scientific reports

Check for updates

OPEN

A fast, accurate and oscillation-free spectral collocation solver for high-dimensional transport problems

Nicola Cavallini[1], Gianmarco Manzini[2], Daniele Funaro[3,4] & Andrea Favalli[1,2]✉

Transport phenomena—describing the movement of particles, energy, or other physical quantities—are fundamental in various scientific disciplines, including nuclear physics, plasma physics, astrophysics, engineering, and the natural sciences. However, solving the associated seven-dimensional transport equations poses a significant computational challenge due to the curse of dimensionality. We introduce the Tensor Train Superconsistent Spectral ($\mathrm{T^2S^2}$) solver to address this challenge, integrating Spectral Collocation for exponential convergence, Superconsistency for stabilization in transport-dominated regimes, and Tensor Train format for substantial data compression. $\mathrm{T^2S^2}$ enforces a dimension-wise superconsistent condition compatible with tensor structures, achieving extremely low compression ratios, such as $\mathscr{O}(\mathbf{10^{-12}})$, while preserving spectral accuracy. Numerical experiments on linear problems demonstrate that $\mathrm{T^2S^2}$ can solve high-dimensional transport problems in minutes on standard hardware, making previously intractable problems computationally feasible. This advancement opens new avenues for efficiently and accurately modeling complex transport phenomena.

Transport phenomena are fundamental across a wide range of scientific disciplines, including nuclear physics, plasma physics, high-energy density physics, astrophysics, condensed matter physics, atmospheric science, oceanography, and engineering. These processes describe the movement and interaction of particles, energy, and other physical quantities within various media. At the heart of modeling these phenomena are transport equations, which comprehensively describe how quantities propagate and interact. For example, the neutron transport equation is fundamental in applications such as reactor design and radiation detection instrumentation, while the Vlasov equation is pivotal in plasma physics, describing the evolution of charged particles under electromagnetic fields, and is fundamental in applications ranging from fusion to space science.[1–6] Although the precise nature of transport phenomena may vary across applications, the underlying mathematical structure remains consistent. A significant challenge in solving transport equations is the associated computational cost, given that a general solution spans seven dimensions: three spatial coordinates, two angles, one energy (or speed), and one time. As the number of dimensions increases, the required memory and computational resources scale exponentially, limiting the feasibility of traditional numerical methods. The challenge has been aptly described by F. Graziani and G. Olson as "Conquering the Seven-Dimensional Mountain"[7].

To establish a robust computational framework, we focus on a fundamental yet representative model of transport phenomena: the linear, time-dependent, convection-diffusion-reaction PDE. This is given by:

$$\frac{\partial f}{\partial t} - \varepsilon \Delta_{\mathbf{x}} f + \boldsymbol{\beta} \cdot \nabla_{\mathbf{x}} f + \rho f = b, \qquad \forall (\mathbf{x}, \mathbf{t}) \in \boldsymbol{\Omega} \times]\mathbf{0}, \mathbf{T}] \ , \tag{1}$$

the equation describes the evolution of the distribution function $f(\mathbf{x}, \mathbf{t})$ in the seven-dimensional phase space $\Omega \times [0, T]$, with $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_d})^{\mathbf{T}} \in \boldsymbol{\Omega} \subset \mathbb{R}^{\mathbf{d}}$, $d = 6$. Time is represented by $t \times [0, T]$ and x_1, x_2, x_3 refer to the 3-D spatial variables. Depending on the specific form of Eq. (1) x_4, x_5, x_6 either describe the momentum vector or x_4, x_5 represent the 2-D angular direction, and x_6 the energy or speed. The computational domain $\Omega \subset \mathbb{R}^d$ is the d-dimensional, open hypercube with characteristic edge length L and five-dimensional boundary Γ. To have a mathematically well-posed model, we assume that f satisfies Dirichlet boundary conditions on the domain boundary

[1]European Commission, Joint Research Centre, Via Enrico Fermi, 21027 Ispra, Italy. [2]Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545, USA. [3]Dipartimento di Scienze Chimiche e Geologiche, Università di Modena e Reggio Emilia, Via Campi 103, 41125 Modena, Italy. [4]Istituto di Matematica e Applicata e Tecnologie Informatiche del CNR, Via Ferrata 1, 27100 Pavia, Italy. ✉email: andrea.favalli@ec.europa.eu

$$f = g \quad \text{on } \Gamma \times [0, T], \tag{2}$$

and an initial condition at $t = 0$

$$f(\cdot, 0) = f_0 \quad \text{in } \Omega. \tag{3}$$

In Eq. (1), ε is the diffusion coefficient, $\boldsymbol{\beta}$ is the convection field, ρ is the reaction field, and b is the right-hand side forcing term. As a matter of notation, in the latter we drop the $\mathbf{x}$ subscript in the differential operators notation: $\nabla_{\mathbf{x}}$ becomes ∇ and $\Delta_{\mathbf{x}}$ becomes Δ.

The numerical solution of high-dimensional problems in the framework of Partial Differential Equations (PDEs) faces a fundamental challenge often referred to as the *curse of dimensionality*[8], where both memory requirements and computational costs scale exponentially with the number of dimensions. This is a serious limit for the effectiveness of numerical methods. While recent approaches have attempted to address this challenge through GPU acceleration[9] and machine learning methods[10,11], these solutions introduce further complexities, as they require either architecture-specific programming or extensive training data.

To address these computational challenges, we take a different path based on a three-level approach. First, we operate in the context of the *low-rank approximation*[12]. We adopt the Tensor Train format[13], a linear algebra framework capable of efficiently handling high-dimensional data and mathematical operators, organized on tensor-product grids. Considerable effort has been devoted over the years to solving PDEs in Tensor Train format. Dolgov and coauthors tackled the three-dimensional Fokker-Planck equation[14]. Basic numerical schemes, such as uniform finite differences applied to the high-dimensional Poisson equation, are commonly used as testbed for the development of the Tensor Train linear system solvers[15,16]. A hybrid approach for parametric PDEs in two dimensions can be found in[17]. In the context of semi-Lagrangian solvers, where special care has to be devoted to conservation of mass, energy, or momentum, Kormann[18] proposed a Tensor Train solver for Vlasov Equations.

Second, we numerically discretize the problem (1) using a variant of Spectral Collocation methods. The term "collocation" indicates that the continuous model, i.e. the PDE, is evaluated, "collocated", at specific grid points. These techniques are known for their long-recognized exceptional *accuracy*[19,20], commonly exhibiting an exponential rate of convergence for smooth solutions. This rate of convergence is also addressed as spectral accuracy. However, their application to higher-dimensional problems, in standard algebra format, has been constrained by the dense structure of the discrete operators and the complexity of solving the resulting linear systems[19,21].

Third, these methods face a fundamental trade-off between *accuracy* and *stability*[22,23]. This becomes particularly problematic in regimes dominated by first-order derivatives ($\varepsilon/|\boldsymbol{\beta}L| \ll 1$), where solutions present spurious oscillations that make the result unreliable. Funaro's concept of superconsistency[24] cures such oscillations by performing collocation on modified nodes. The new position of the nodes is a function of the ratio $\varepsilon/|\boldsymbol{\beta}L|$ and the grid density[21,25,26]. We should note that superconsistency alone does not reduce the computational costs; the solution of high-dimensional problems remains prohibitive.

Our approach, *the Tensor Train Superconsistent Spectral* ($\mathrm{T^2S^2}$) method, builds on the three techniques described above. It addresses the limitations of each individual block through a novel dimension-wise application of the superconsistent method. This approach fundamentally redefines the original superconsistent framework, making it compatible with the Tensor Train format and enabling an efficient storage of the mathematical operators. The method preserves the hallmark exponential accuracy of Spectral Collocation methods, while maintaining the stabilization properties with respect to the $\varepsilon/|\boldsymbol{\beta}L|$ ratio. At the same time $\mathrm{T^2S^2}$ achieves extremely low compression ratios, such as $\mathcal{O}(10^{-12})$. As a result, it can solve high-dimensional transport problems in minutes on standard hardware, rendering previously intractable problems computationally feasible.

The paper is organized as follows. The first section, "Overview of the $\mathrm{T^2S^2}$ Approach", presents the method, followed by the second section, "Results", which demonstrates the validation of the method through numerical experiments. The third section, "Discussion", summarizes the method's strengths, applications and future directions. The fourth section, "Methods", provides the detailed mathematical framework underlying the $\mathrm{T^2S^2}$ solver.

