tables by direct integration. DLB-ISAT ($\varepsilon_{PI} = 0.25$) suppresses the overshooting quite well once the initial transient phase passes.

Comparing Figure 10(b) with Figure 8 indicates that, after an initial period to build the table, ISAT effectively reduces the computational wall time by around 87% compared to DI. Using ISAT alone exhibits relatively large fluctuation in computation wall time, but there is a decreasing cost trend over time. Both DLB-ISAT methods show similar fast-developing stages initially and then transit to stages presenting slow but steady computational efficiency increase, achieving similar computational at the end of the simulations. Although more spikes are produced by the DLB-ISAT (Global) method, the overall computational cost using DLB-ISAT (Global) and DLB-ISAT ($\varepsilon_{PI} = 0.25$) are similar.

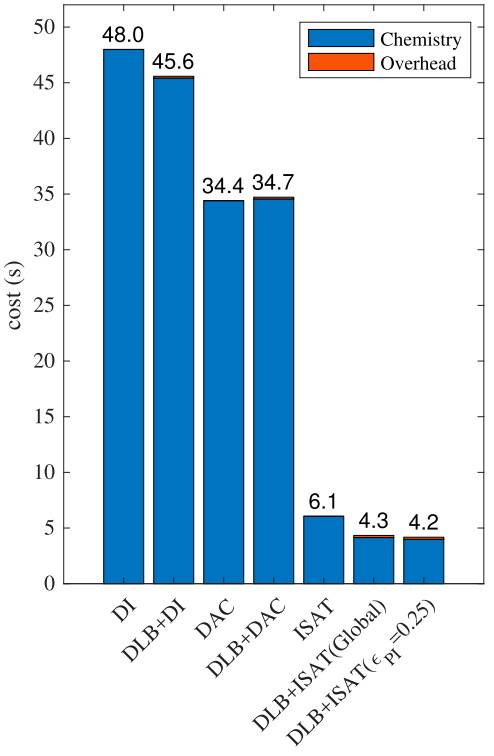


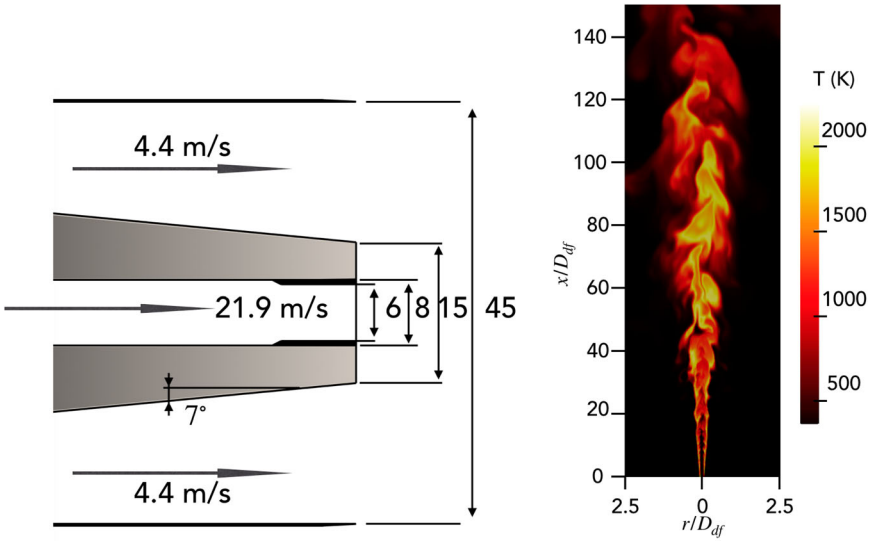
Figure 11. Averaged computational cost using different methods.

The averaged computational cost of chemistry for different expediting algorithms is displayed in Figure 11. The averaged wall time and overhead are the results averaged over the second half of the simulation runs. The most obvious feature is the vanishingly small overhead, which is around 0.17 s per time step for all methods using DLB. This demonstrates the excellent efficiency of the DLB methods and implementation. However, overall for this flame, the DLB methods lead to only a limited reduction in computational cost because the load distribution is already well balanced by physical domain decomposition. However, the kinetics reduction (DAC) and tabulation (ISAT) methods provide appreciable savings in computational cost compared to DI. The DLB-DAC method is slightly slower than DAC by 0.3 s resulting from the DLB overhead and fluctuations. In contrast, the DLB-ISAT tends to have higher efficiency than straight ISAT due to slightly improved load distribution and the localised tabulation in the mixture fraction space. Nevertheless, the results shown in this section prove the correctness of the code implementation and the low overhead associated with the DLB methods

5.2. *Delft III methane/air non-premixed flame*

5.2.1. *Experimental setup and numerical configuration*

A schematic diagram of the burner is shown in Figure 12 along with the temperature contour of the flame. The burner exit has a central fuel jet with an inner diameter $D_{df} = 6$ mm surrounded by the pilot issued from 12 equidistant 0.5-mm-diameter holes positioned



(a) (b)

Figure 12. (a) A schematic of the Delft flame burner. (b) Instantaneous Eulerian equivalent temperature field from MMC-LES.

on a 7-mm circle with an outer diameter of 8 mm. A concentric metal rim surrounding the pilot insert is centred in a 45-mm pipe, producing an annulus of 15 mm from where the primary coflow air is ejected. The fuel jet and the primary air leave the burner exit with velocities of 21.9 m/s (Re = 9700) and 4.4 m/s (Re = 8800), respectively, while the entire burner is situated in a wind tunnel with a coflowing secondary airstream at 0.3 m/s. The pilot flame insert in the fuel pipe causes a decrease in jet diameter from 8 mm to 6 mm starting at a location 16 mm upstream of the burner exit, resulting in a convergent fuel nozzle, and the reduction of the outer diameter of the rim from 30 to 15 mm with a 7° angle produces a divergent air annulus nozzle that causes the non-negligible radial velocity at burner exit. The fuel is natural gas, and the pilot burns a mixture of acetylene/hydrogen/air with a carbon-to-hydrogen ratio equal to 1:4 corresponding to an equivalence ratio of 1.4 and accounts for around 1% of the total thermal power of the flame [37, 38].