Overview of the $\mathrm{T^2S^2}$ approach

This section outlines the key components of our methodology and demonstrates how they synergize to create our computational framework. We illustrate the underlying principles through one- and two-dimensional examples, while the detailed mathematical framework is provided in section "Methods".

(i) **Spectral collocation methods.** Spectral collocation relies on two fundamental elements: *basis functions* and *representation points*. We employ Lagrange basis functions defined on $n + 1$ representation points[21], which are the $n - 1$ zeros of P_n', the derivative of the n-th degree Legendre polynomial P_n, complemented with the two extrema of the interval $[-1, 1]$. We will refer to this approach as *plain spectral collocation* (Plain-SC) method. The solution to a one-dimensional transport problem is a linear combination of these basis functions. The coefficients are determined by solving the linear system obtained by *collocating the equation* at the same representation points, i.e., by evaluating the left-hand side and the right-hand side of Eq. (1) at such locations. For nomenclature clarity, in one dimension, the number of degrees of freedom ($\#\mathrm{dofs}_{1\mathrm{d}}$) equals $n + 1$. The multidimensional extension of the method is achieved through the tensor product of the one-dimensional discretizations along all the problem's directions, $\#\mathrm{dofs}$ addresses the total number of degrees of freedom.

(ii) **Superconsistent formulation.** To illustrate the concept of superconsistency we consider the time-independent one-dimensional model

$$\beta f'(x) - \varepsilon f''(x) = 1 \quad x \in \left[-1, 1\right], \quad f(-1) = f(1) = 0, \tag{4}$$

where $\beta > 0$ and $\varepsilon > 0$ are the scalar convective and diffusion coefficients, respectively. The Plain-SC method, employing the zeros of P_n', (blue dots on the top axis of Fig. 1), develops spurious oscillations when the convective term becomes dominant relatively to the diffusive one, meaning that $\varepsilon/|\beta L| \ll 1$. This effect is clearly visible in panels (c) and (d) of Fig. 1.

Funaro[24] cured this phenomenon by introducing the superconsistent condition. The superconsistent condition takes into account the differential operator and is satisfied by introducing a different set of *collocation points* where the partial differential equation and the right-hand side are evaluated.

According to the superconsistent condition, the zeros of P_n' are optimal for the approximation of the diffusion operator $\varepsilon f'$, while the zeros of P_n are optimal for $\beta f'$. In hybrid regimes, the superconsistent points establish an optimal intermediate distribution between these two configurations. As illustrated in Fig. 1, when convection and diffusion terms are balanced (Fig. 1a), the superconsistent points are close to the zeros of P_n'. As the diffusion coefficient ε decreases maintaining n fixed (Fig. 1b, c), the superconsistent points shift towards the zeros of P_n. In the convection-dominated regime (Fig. 1d), the superconsistent points nearly coincide with the zeros of P_n.

In summary, the superconsistent approach is based on two grids: the *representation grid* and the actual *collocation grid*. The position of the collocation grid depends on the ratio $\varepsilon/|\beta L|$ and n. Enforcing the superconsistent condition provides the numerical strategy to locate the collocation points in order to achieve stabilization with respect to $\varepsilon/|\beta L|$. We use the representation points to build the Lagrange basis functions. These basis functions are used to evaluate the discrete operators at the collocation grid. The result of this *collocation* operation, together with boundary conditions, leads to the construction of the final linear system.

(iii) **Tensor-train format.** Traditional spectral methods struggle with the *curse of dimensionality*[8], where computational complexity increases by scaling exponentially with the number of dimensions.

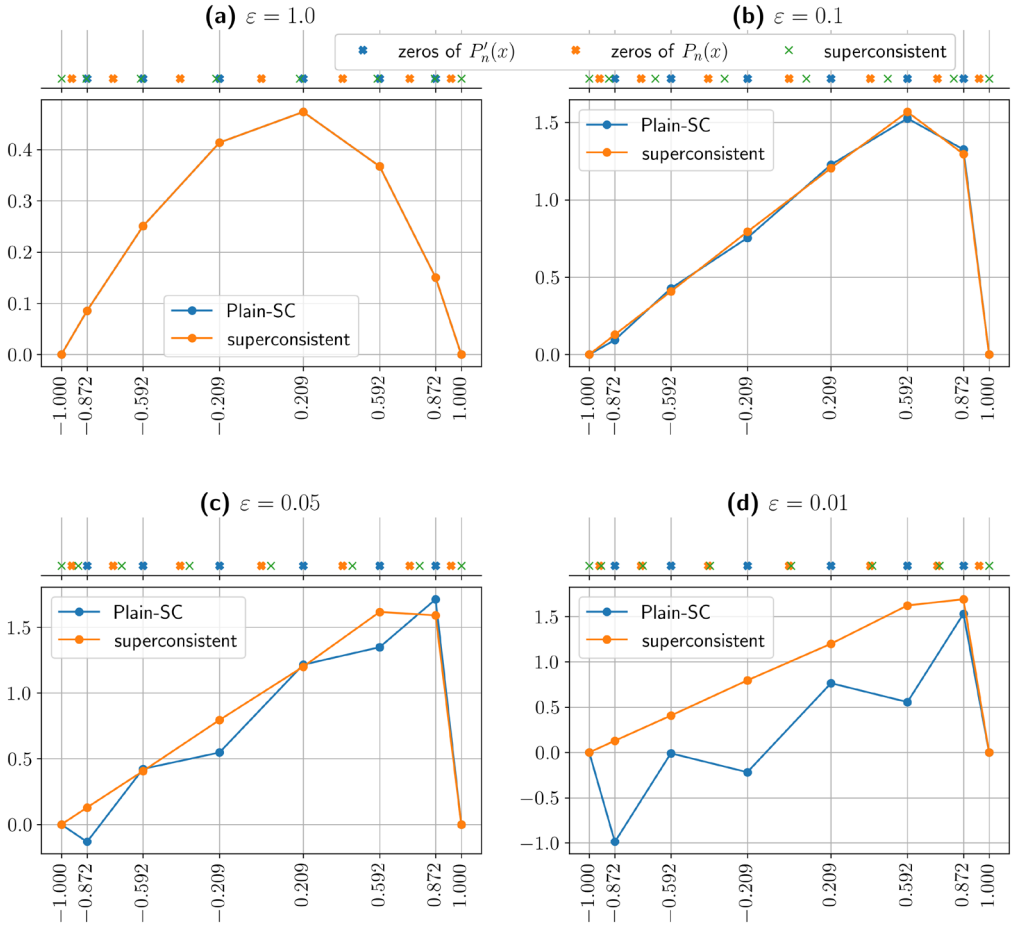


Fig. 1. Superconsistent approximation of Eq.(4) for the polynomial degree $n = 7$, and $\beta = 1$. On the top horizontal axis, the blue crosses mark the zeros of P_n', optimal for diffusion; the orange crosses mark the zeros of P_n, optimal for convection; the green crosses mark the superconsistent collocation points for mixed regimes. **(a,b)** Relatively accurate solutions when convection and diffusion are balanced, in particular in panel **(a)** the two solutions almost coincide. As convection dominates **(c,d)**, the Plain-SC solution is unreliable, whereas the superconsistent one does not present problems.

The Tensor-train (TT) format[13] efficiently represents a d-dimensional tensor as the product of d three-dimensional arrays, the *TT cores*, significantly reducing the memory requirement. TT core, denoted by G_ℓ and indexed by $\ell = 1, 2 \ldots, d$, has dimensions $r_{\ell-1} \times n_\ell \times r_\ell$, where n_ℓ is the original dimension size also called the *mode size*, and r_ℓ are the *TT-ranks* (with $r_0 = r_d = 1$). As shown in Fig. 2 for a six-dimensional tensor, an element of the multidimensional array $\mathscr{A} \in \mathbb{R}^{n_1 \times \ldots n_6}$ is reconstructed by multiplying the corresponding core slices $G_\ell \in \mathbb{R}^{r_{\ell-1} \times n_\ell \times r_\ell}$, $\ell = 1, 2, \ldots, 6$. The TT decomposition is not unique and is most effective when the TT-ranks are much smaller than the mode size, i.e., for $r_\ell \ll n_\ell$. In such a case, this method efficiently compresses large multidimensional arrays and, when applied to operators, mitigates the high bandwidth issue arising from tensor products.

We build the $\mathrm{T^2S^2}$ solver combining principles (*i*), (*ii*), and (*iii*) to extend superconsistency to high-dimensional equations. Our approach implements the superconsistent condition independently along each dimension, treating them as decoupled problems. This formulation works with two distinct time integration frameworks: the *unified space-time formulation*[27] that handles time as an additional dimension, and the *method-of-lines formulation*[28,29] that uses conventional time-stepping schemes. For the latter, we implement the Crank–Nicolson and the backward Euler schemes. Highly accurate variants of these schemes, such as[30] are combinations of the mentioned methods. Superconsistency is applied only to the six spatial dimensions, demonstrating an oscillatory-free performance with respect to the ratio $\varepsilon/|\beta L|$.

In Fig. 3 we show a two dimensional grid obtained using $\mathrm{T^2S^2}$. The resulting Cartesian structure naturally fits with the TT format and extends straightforwardly to higher dimensions. This approach differs from Funaro's original formulation[21] , which applies superconsistency to the full multidimensional problem and produces non-Cartesian grid deformations, see Fig. 3. Our method is stable across physically relevant parameter ranges and preserves the exponential accuracy that is characteristic of the spectral collocation methods.

Results

In this section, we analyze the $\mathrm{T^2S^2}$ performance from various perspectives. First, we assess its accuracy and convergence by investigating its behavior when approximating a low-rank, high-frequency analytical solution. The high-frequency test examines $\mathrm{T^2S^2}$'s performance on a grid density with $300^7 \approx 2.2\,10^{17}$ nodes that can only be explored by low-rank solvers since a full format traditional solver, even at minimal complexity, would require millions of years on modern supercomputers. Next, we present a six-dimensional test case that triggers the $\varepsilon/|\beta L| \ll 1$ instabilities when we approximate a high-gradient solution that challenges the low-rank approximation on moderate to high-density grids. It tests the robustness of $\mathrm{T^2S^2}$ in handling such instabilities. Finally, we complete our investigation with two low-dimensional test cases that challenge the non-oscillatory performance of the $\mathrm{T^2S^2}$ solver: the "traveling bump" benchmark[21,31] and the Hughes double-layer benchmark[32,33].

Global accuracy and computational cost

Testing numerical algorithms with manufactured analytical solutions is a cornerstone practice in numerical analysis. It provides a reliable way to create benchmark solutions for validating numerical solver[34]. In this case, we construct an "exact" solution by choosing a suitable expression, and we determine the corresponding right-hand side analytically. To this end, we apply our $\mathrm{T^2S^2}$ solver to problem (1)–(3) on $\Omega \times [0, T]$ with $\Omega = [-1, 1]^6$ and $T = 1$. For convenience, we rescale the time interval to match the interval $[-1, 1]$. We set $\varepsilon = 10^{-4}$, and define:

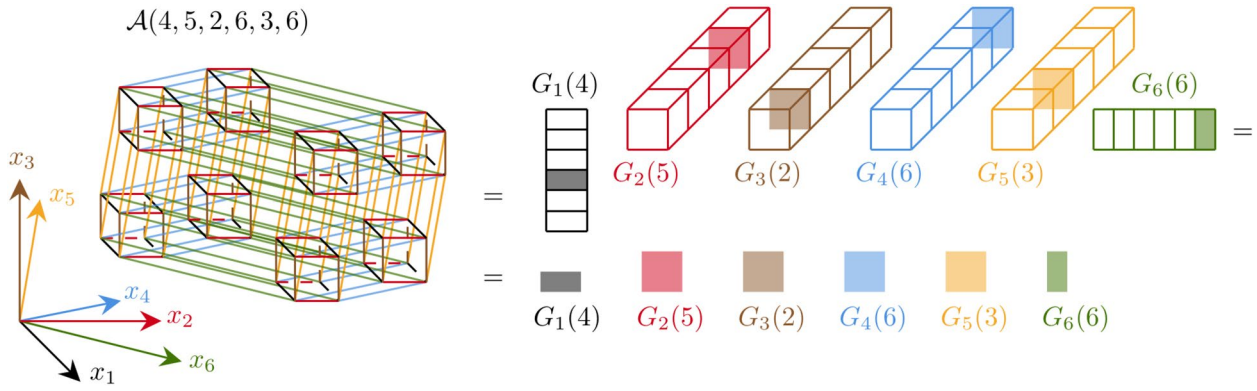


Fig. 2. Focusing on the six spatial dimensions, a six-dimensional array $\mathscr{A}$ has size n^6, with $x_1, x_2, \ldots, x_6$ coordinate axis of the six-dimensional space. The TT format represents the whole array on the left as the product of six cores on the right. The ℓ-th core for $\ell = 1, 2, \ldots, 6$ has size $r_{\ell-1} \times n \times r_\ell$, with $r_0 = r_6 = 1$, so the first and last cores are matrices. The (4, 5, 2, 6, 3, 8)-th entry of the full tensor is represented in the TT-format as the product $\mathscr{A}(4, 5, 2, 6, 3, 8) = G_1(4)G_2(5)G_3(2)G_4(6)G_5(3)G_6(8)$, i.e., as the product of the 4-th, 5-th, 2-nd, 6-th, 3-rd, 8-th slices of the corresponding cores.

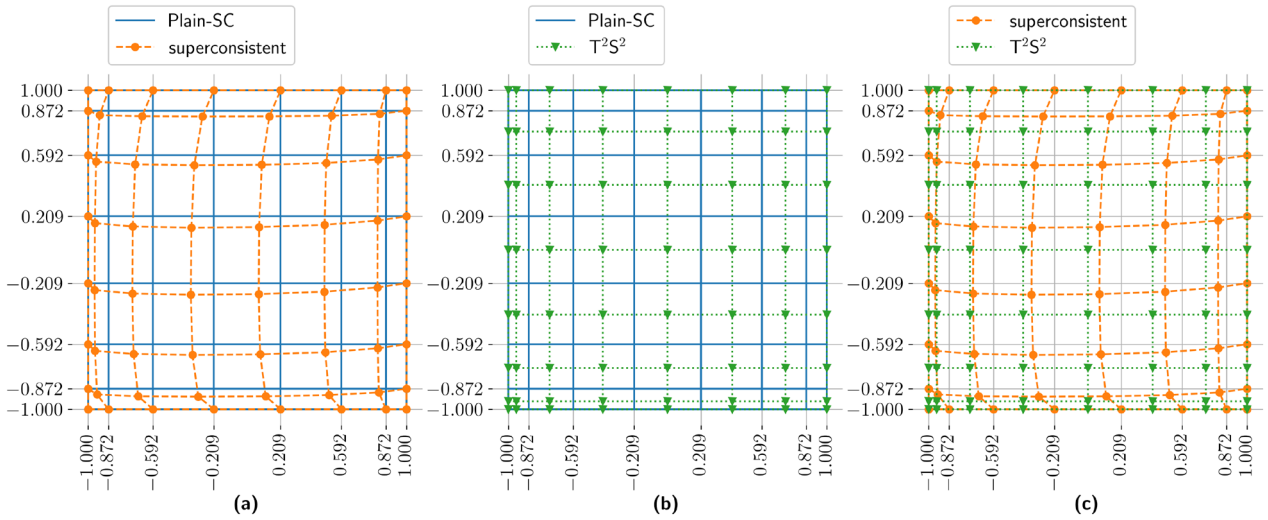


Fig. 3. Left panel: Plain-SC nodes and Funaro's superconsistent nodes[21]. Such a grid has been evaluated for problem (4) with parameters $\varepsilon = 1/50$, $\beta = (1, 0.5)^T$ and $n = 7$. Middle panel: Plain-SC nodes and $\mathrm{T}^2\mathrm{S}^2$ collocation nodes, which have been independently computed in each direction. Right panel: Comparison between the 2D Funaro's superconsistent nodes[21] and $\mathrm{T}^2\mathrm{S}^2$ collocation nodes.

$$\boldsymbol{\beta}(x_1, x_2, \ldots, x_6) = (\sin(\pi x_1), \sin(\pi x_2), \ldots, \sin(\pi x_6))^T,$$
$$\rho(x_1, x_2, \ldots, x_6) = \prod_{1 \le i \le 6} \left[\frac{1}{2} - \left(x_i + \frac{11}{10}\right)\left(x_i - \frac{1}{2}\right)\left(x_i + \frac{1}{2}\right)\left(x_i - \frac{11}{10}\right)\right].$$

The right-hand side b and the Dirichlet boundary conditions are determined from the proposed analytical solution:

$$f(\mathbf{x}, t) = e^t \cdot \prod_{1 \le i \le 6} \left(\sin(\pi x_i) + 0.3 \sin(80\pi x_i)\right).$$

Here, we test the space-time formulation. In this setting, the initial condition is naturally incorporated as a boundary condition at $t = -1$, while at $t = 1$, the numerical solution is a result of the numerical scheme. The analytical solution $f(\mathbf{x}, t)$ incorporates two spatial components: a low-frequency one, i.e. $\sin(\pi\cdot)$, that spans the computational domain, and a high-frequency one, i.e. $\sin(80\pi\cdot)$, that introduces features at approximately 1/100 of the domain size. Such multi-scale phenomena are commonly encountered in practical applications[35,36].