The computational domain is a cylinder with a diameter of 250 mm and length of 900 mm, discretised by a stretched grid with 623, 114 and 48 cells in axial, radial, and circumferential directions, respectively. Following Mueller and Pitsch [40], two-stage pipe flow simulations are used to produce inflow boundary data for the fuel jet and primary coflow exits. However, similar to the FDF study by Donde et al. [41], fluctuations of the primary coflow data are discarded to suppress excessive flame extinction. For the composition, approximately 0.95 million particles in total are used, leading to a sparseness of one Lagrangian particle per 3.7 Eulerian cells (1L/3.7E). During a simulation, a particle number control algorithm maintains spatial resolution by cloning or removing stochastic particles within control cells containing multiple LES cells. The number is controlled to be 25 $\pm$ 5 particles per control cell. The kernel estimation method is used for density coupling. The base simulation is carried out by 32 processors on Tianhe-II with CPU Intel(R) Xeon(R) CPU E5-2697A v4 @ 2.60GHz.

5.2.2. *Results*

The temporal evolution of *PI* and computational cost and the load distributions of the latest timestep using DI and DLB-DI are shown in Figure 13. The indicator *PI* reaches around 0.44 on average using DI, implying significant load imbalance by conventional physical domain decomposition. Applying the DLB method shows excellent performance and reduces the *PI* to around 0.03, indicating a well-improved load distribution. A direct comparison of load distribution at the latest timestep is shown in Figure 13(b). The computational wall time of solving the reacting fractional step is determined by the slowest processor; therefore, using DI without the DLB method requires around 180 s to solve chemistry. The DLB method redistributes the computational load rather evenly among computing cores, and all cores finish calculation with a similar time around 105 s, resulting in lower computational wall time than that of DI. As the flame evolves, DI exhibits a slight decrease in computational cost due to particle convection in the slowest core, whereas DLB-DI balances load and presents an almost constant cost for each timestep.

Temporal evolution of *PI* and computational cost along with the load distribution at the last timestep in the tests are shown in Figure 14 for DAC and DLB-DAC. Relative to DI, DAC reduces the size of chemical ODE and achieves enhanced computational efficiency in all the cores; however, the *PI* increases to about 0.55 which is higher than the *PI* value obtained for DI and this is due to the different extent of kinetics reduction on processors. The coupled DLB-DAC method experiences a short period of oscillation at the start of the simulation and then converges to a steadily balanced condition with the *PI* of around 0.02, leading to a lower computational cost. The excellent performance of the DLB method can be clearly confirmed by comparing the load distributions with and without the DLB method at the final timestep in the test in Figure 14(b), i.e. the computational load is effectively balanced by the DLB method.

The temporal evolution of *PI* and computational cost using ISAT and DLB-ISAT methods are shown in Figures 15(a) and 15(b), respectively. When using ISAT, both *PI* and computational cost present a similar decreasing trend at the start of the simulation and then fluctuates around 0.4 and 30 s, respectively. Compared to the DI and DAC results discussed

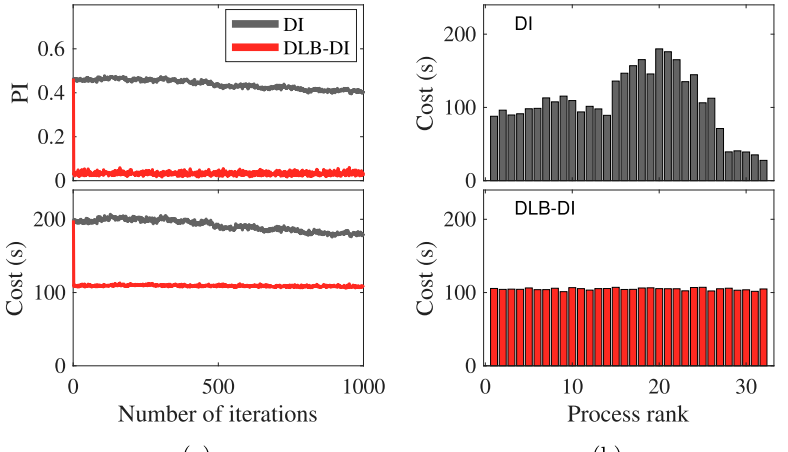


Figure 13. (a) Temporal recordings of *PI* and cost for simulations of the Delft III flame using DI and DLB-DI; (b) Load distribution using DI and DLB-DI at the latest timestep.