The results are presented in Fig. 4. Each "dot" represents an approximation of the solution of problem (1) for a given polynomial degree (associated with the number of degrees of freedom). The convergence tests are based on the following L^2-type error norm:

$$\text{error} = \frac{\|f - f_n\|_{\mathscr{F}}}{\|f\|_{\mathscr{F}}},$$

where $\|\cdot,\cdot,\|_{\mathscr{F}}$ denotes the Frobenius norm and f_n represents the polynomial approximation of degree n of the solution f.

Panel (b) shows that we need a grid of approximately 256^7 size to capture the two dominant frequencies of the solution. From that point onwards, we can appreciate the exponential decay of the error characteristic of the spectral methods. The solver achieves an $\mathscr{O}(10^{-11})$ approximation error for a grid size having a total number of 300^7 nodes (including time). Panel (a) shows that $\mathrm{T}^2\mathrm{S}^2$ achieves the desired accuracy in under three minutes on an Intel i9 laptop computer with 64 GBytes RAM. We put this number into perspective theorizing a full format Super Spectral Collocation solver with minimal quadratic complexity $((\#\mathrm{dofs}_{1\mathrm{d}})^7)^2$, labeled *full format solver*[21,24]. We estimate that a modern supercomputer, characterized by a peak performance of 2.7^{18} floating point operations per second[37], would require approximately fifty million years to run the full format solver on the 256^7 grid. The computational cost for the first grid that exhibits spectral accuracy, the one with 300^7 total degrees of freedom, would approximately be 300 million years. These two timescales respectively compare with the geological age of the Himalayas and the Dolomites. These results are a consequence of the fact that $\mathrm{T}^2\mathrm{S}^2$ can capture a low-rank solution of a maximum rank of 9, resulting in a compression ratio of $(7 \cdot 9^2 \cdot 300)/300^7 = \mathscr{O}(10^{-12})$. The corresponding memory requirement for the full format solver exceeds one exabyte, indicated by the arrow in Fig. 4(a). One exabyte compares at best, or exceeds, the storage capacity of todays largest supercomputers[37]. The small drop in the computational time for $\mathrm{T}^2\mathrm{S}^2$ depends on the fact that

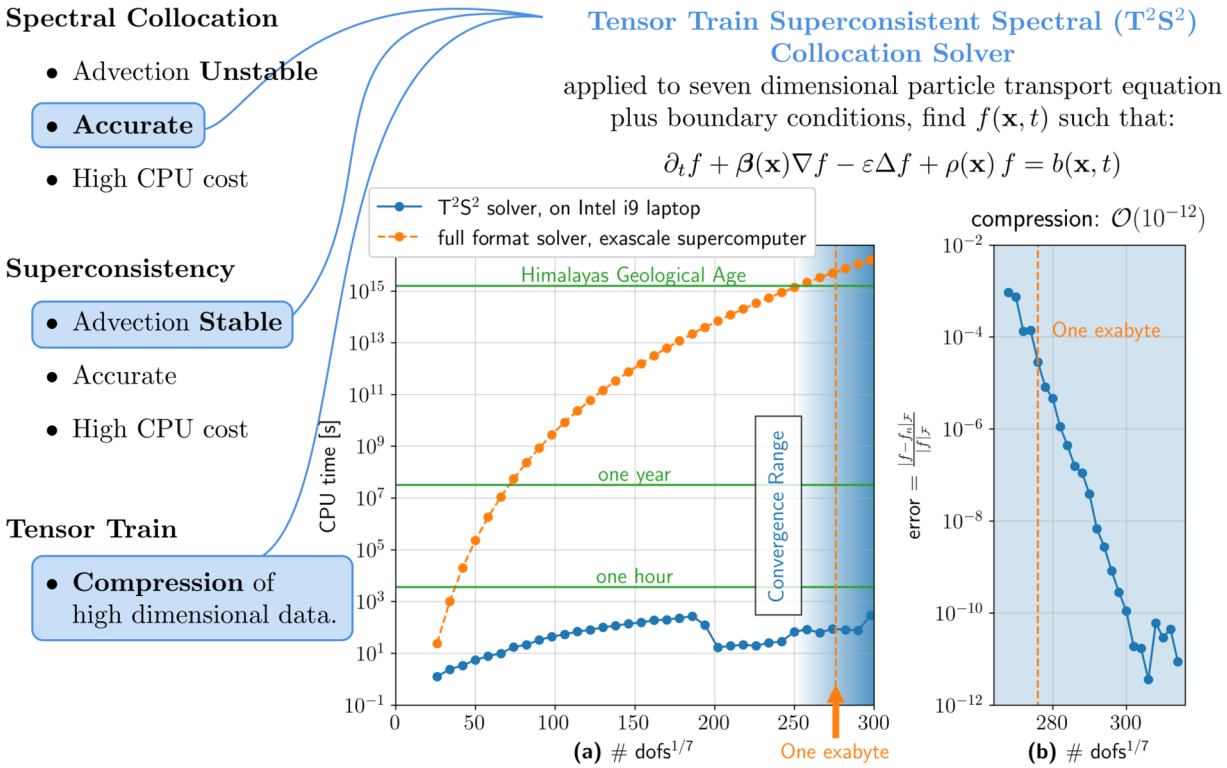


Fig. 4. This figure showcases the key components of T^2S^2 and its performance in solving the manufactured transport problem. (**a**) Illustrates T^2S^2's compression capabilities, which enable the real-time solution of a seven-dimensional space-plus-time problem. To convey to the reader the scale of the performance, we provide a comparison with the estimated computational time for a full format solver with minimal numerical complexity[19,21], whose cost scales as $\mathcal{O}((\#\text{dofs}_{1d})^7)^2$. Assuming such a full format solver runs on a 2.7 exaFLOPS nominal supercomputer, the projected execution time would be on the order of fifty million years, comparable to the geological age of the Himalayas. An arrow is used to indicate the "One exabyte" memory for the full format solver. (**b**) The convergence behavior. The error decreases exponentially to $\mathcal{O}(10^{-11})$.

the solver, on the finer meshes, is closed to the real solution and in general needs less iterations, or swaps in the TT nomenclature, to get to convergence.

Figure 5 shows the convergence analysis of T^2S^2 on a six-dimensional, constant coefficient test case. We set $\varepsilon = 10^{-6}$, $\boldsymbol{\beta}$ as a six-dimensional unit vector and $\rho = 0$. The analytical solution is prescribed as

$$f(\mathbf{x},t) = e^{-t} \cdot \prod_{1\le i\le 6} (\sin(\pi x_i)),$$

from this, we derive the corresponding right-hand side and boundary condition for the discrete model. Figure 5 presents the error curve as a function of the number of degrees of freedom along one dimension. The results demonstrate that the method preserves spectral accuracy.

Superconsistent stabilization of convection-dominated problems

Stabilization techniques for convection-diffusion problems have been extensively studied in the literature[32,33], mainly focusing on low dimensional stationary convection-diffusion equation with Dirichlet boundary conditions. We consider a six-dimensional equivalent:

$$\begin{aligned} -\varepsilon\Delta f + \boldsymbol{\beta}\cdot\nabla f &= b \quad \text{in } \Omega, \\ f &= 0 \quad \text{on } \Gamma. \end{aligned} \tag{5}$$

This equation is a specialized case of our target problem (1), which we analyze on the six-dimensional hypercube domain $\Omega = [-1,1]^6$. Conventional spectral approximations perform effectively in diffusion-dominated regimes where $\varepsilon\Delta f$ predominates over $\boldsymbol{\beta}\cdot\nabla$. However, they exhibit spurious oscillations in convection-dominated scenarios which are common in many practical applications[35,38,39].