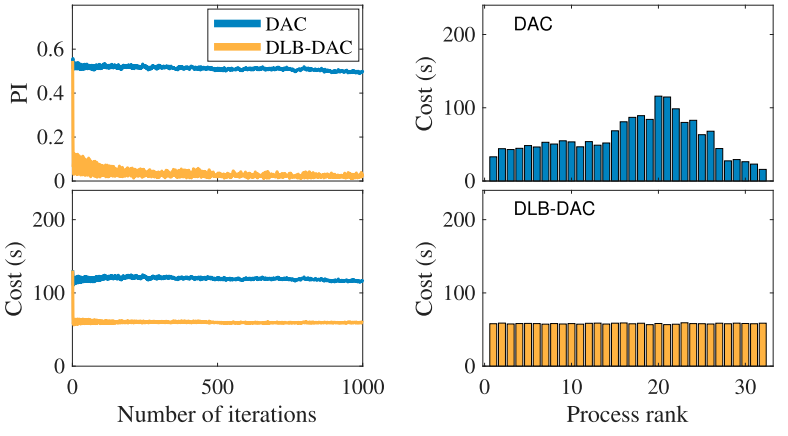


Figure 14. (a) Temporal recordings of *PI* and cost for simulations of the Delft III flame using DAC and DLB-DAC; (b) Load distribution using DAC and DLB-DAC at the latest timestep.

above, ISAT produces a higher load imbalance since the increased computational efficiency depends on the quality of tabulation and computation history, which differs significantly in different cores.

Both ISAT-DLB methods lead to overall lower *PI* and cost, but distinct performances are observed. The DLB-ISAT (Global) method shows significant oscillation in its balancing performance because it enforces continuous global adjusting to update z decomposition. The frequent and abrupt changes to the z divisions lead to composition spaces being accessed which are not in the table requiring additional computational effort to grow the tables in these new z-intervals by direct integration. Therefore, the computational cost information collected from the last iteration may not reflect an accurate cost prediction for the current time step, causing degraded load distribution and spikes in the *PI* and computational cost time series. The computational cost shows similar fluctuations with occasionally extreme overshooting (capped at 80 s in the plot, but the slowest timestep can take around 17 times longer than using ISAT alone) and the spikes of computational cost and *PI* are synchronised, appearing at the same timestep.

The DLB-ISAT ($\varepsilon_{PI} = 0.25$) method effectively suppresses the global adjusting method when it is not necessary, i.e. $PI_{min} \leq \varepsilon_{PI}$, and leads to far fewer temporal fluctuations in *PI* and computational cost. When the extent of load imbalance is beyond the user-defined threshold, i.e. $PI_{min} > \varepsilon_{PI}$, the global adjusting method is activated to provide prompt z-intervals adjustments. Such a condition is encountered in the simulation at around the 1200th timestep, where spikes in *PI* and cost are observed. The spike and the subsequent gradual decrease in the cost profile indicate a new table-buildup stage, after which lower *PI* and cost are both achieved.

The load distribution of two timesteps, the 1600th and 2000th timestep, are extracted to illustrate the details of the methods' performance, as shown in Figure 15(c ,d), respectively. It can be seen that the majority of computational cost of ISAT with and without DLB is associated with the unsuccessful retrieval from the ISAT table followed by necessary DI, whereas successfully retrieving solutions from tables is efficient and accounts for only a small fraction of the total cost. The standalone ISAT presents various speedup levels on

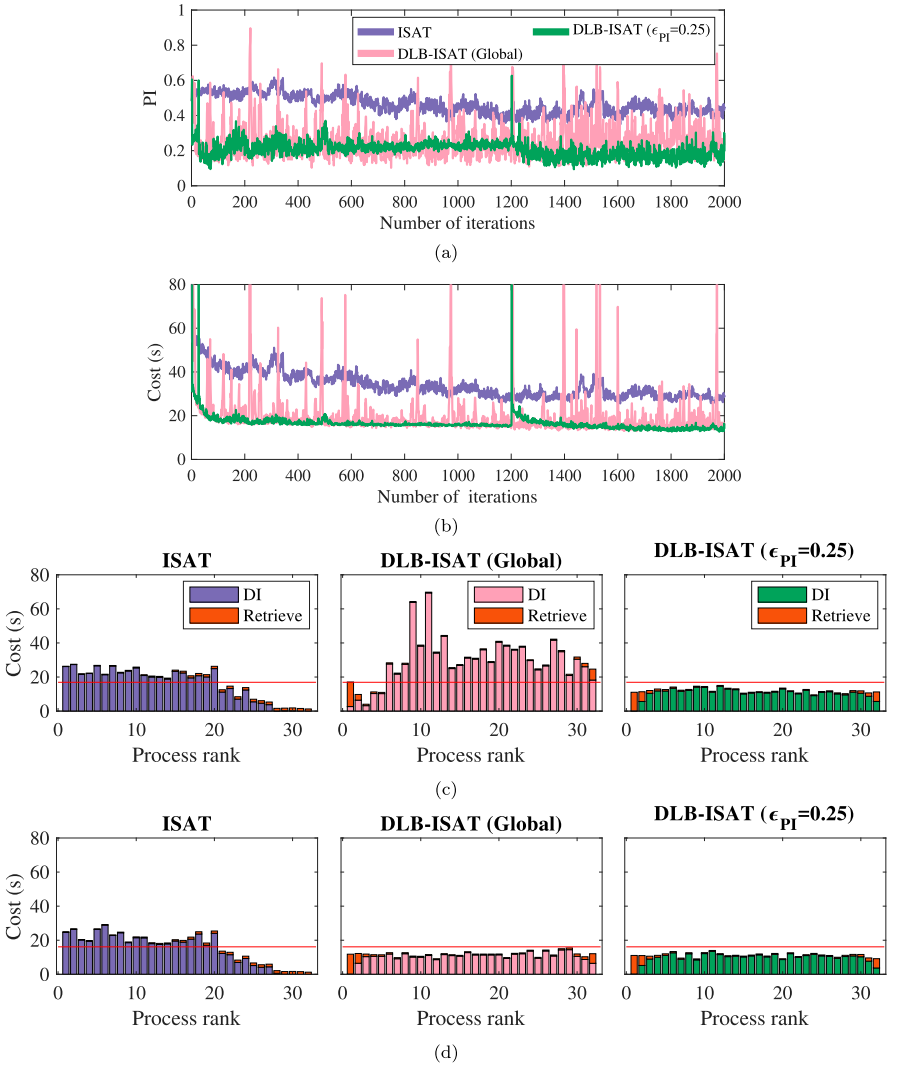


Figure 15. Temporal recordings of (a) *PI* and (b) computational cost for simulations of the Delft III flame using ISAT and DLB-ISAT. Computational load distribution at the (c) 1600th and (d) 2000th timestep. The red horizontal lines indicate the ideal computational cost using ISAT alone when load is perfectly balanced.

different processors and results in a worsened load distribution (larger *PI*). The DLB-ISAT (Global) method can cause extreme imbalance when too abrupt adjusting is applied to z-intervals, shown in Figure 15(c). Although overshooting exists, the DLB-ISAT (Global) method has the potential to achieve proper load distribution as shown in Figure 15(d). As for the DLB-ISAT ($\varepsilon_{PI} = 0.25$) method, load distribution between the 1600th and 2000th timestep is only via the diffusive method (no abrupt changes) and the load balance is very stable.

The computational time required for an ideally balanced load distribution using ISAT, $Cost_{av,ISAT}$, is indicated by the horizontal line in the figures. Although the DLB-ISAT

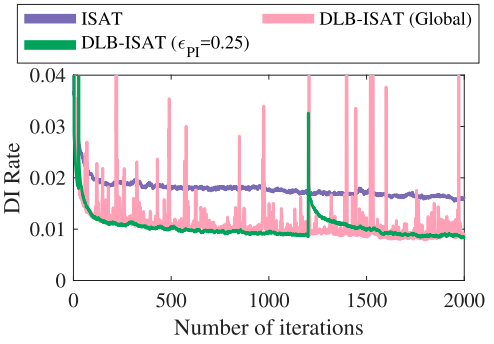


Figure 16. Temporal recordings of the DI rate for simulations of the Delft III flame using ISAT and DLB-ISAT.

methods do not balance load perfectly, they can both achieve lower computational time than $Cost_{av,ISAT}$. This is due to two aspects: (1) more even computational load allocation and (2) ISAT tables on each processor representing a smaller region of the accessed composition space. The former better harnesses the underloaded processors to solve a fixed amount of computational load within a shorter wall time. The latter is an outcome of the z decomposition introduced by the DLB method. In the DLB method, chemical ODEs with similar mixture fractions are grouped, solved and tabulated by the same processors. Because particles that are close in mixture fraction space usually carry similar composition for non-premixed combustion, the tables built by the DLB method are more frequently accessed than those produced via conventional physical domain decomposition. As shown in Figure 16, localised tabulation in z space can reduce the number of unsuccessful retrievals followed by DI by around 50% for DLB-ISAT ($\varepsilon_{PI} = 0.25$). Here, the DI rate is defined as the ratio between the number of DI executed and the total number of chemical state space updates.

The sensitivity of DLB-ISAT performance to the parameter ε_{PI} is investigated in Figure 17. Overall, larger ε_{PI} leads to less temporal fluctuation of the PI. Larger ε_{PI} is associated with fewer global adjustments, reducing overshooting. However, $\varepsilon_{PI} = 0.25$ shows the lowest PI and cost by the end of the test, implying that proper global adjustment improves load distribution and increases computational efficiency. This is because some of the z-intervals may be very wide and require an impractically long time to balance via the diffusive adjusting method, while appropriate global adjusting can efficiently update the z boundaries.

In the 2000-iteration test, DLB-ISAT ($\varepsilon_{PI} = 0.5$) activates global adjusting only once at the first iteration and then enforces pure diffusive adjusting after that. Therefore, the effectiveness of the diffusive adjusting method can be demonstrated by the evolution of computational load distribution shown in Figure 18. After a sufficient number of iterations, the crest of load distribution at the 100th timestep diffuses to a relatively flat shape at the 2000th timestep. However, the diffusive method is too slow to balance the load at the two ends of the mixture fraction range because the z-intervals are wide and the increments Δz that are calculated by the diffusive method at each time step are small. Note that Δz is linked to the load diffusivity D_c whose magnitude is limited by the $(\Delta z)_{min}$ in Equation (21). In contrast, the results of DLB-ISAT ($\varepsilon_{PI} = 0.25$) in Figure 15 have good load distribution even at the two ends of the mixture fraction range after a global adjusting at around the 1200th timestep.