For our numerical experiments, we take $b = 1$ and define $\boldsymbol{\beta}$ as a vector of ones, and homogeneous boundary conditions are applied. In Fig. 6, we vary the degree of the space discretizations and ε. We compare T^2S^2 with the Plain-SC stored in TT format, we label this method Plain-SC-TT. The results are displayed on colormaps (c) and (d), where red pixels indicate an oscillatory solution; green pixels represent a non-oscillatory solution. We

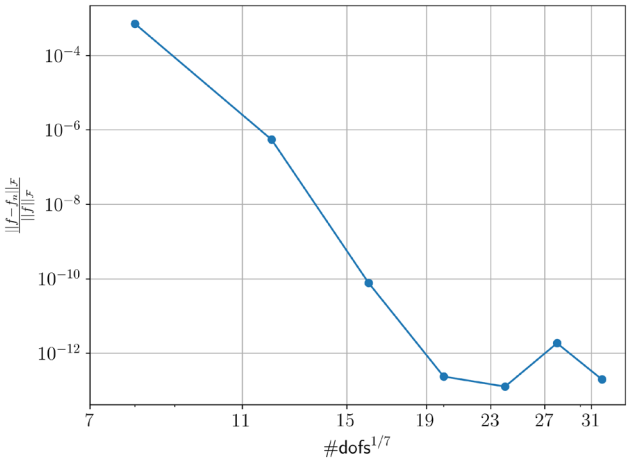


Fig. 5. This convergence curve shows the exponential decay of the approximation error of the numerical solution versus the number of degrees of freedom in each direction from the application of $\mathrm{T^2S^2}$ to a seven-dimensional convection-diffusion problem with velocity field $\boldsymbol{\beta} = (1, 1, \ldots, 1)^T$ and diffusion coefficient $\varepsilon = 10^{-6}$.

count the number of sign changes in the first derivative of the numerical solution. We compare this number with the expected sign changes in the regular solution. If the first count exceeds the second, the solution is marked as oscillatory. Oscillations are evaluated on the mid-axis along the first direction, at the central degree of freedom.

Panels (a), (b) of Fig. 6 show a two-dimensional slice of the six-dimensional solution computed with $\varepsilon = 10^{-5}$ and 40 degrees of freedom. Panel (b) presents the non-oscillatory solution, which exhibits a continuous profile with a well-resolved boundary layer at the domain perimeter. In contrast, panel (a) shows the oscillatory solution, where spurious oscillations dominate the numerical approximation, rendering the spatial discretization ineffective. Colormap in Fig. 6(c) shows that the Plain-SC-TT approach is unstable for a wide range of parameters. The stable/unstable interface follows a precise law; diffusive terms scale like $\mathcal{O}(\#\text{dofs}_{1\text{d}}^4)$, while transport terms scale like $\mathcal{O}(\#\text{dofs}_{1\text{d}}^2)$, see section "Methods".

It is, in principle, true that even for a very small value of $\varepsilon/|\boldsymbol{\beta}L|$, we can find a sufficiently refined mesh where the diffusion term becomes numerically dominant, and the Plain-SC-TT approximation is stable. This approach is unfortunately illusive, since it is always possible to choose smaller values of $\varepsilon/|\boldsymbol{\beta}L|$ that result in spurious oscillations in the solution. Moreover, the computational capability of TT linear solvers is remarkable but not infinite. AMEn[15], an advanced TT linear system solver requires solving local problems at each TT core iteration, with each local direct solution having complexity $\mathcal{O}((r_{\ell-1}, n_\ell, r_\ell)^3)$, where $r_{\ell-1}, r_\ell$ denote the TT ranks and n_ℓ the mode size of the corresponding core. This results in a challenging computational limit on the maximum possible resolution determined by the hardware's RAM. This is why $\mathrm{T^2S^2}$ comes into play. The super-consistent discretization is always unconditionally stable regardless of the choice of the parameters; see Fig. 6(d).

The boundary layer in Fig. 6(b), characterized by a high gradient connecting the domain solution to the zero boundary condition challenges the rank compression of the TT format, as higher gradients typically result in a more dense solution structure in terms of dominant patterns[40].

In Fig. 7, we relate the linear solver "iterations" to the TT ranks, compression factors, and residuals for $\mathrm{T^2S^2}$ and Plain-SC-TT. The TT solver is AMEn[15], and the "iterations" are addressed as swaps in the TT nomenclature. Panel (b) illustrates that the TT solver adjusts the solution rank with each swap, reaching a stationary point at swap 11 with a rank of 11. At this point, both our method and the Plain-SC-TT approach reach their resolution limit, and beyond it, they are unable to extract further information. Maximum rank 11 corresponds to a compression factor around $\mathcal{O}(10^{-6})$, cf. the details in panels (b) and (c). Although $\mathrm{T^2S^2}$ and Plain-SC-TT achieve a similar compression ratios, their convergence properties diverge substantially. The residual analysis in panel (c) reveals that $\mathrm{T^2S^2}$ maintains consistent convergence with regular residual decay while Plain-SC-TT exhibits non-convergent behavior characterized by irregular residual patterns. The final residual value of $\mathcal{O}(10^{-2})$ reflects the poor condition number of the linear system involved[19,24]. Moreover, the computational cost of $\mathrm{T^2S^2}$ for solving a problem with $\#\text{dofs} = \mathcal{O}(10^9)$ on an Intel i9 laptop computer is in the range of seconds.

Low-dimensional numerical challenges

This section presents two low-dimensional test cases to assess the characteristic capabilities of $\mathrm{T^2S^2}$. The first test is the traveling bump benchmark[21,31]. The bump benchmark is a two-dimensional time-dependent problem where the initial condition is known, but there is no a priori knowledge of the evolution of the solution. The test is based on a rotational velocity flow, that challenges the Cartesian structure of the $\mathrm{T^2S^2}$ grid. Although we choose collocation points to be the zeros of P_n, considering positive convection coefficients in both directions, $\mathrm{T^2S^2}$ nonetheless proves to be robust. We solve problem (1)–(3) with $\boldsymbol{\beta} = (-x_2, x_1)^T$, $\varepsilon = 10^{-6}$ and $\rho = 0$. The space-time domain is given by $\Omega \times T = ([-1, 1] \times [-1, 1]) \times [0, \pi]$. The initial condition is:

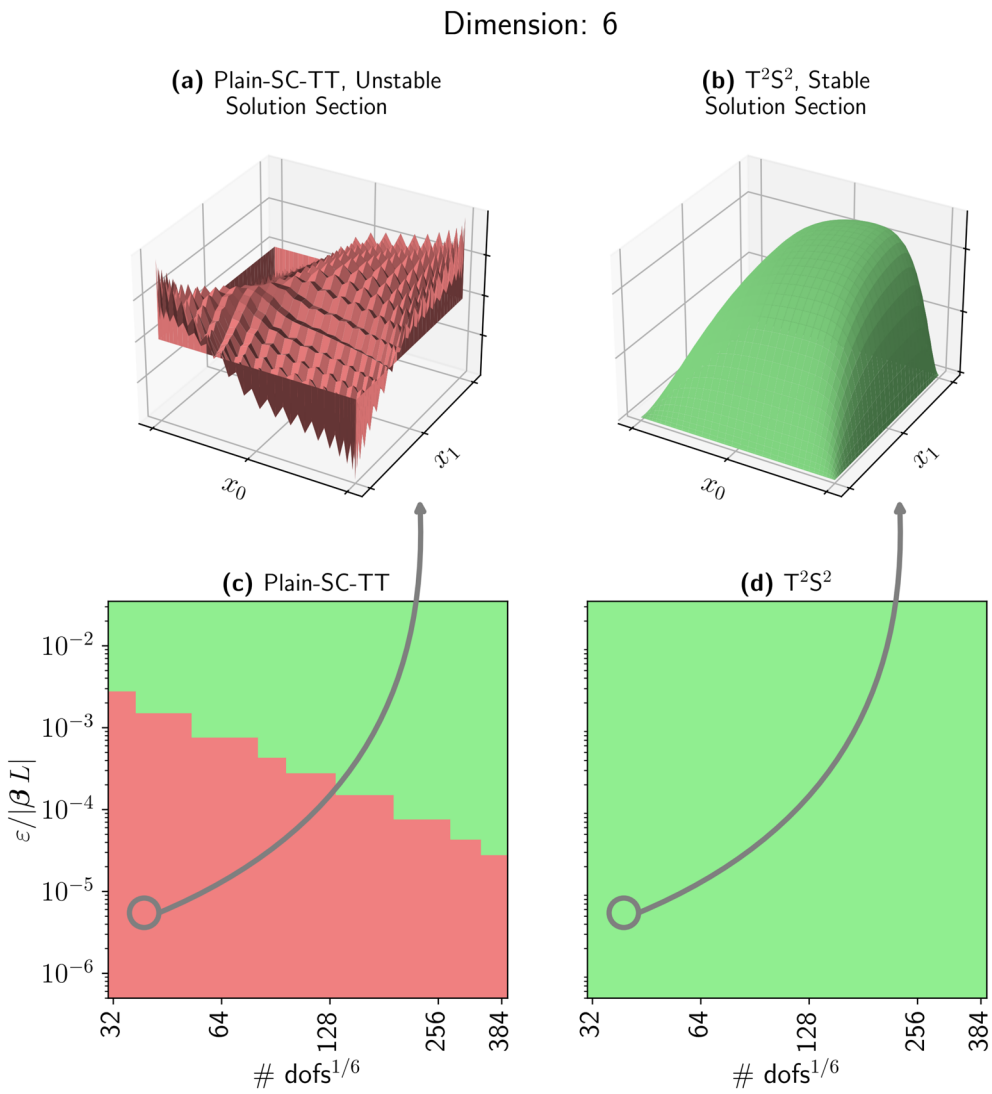


Fig. 6. With $\boldsymbol{\beta}$ fixed as a six-dimensional unit vector, we conduct a detailed analysis of the solution's oscillatory behavior. (**a**) shows a two-dimensional cross-section of the solution to the six-dimensional problem (5) with $\varepsilon = 10^{-5}$ and polynomial degree 39, using collocation points at the zeros of P_n'. The solution presents macroscopic spurious oscillations. The corresponding solution using superconsistent collocation points exhibits the expected regularity and characteristic boundary layers. (**c,d**) Colormaps where each pixel represents a discrete solution of problem (5) for specific combinations of polynomial degree and $\varepsilon/|\boldsymbol{\beta} L|$. (**c**) Highlights regions of oscillatory solutions in red, demonstrating the limitations of Plain-SC-TT. In contrast, (d) illustrates how our method maintains non-oscillatory behavior across all values of $\varepsilon/|\boldsymbol{\beta} L|$ and polynomial degrees.

$$f(x_1, x_2, 0) = \begin{cases} 16\left(1 - 4\left(x_2 - \frac{1}{2}\right)^2\right)^2 \left(1 - 4\,x_1^2\right)^2 & x_2 > 0,\ \frac{1}{2} \le x_1 < \frac{1}{2}, \\ 0 & \text{elsewhere.} \end{cases}$$

The polynomial degree of the basis functions is 16. Figure 8 presents the results of this calculation, in the case of the spacetime formulation. Panel (a) illustrates the evolution of the bump's position within the domain, which describes a circular trajectory through contour lines at various time snapshots, together with the prescribed velocity field. Theoretically, the configuration at time π should be mirror symmetric versus the initial condition. Panel (b) provides quantitative accuracy analysis through the normalized maximum norm $\|f_n(\cdot, t_i)\|_\infty/16$, computed on a 128×128 uniform grid. The results demonstrate that $\mathrm{T}^2\mathrm{S}^2$ preserves the solution structure with minimal oscillation behavior. In this panel we superimpose the result for the spacetime formulation and the time marching schemes: backward Euler and Crank-Nicolson. The implicit backward Euler scheme reads:

$$f^{k+1} - \Delta t\left(\varepsilon \Delta f^{k+1} - \boldsymbol{\beta} \cdot \nabla f^{k+1}\right) = f^k, \tag{6}$$

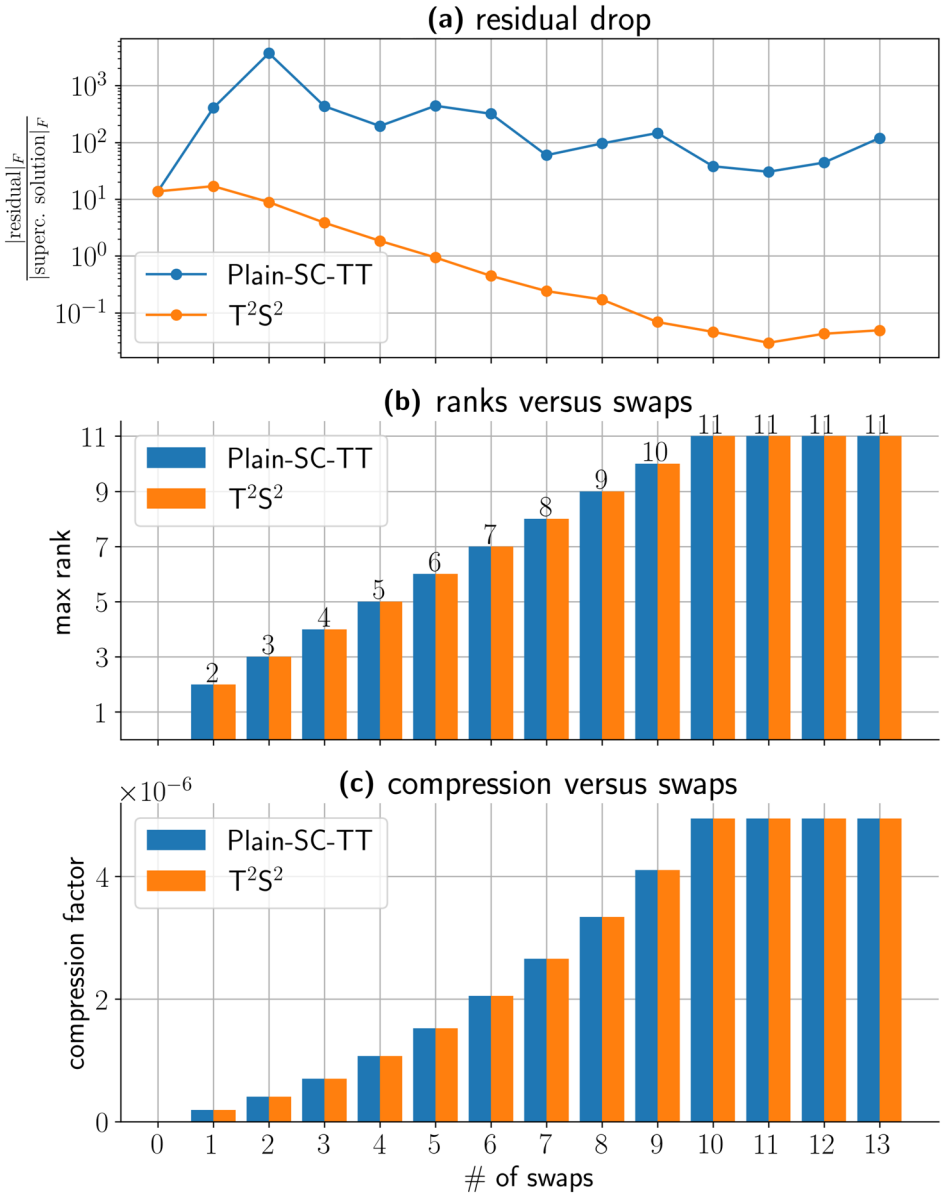


Fig. 7. We present the residual decay, the ranks, and the compression factor against the number of AMEn swaps[15] for our T^2S^2 solver and the Plain-SC-TT collocation methods in the case of 6 dimensional problem 5, degree 39, $\varepsilon = 10^{-5}$ and β six-dimensional vector of ones. This problem is of special interest because it presents a boundary layer. The boundary layer is a high-gradient region of the solution that connects a boundary condition with the internal values. The approximation of such high gradients is generally challenging in the TT format. (**a**) Reports how the residual drops against the number of swaps. The residual does not converge in the Plain-SC-TT case, while in T^2S^2 case, the residual drops with a regular gradient. The final value of the residual $\mathcal{O}(10^{-2})$ is a consequence of the overall bad conditioning of the problem. (**b**) The solver increases the rank of the solution at each swap until it finds a stationary point at swap 11 for a corresponding rank equal to 11. After the 11th swap, the solver does not find any further information. The compression factor is in the range of $\mathcal{O}(10^{-6})$. (**c**) The computational cost on an i9 Intel laptop with 64 Gigabytes of RAM is in the range of seconds.