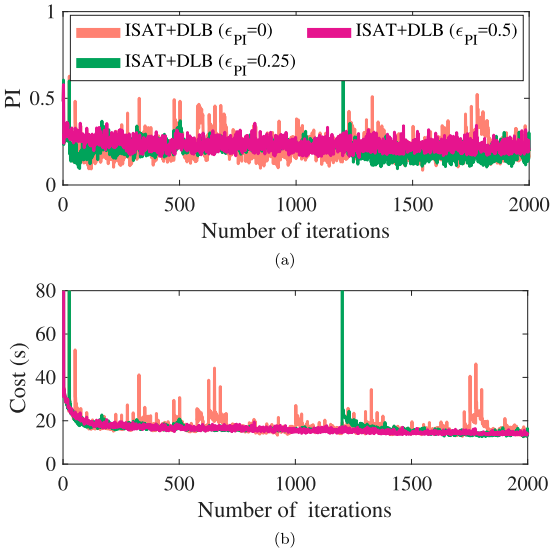


Figure 17. Temporal recordings of (a) *PI* and (b) computational cost for simulations of the Delft III flame using DLB-ISAT with different ε_{PI}.

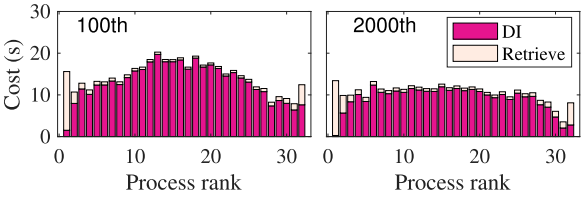


Figure 18. Load distribution of DLB-ISAT ($\varepsilon_{PI} = 0.5$) at the 100th and 2000th timestep.

The computational cost of solving chemistry using the different methods is summarised in Figure 19. The values are the averages over the second half of the time steps in the tests. In summary, applying the DLB method to DI and DAC can effectively reduce the computational time by around 41% and 50%, respectively. The DLB-ISAT method shows a slight difference depending on the adjusting strategies and the parameter ε_{PI} which controls the infrequent activation of the global adjusting method. Despite the overshooting of pure global adjusting, it shortens the computational time by around 36%. Using $\varepsilon_{PI} = 0$, 0.25 and 0.5 reduces the computational load by around 45%, 48% and 50%, respectively, compared to using ISAT alone. The overhead induced by DLB operations varies from 0.28 to 0.6 s and accounts for a negligibly small fraction of the computational time. Based on Figure 19, the speedup factors, which are the ratio between the wall time using DI and different methods, are summarised in Table 1.

Lastly, the parallel scalability of the DLB methods for solving chemistry is examined. Simulations using DI, DLB-DI and DLB-DAC are carried out for 100 iterations, and the statistics are averaged over the final 50 iterations due to the relatively stable performance in that period. For DLB-ISAT with $\varepsilon_{PI} = 0.25$ and $\varepsilon_{PI} = 1$, simulations were run for 2000 iterations and statistics are averaged over the final 1000 iterations. Here, DLB-ISAT ($\varepsilon_{PI} = 1$) represents a case enforcing a pure diffusive adjusting method. Figure 20(a) displays the

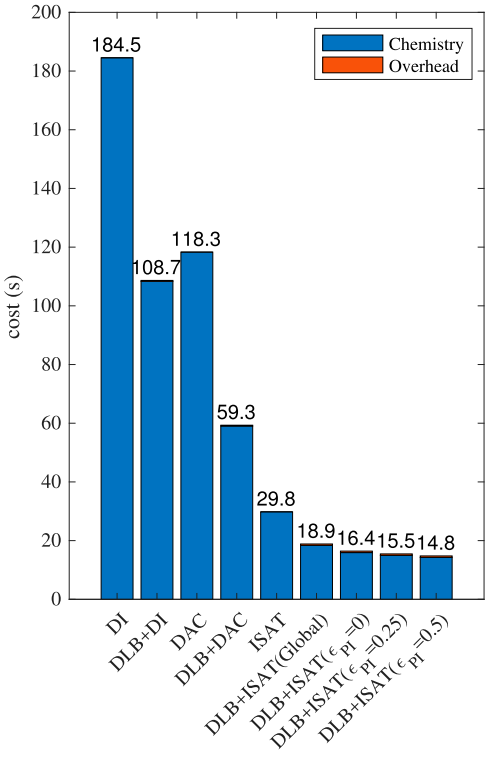


Figure 19. Averaged computational cost using different methods.

Table 1. Speedup factors using different methods.

	DI	DAC	ISAT
w/o DLB	1.00	1.56	6.19
with DLB	1.70	3.11	9.76 (Global)
			11.90 ($\varepsilon_{PI} = 0.25$)
			12.46 ($\varepsilon_{PI} = 0.5$)

so-called strong scalability results in which the problem has a constant size (i.e. constant number of particles) while the number of cores is increased sequentially in multiples of two. The speedup is the ratio relative to the computational cost of simulations using just 32 cores.

It is evident that all models coupled with DLB present improved scalability compared to that without DLB models. For 512 cores, the computational speed of DLB-DI, DLB-DAC and DLB-ISAT is improved by around 59%, 84% and 126% compared to DI, DAC and ISAT, respectively. The deviations of the scalability from the ideal speedup factors indicate drops in DLB performance as the number of cores increases. It should be emphasised that such drops are not caused by the increase in inter-processor communication, because the

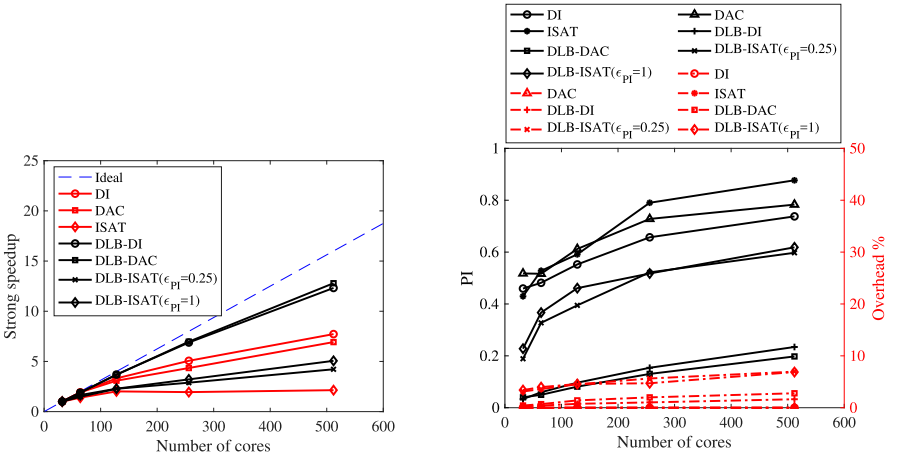


Figure 20. (a) Strong scalability of the DI and DLB methods. (b) Averaged *PI* and overhead percentage of the DI and DLB methods.

overhead of all models accounts for less than 10% of the total cost. Instead, the scalability drops are mainly related to the stochastic attribute of the notional particles. As the number of processors increases, fewer particles are distributed to each processor and the stochastic effect becomes stronger. As a result, the DLB model cannot maintain the same level of balanced load condition, as shown by the increase of *PI* in Figure 20(b), and thus cause drops of strong scalability. As for DLB-ISAT, the ISAT operations can amplify the stochastic fluctuations of computational cost because the add and growth operations of ISAT can be several orders of magnitude more computationally expensive than the retrieval operations, exacerbating load imbalance.

The strong scaling test indeed provides useful information on the parallel computing performance; however, from a practical point of view, the wall time required for computation using a different number of cores is more of interest to users. As can be seen from Figure 21, despite the relatively low scalability of DLB-ISAT for both ε_{PI} values compared to DI and DAC, the ISAT approaches still rank as the fastest of methods in these tests. The superior computing speed of DLB-ISAT is predominant when a small number of cores are

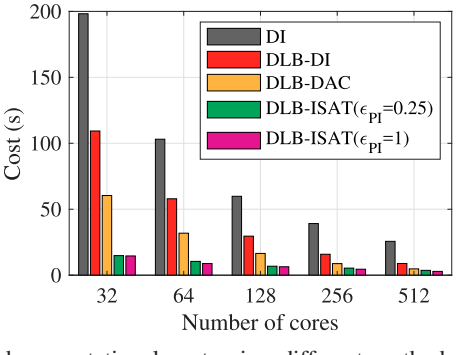


Figure 21. The averaged computational cost using different methods and different number of cores.

Table 2. Speedup factors relative to the cost of DI using different methods with the same number of cores.

#cores	DLB-DI	DLB-DAC	DLB-ISAT ($\varepsilon_{PI} = 0.25$)	DLB-ISAT ($\varepsilon_{PI} = 1$)
64	1.78	3.24	9.85	11.66
128	2.02	3.63	8.79	9.37
256	2.47	4.46	7.30	8.62
512	2.89	5.36	7.00	8.90

used, i.e. $N_{pc} \leq 64$. However, due to the different scalability, the computational cost for DLB-DAC and DLB-ISAT have become comparable when the number of cores increases to 512, while the relative performance between DLB-DI and DLB-DAC remains fairly constant. Based on these results, the speedup factors relative to the corresponding cost of DI using the same number of cores are summarised in Table 2. DLB-DI and DLB-DAC show higher speedup factors when more cores are used due to better scalability than DI, whereas the speedup of DLB-ISAT drops with more cores since the parallel efficiency is lower.

6. Conclusion

Parallel simulations of turbulent combustion may impose imbalanced load distribution due to unbalanced reactive volumes and differing stiffness of the ODEs in different regions of the flame. A DLB method employing load redistribution has been proposed in the present study to alleviate this issue. The DLB method employs a decomposition in mixture fraction space to provide a more precise estimation of computational cost based on which load is evenly redistributed among computing cores. This method also increases the locality of ISAT tables and hence the computational efficiency of ISAT. Two dynamic adjusting algorithms are proposed namely global and diffusive adjusting methods. The global method is found to be sufficient for the DLB-DI and DLB-DAC methods, while a combination of the global and diffusive methods is the best approach when using ISAT as it achieves the best compromise between fast response to load imbalances and stable performance avoiding too frequent changes in the accessed region in composition space on each processor.

The DLB methods have been tested against two laboratory flames. The Sydney piloted flame with homogeneous inlet composition presents a special case where the computational load distribution is achieved rather efficiently by conventional spatial domain decomposition. The tests for this flame case validate the DLB implementation even though there is little margin for improvement. The Delft III flame simulation exhibits appreciable load imbalance by conventional spatial domain decomposition. The DLB-DI and DLB-DAC methods achieve excellent load balancing compared to those without DLB. While the global adjusting method causes significant performance oscillation for DLB-ISAT, the combined global and diffusive approach works well. A parameter ε_{PI} is introduced to control the timing of infrequent activation of the global adjusting method amongst regular, steady and diffusive load balancing. Although performance differences for different ε_{PI} are observed, the influence on the overall computational time is small. It is found that the DLB-ISAT method can achieve even shorter computational time than an ideally balanced

load distribution using ISAT. This is attributed to the joint benefits from the better usage of computing resources and localised tabulation in mixture fraction space.