and the semi-implicit Crank–Nicolson scheme is:

$$f^{k+1} - \frac{\Delta t}{2}\left(\varepsilon \Delta f^{k+1} - \boldsymbol{\beta} \cdot \nabla f^{k+1}\right) = f^k + \frac{\Delta t}{2}(\varepsilon \Delta f^k - \boldsymbol{\beta} \cdot \nabla f^k). \tag{7}$$

We choose a time step $\Delta t = \pi/160$ in both cases. Since the implicit backward Euler scheme is only first-order accurate, the solution approximation shows significative numerical diffusion as the peak's value is reduced by roughly 10%. The Crank-Nicolson scheme is particularly well-suited for this problem due to its reduced

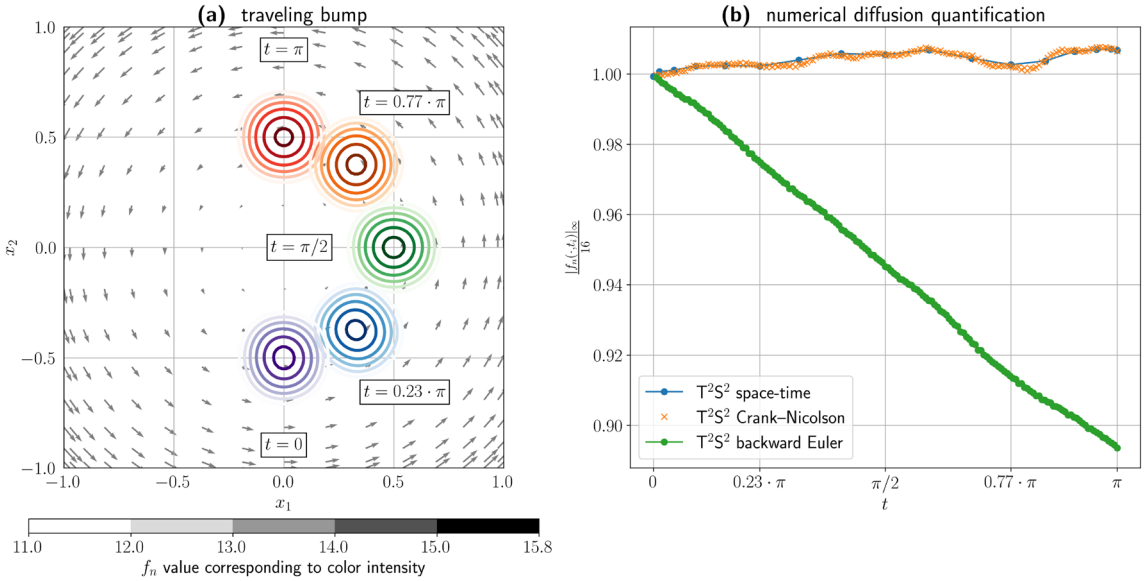


Fig. 8. Results for the traveling "bump" test case. (**a**) Contour lines for the "bump" at different time steps. Different colors distinguish the different instants; the shades distinguish the different function values. The different levels are reported in the gray color bar. (**b**) The time evolution of the solution peak values for three T^2S^2 method's implementations using the implicit backward Euler scheme, the semi-implicit Crank–Nicolson scheme, and the space-time formulation. A video of the traveling bump is provided as Supplementary Materials.

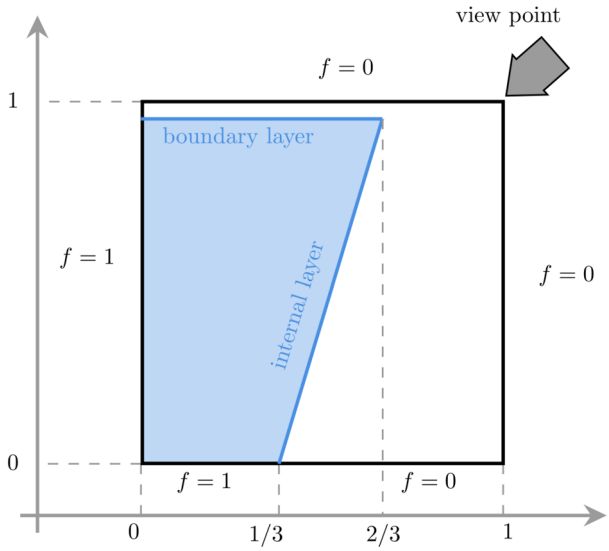


Fig. 9. This picture gives a schematic overview of the convection-diffusion test case as proposed by Hughes and coauthors[32,33], highlighting the boundary and internal layers of the solution.

numerical dissipation compared to other schemes. This property is critical as it better preserves the system's physical quantities, which is a key characteristic in many applicative problems.

The second test case, originally proposed by Tom Hughes and coauthors[32,33], evaluates the method's ability to resolve both boundary and internal layers. We solve problem (5) in two dimensions with $b = 0$ and decreasing values of ε. The discontinuous boundary at $x_2 = 0$, which is transported in the interior by the convection field $\boldsymbol{\beta} = (1, 3)^T$, generates the internal layer on the bottom side of the computational domain. The boundary layer is determined by setting a homogeneous boundary condition on the top side of the computational domain. The test case specifics are sketched in Fig. 9. The boundary layer connects the solution with the boundary conditions and the internal layer connects two flat regions of the solution, i.e., $f = 1$ (on the left), and $f = 0$ (on the right). The reader will notice that the smaller is ε, the higher is the gradient of the internal layer; see Figs. 10a–c. T^2S^2 method confirms its oscillatory-free behavior. In contrast, Plain-SC method exhibits spurious numerical

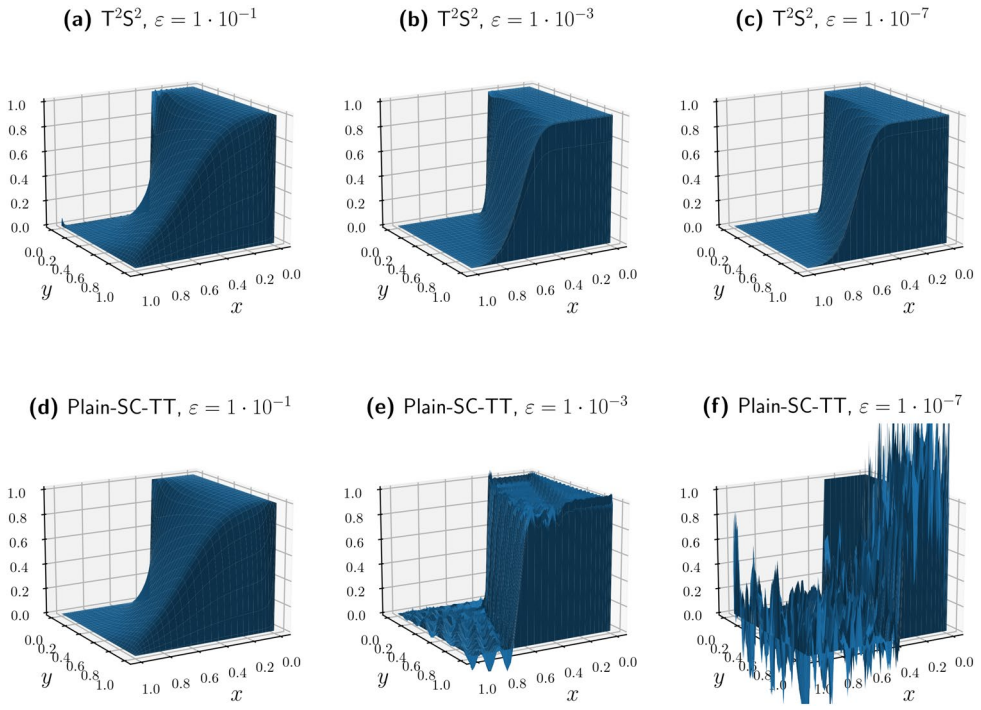


Fig. 10. This picture represents the solutions of the test proposed by Hughes and coauthors[32,33] for various values of the parameter ε and for the Plain-SC-TT collocation method versus $\mathrm{T}^2\mathrm{S}^2$. The solution degree is 63. While Plain-SC-TT shows spurious oscillations at $\varepsilon = 10^{-3}$, $\mathrm{T}^2\mathrm{S}^2$ is stable regardless of the value of ε.

oscillations, Fig. 10d–f, which start being visible for a relatively big value of $\varepsilon = 10^{-3}$, and fails to resolve the problem accurately.