Significantly improved parallel scalability of the DLB-DI and DLB-DAC methods are observed compared to DI, whereas the DLB-ISAT presents considerable efficiency drops starting from 128 cores. For all the methods, the decline of parallel scaling efficiency is mostly attributed to the degraded balancing operation stemming from the stochastic fluctuations of computational cost since fewer notional particles are solved on each core. In all cases, the communication cost only accounts for a reasonably small fraction of the total computational cost. Lastly, although the DLB-ISAT method shows lower scalability than the DLB-DI and DLB-DAC, it still has the lowest computational cost of the various algorithms in these test cases.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was financially supported by the National Key Project [grant number GJXM92579] and the Australian Research Council [grant number DP180104190].

ORCID

Matthew J. Cleary http://orcid.org/0000-0003-1558-7222

References

[1] S.B. Pope, *PDF methods for turbulent reactive flows*, Prog. Energy Combust. Sci. 11 (1985), pp. 119–192.
[2] P.J. Colucci, F.A. Jaberi, P. Givi, and S.B. Pope, *Filtered density function for large eddy simulation of turbulent reacting flows*, Phys. Fluids 10 (1998), pp. 499–515.
[3] F. Jaberi, P. Colucci, S. James, P. Givi, and S. Pope, *Filtered mass density function for large-eddy simulation of turbulent reacting flows*, J. Fluid Mech. 401 (1999), pp. 85–121.
[4] N. Peters, *Laminar diffusion flamelet models in non-premixed turbulent combustion*, Prog. Energy Combust. Sci. 10 (1984), pp. 319–339.
[5] A.Y. Klimenko and R.W. Bilger, *Conditional moment closure for turbulent combustion*, Prog. Energy Combust. Sci. 25 (1999), pp. 595–687.
[6] A.Y. Klimenko and S.B. Pope, *The modeling of turbulent reactive flows based on multiple mapping conditioning*, Phys. Fluids 15 (2003), pp. 1907–1925.
[7] M.J. Cleary and A.Y. Klimenko, *A detailed quantitative analysis of sparse-Lagrangian filtered density function simulations in constant and variable density reacting jet flows*, Phys. Fluids 23 (2011), pp. 115102.
[8] G. Neuber, F. Fuest, J. Kirchmann, A. Kronenburg, O.T. Stein, S. Galindo-Lopez, M.J. Cleary, R.S. Barlow, B. Coriton, J.H. Frank, and J.A. Sutton, *Sparse-Lagrangian MMC modelling of the sandia DME flame series*, Combust. Flame 208 (2019), pp. 110–121.
[9] Y. Ge, M.J. Cleary, and A.Y. Klimenko, *A comparative study of sandia flame series (D–F) using sparse-Lagrangian MMC modelling*, Proc. Combust. Inst. 34 (2013), pp. 1325–1332.
[10] S. Galindo, F. Salehi, M.J. Cleary, and A.R. Masri, *MMC-LES simulations of turbulent piloted flames with varying levels of inlet inhomogeneity*, Proc. Combust. Inst. 36 (2017), pp. 1759–1766.
[11] Z. Huo, F. Salehi, S. Galindo-Lopez, M.J. Cleary, and A.R. Masri, *Sparse MMC-LES of a sydney swirl flame*, Proc. Combust. Inst. 37 (2019), pp. 2191–2198.
[12] G. Neuber, A. Kronenburg, O.T. Stein, and M.J. Cleary, *MMC-LES modelling of droplet nucleation and growth in turbulent jets*, Chem. Eng. Sci. 167 (2017), pp. 204–218.

[13] S. Vo, A. Kronenburg, O.T. Stein, and M.J. Cleary, *Multiple mapping conditioning for silica nanoparticle nucleation in turbulent flows*, Proc. Combust. Inst. 36 (2017), pp. 1089–1097.
[14] G. Neuber, A. Kronenburg, O.T. Stein, C.E. Garcia, B.A.O. Williams, F. Beyrau, and M.J. Cleary, *Sparse-Lagrangian PDF modelling of silica synthesis from silane jets in vitiated co-flows with varying inflow conditions*, Flow, Turbul. Combust. 106 (2021), pp. 1167–1194.
[15] G. Wanner and E. Hairer, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, Springer, Berlin, 1996.
[16] S. Pope, *Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation*, Combust. Theory Modell. 1 (1997), pp. 41–63.
[17] L. Liang, J.G. Stevens, and J.T. Farrell, *A dynamic adaptive chemistry scheme for reactive flow computations*, Proc. Combust. Inst. 32 (2009), pp. 527–534.
[18] A. Niemoeller, M. Schlottke-Lakemper, M. Meinke, and W. Schroeder, *Dynamic load balancing for direct-coupled multiphysics simulations*, Comput. Fluids 199 (2020), pp. 104437.
[19] S. Herff, A. Niemöller, M. Meinke, and W. Schröder, *Les of a turbulent swirl flame using a mesh adaptive level-set method with dynamic load balancing*, Comput. Fluids 221 (2021), pp. 104900.
[20] H. Sitaraman and R. Grout, *Balancing conflicting requirements for grid and particle decomposition in continuum-lagrangian solvers*, Parallel Comput. 52 (2016), pp. 1–21.
[21] A. Amritkar, S. Deb, and D. Tafti, *Efficient parallel cfd-dem simulations using openmp*, J. Comput. Phys. 256 (2014), pp. 501–519.
[22] A. Thari, N.C. Treleaven, M. Staufer, and G.J. Page, *Parallel load-balancing for combustion with spray for large-scale simulation*, J. Comput. Phys. 434 (2021), pp. 110187.
[23] S. Yakubov, B. Cankurt, M. Abdel-Maksoud, and T. Rung, *Hybrid mpi/openmp parallelization of an euler–lagrange approach to cavitation modelling*, Comput. Fluids 80 (2013), pp. 365–371.
[24] G. Houzeaux, M. Garcia, J.C. Cajas, A. Artigues, E. Olivares, J. Labarta, and M. Vázquez, *Dynamic load balance applied to particle transport in fluids*, Int. J. Comput. Fluid Dyn. 30 (2016), pp. 408–418.
[25] B. Tekgül, P. Peltonen, H. Kahila, O. Kaario, and V. Vuorinen, *Dlbfoam: an open-source dynamic load balancing model for fast reacting flow simulations in openfoam*, Comput. Phys. Commun. 267 (2021), pp. 108073.
[26] J. Muela, R. Borrell, J. Ventosa-Molina, L. Jofre, O. Lehmkuhl, and C.D. Pérez-Segarra, *A dynamic load balancing method for the evaluation of chemical reaction rates in parallel combustion simulations*, Comput. Fluids 190 (2019), pp. 308–321.
[27] A.Y. Klimenko, *Lagrangian particles with mixing. II. Sparse-Lagrangian methods in application for turbulent reacting flows*, Phys. Fluids 21 (2009), pp. 065102.
[28] S. Vo, O.T. Stein, A. Kronenburg, and M.J. Cleary, *Assessment of mixing time scales for a sparse particle method*, Combust. Flame 179 (2017), pp. 280–299.
[29] P. Pepiot-Desjardins and H. Pitsch, *An efficient error-propagation-based reduction method for large chemical kinetic mechanisms*, Combust. Flame 154 (2008), pp. 67–81.
[30] L. Lu and S.B. Pope, *An improved algorithm for in situ adaptive tabulation*, J. Comput. Phys. 228 (2009), pp. 361–386.
[31] S.B. Pope, *Algorithms for ellipsoids*, Cornell University Report No. FDA 2008, pp. 08–01
[32] I. Veljkovic, P.E. Plassmann, and D.C. Haworth, *A scientific on-line database for efficient function approximation*, in *Computational Science and Its Applications — ICCSA 2003. Lecture Notes in Computer Science*, Vol. 2667. Kumar V., Gavrilova M.L., Tan C.J.K., L'Ecuyer P., eds., Springer, Berlin, Heidelberg, 2003. pp. 643–653.
[33] S. Galindo-Lopez, F. Salehi, M.J. Cleary, A.R. Masri, G. Neuber, O.T. Stein, A. Kronenburg, A. Varna, E.R. Hawkes, B. Sundaram, and A.Y. Klimenko, *A stochastic multiple mapping conditioning computational model in OpenFOAM for turbulent combustion*, Comput. Fluids 172 (2018), pp. 410–425.
[34] M. Muradoglu, S.B. Pope, and D.A. Caughey, *The hybrid method for the PDF equations of turbulent reactive flows: consistency conditions and correction algorithms*, J. Comput. Phys. 172 (2001), pp. 841–878.
[35] R. Barlow, S. Meares, G. Magnotti, H. Cutcher, and A. Masri, *Local extinction and near-field structure in piloted turbulent ch4/air jet flames with inhomogeneous inlets*, Combust. Flame 162 (2015), pp. 3516–3540.

[36] S. Meares and A.R. Masri, *A modified piloted burner for stabilizing turbulent flames of inhomogeneous mixtures*, Combust. Flame 161 (2014), pp. 484–495.
[37] T.W.J. Peeters, P.P.J. Stroomer, J.E. De Vries, D.J.E.M. Roekaerts, and C.J. Hoogendoorn, *Comparative experimental and numerical investigation of a piloted turbulent natural-gas diffusion flame*, Symp. (Int.) Combust. 25 (1994), pp. 1241–1248.
[38] P.A. Nooren, M. Versluis, T.H. van der Meer, R.S. Barlow, and J.H. Frank, *Raman-Rayleigh-LIF measurements of temperature and species concentrations in the delft piloted turbulent jet diffusion flame*, Appl. Phys. B 71 (2000), pp. 95–111.
[39] G.P. Smith, D.M. Golden, M. Frenklach, N.W. Moriarty, B. Eiteneer, M. Goldenberg, C.T. Bowman, R.K. Hanson, S. Song, and W.C. Gardiner, *The GRI 3.0 Chemical Kinetic Mechanism*, Gas Research Institute, Chicago, IL, 1999. Available at http://www.me.berkeley.edu/gri_mech.
[40] M.E. Mueller and H. Pitsch, *LES model for sooting turbulent nonpremixed flames*, Combust. Flame159 (2012), pp. 2166–2180.
[41] P. Donde, V. Raman, M.E. Mueller, and H. Pitsch, *LES/PDF based modeling of soot–turbulence interactions in turbulent flames*, Proc. Combust. Inst. 34 (2013), pp. 1183–1192.