Discussion

$\mathrm{T}^2\mathrm{S}^2$ effectively integrates the strengths of its core components: Spectral Collocation for accuracy, Superconsistency for stability in the presence of boundary or internal layers, and TT format as a low-rank linear algebra framework. In Section "Results", we demonstrated that $\mathrm{T}^2\mathrm{S}^2$ reduces the approximation error to the range of $\mathscr{C}^{12}$, with the error decreasing sharply when plotted on a logarithmic scale. Despite using a low-rank solution, the grid density is such that traditional solvers—even at theoretically optimal efficiency would require millions of years on the current world's most powerful architectures. $\mathrm{T}^2\mathrm{S}^2$ preserves spectral accuracy in high-dimensional, high dense grid scenarios. In the section about stability, we tested the solver's stability using a six-dimensional stationary test case characterized by a boundary layer solution, i.e., a solution characterized by a high gradient region connecting the domain interior to the homogeneous boundary conditions. This test not only confirms the unconditional stability of the method versus the parameters choice, but also shows that $\mathrm{T}^2\mathrm{S}^2$ can approximate high dimensional/high gradient solutions with a rank in the range of 10. $\mathrm{T}^2\mathrm{S}^2$ achieves this without requiring any intermediate approximation layers, either at the continuous or discrete level: the discrete version of problem (1) fits naturally into the TT structure. Moreover, $\mathrm{T}^2\mathrm{S}^2$ differs fundamentally from low-rank data-driven approaches, such as machine learning and reduced-order modeling[41,42] as it does not require any high-fidelity offline computations or training phases. As discussed in "Results" , we remarked that a high-fidelity computation in seven dimensions results in prohibitive computational costs, even for a relatively coarse grid. Following the characterization of low-rank methods proposed in the comprehensive review by Einkemmer et al.[12], a time discretization of $\mathrm{T}^2\mathrm{S}^2$, such as the backward Euler and Crank-Nicolson schemes, falls into the category of the Step And Truncate (SAT) methods. SAT methods offer the flexibility to incorporate $\mathrm{T}^2\mathrm{S}^2$. Alternatively, sparse grid methods represent a worthmentioning attempt to tackle the course of dimensionality. They are capable of reducing the total number of degrees of freedom to $\mathscr{O}(\#\mathrm{dofs}_{1\mathrm{d}} \cdot \log(\#\mathrm{dofs}_{1\mathrm{d}})^{d-1})$, possibly lower compression ratios can be achieved through adaptivity[43]. While this is a remarkable result, considering a grid with a number of nodes in the order of $100^7 = 10^{14}$ and solution rank around ten, sparse grids still need 10^8 degrees of freedom, while $\mathrm{T}^2\mathrm{S}^2$ can compress the data from 10^{14} to $7 \cdot 10^4$, without compromising for accuracy. Low-rank approximation covers a wide variety of practical problems. However even if this prerogative is not satisfied, our method can still be effectively integrated to approximate initial solution guesses or as a preconditioner. Regular grids are another commonly highlighted limitation of TT technology. Even though this limitation holds true, $\mathrm{T}^2\mathrm{S}^2$ strongly cuts computational costs enabling much denser grids than standard discretisations. This $\mathrm{T}^2\mathrm{S}^2$ capability also nicely couples with immersed methods like immersed boundary method[44], fictitious domain[45], or distributed Lagrangian multiplier[46]. These techniques are valuable tools to enforce non-standard boundary conditions or geometries.

Problem (1) serves as the foundation for a wide range of solvers in fields such as radiation transport physics, kinetic theory, and computational fluid dynamics. As a result, T^2S^2 is a tool for a wide range of applications in modeling high-dimensional transport phenomena. Technological advances are driving the development of next-generation nuclear technologies, including small modular reactors, advanced modular reactors, microreactors, and nuclear fusion systems. These innovations demand large-scale neutron transport and fluid dynamics calculations for reactor design and throughout the nuclear fuel cycle, including nuclear safeguards, nuclear non-proliferation, and security needs[47–51]. For instance, the Boltzmann neutron transport equation, which neglects neutron-neutron collisions, is a linear equation. Similarly, the Vlasov equation, used in plasma physics, is also a linear equation where the non-linearity comes from the coupling with Maxwell's equations. The same, linear transport equations govern photon transport. We anticipate T^2S^2 to be a key numerical solver in addressing these complex computational challenges. The applications are not limited to linear problems only. In the kinetic theory of gases, in computational fluid dynamics and computational chemistry[52,53], the transport equations often involve nonlinear terms, whose solution strategy can be approximated by solving a sequence of linear problems, an approach that can be addressed using the T^2S^2 method. In summary, T^2S^2 has the potential to impact a wide range of fields requiring efficient, high-dimensional transport computations. Its ability to solve such problems on standard computing hardware makes it particularly valuable for practical applications with limited computational resources. Future directions include extending the T^2S^2 architecture to run efficiently on supercomputers, expanding its applicability to large-scale scientific and engineering problems.

Methods

In this section, we provide a formal mathematical description of the T^2S^2 scheme. We introduce the one-dimensional operators in the subsection "Spectral Collocation Methods". Next, in section "TT representation" we introduce the TT linear algebra format. Finally, in the section "T^2S^2algorithm for convection-diffusion-reaction equation" we present how the building blocks are integrated to construct the high-dimensional solver.

Spectral collocation methods

Spectral approximations. Let $\mathscr{X}$ and $\mathscr{Y}$ be suitable functional spaces. We consider a linear operator $L : \mathscr{X} \to \mathscr{Y}$ and, for $b \in \mathscr{Y}$, we are interested in solving the problem

$$\text{Find } f \in \mathscr{X} \text{ such that:} \quad Lf = b. \tag{8}$$

The operator L can be rather generic, but for this paper, we only examine applications in the framework of transport-diffusion PDEs. In this case, formulation (8) includes boundary and initial conditions to be assigned to f. We assume that problem (8) is mathematically well-posed in the sense of Hadamard[54], and we numerically discretize it through the spectral collocation method based on algebraic polynomials. Spectral methods are a very effective tool in numerical approximation of various mathematical models due to their excellent convergence rate in the presence of very smooth solutions[19,21].

We express f as a linear combination of $n+1$ basis functions denoted by $l_i(x)$, $0 \le i \le n$. These functions form a polynomial Lagrangian basis for a predefined set of nodes $\vec{\mathbf{x}} = \{x_j\}_{j=0,1,\dots,n}$ in the computational domain, so that $l_i(x_j) = \delta_{ij}$. The points x_i are the *collocation points* and form the *collocation grid*. In this way, we will assume that $f(x) \approx f_n(x) = \sum_{0\le i\le n} f_i\ l_i(x)$. As a consequence $f_n \in \mathscr{X}_n$, where $\mathscr{X}_n$ is a finite dimensional space.

In the examples to follow, the distribution of such nodes is not uniform (i.e., with fixed mesh size). Indeed, we set $x_0 = -1$ and $x_n = 1$, and the remaining nodes are the zeros of P_n', which is the first derivative of the n -th Legendre polynomial P_n.

In the pseudospectral method, Eq. (8) is required to be satisfied at the collocation nodes $\vec{\mathbf{x}}$. To fix the idea, let us take $L := d^2/dx^2$ and let l_i'' be the second derivative of each element of the basis function l_i, for $i = 0, 1, \dots, n$, so that $f_n''(x) = \sum_{0\le i\le n} f_i\ l_i''(x)$ (note that there is no approximation here since the second derivative of a polynomial can be computed exactly). We then build a corresponding linear system, that, by choosing for instance $n = 4$, takes the form:

$$\big(L_n f_n\big)(\vec{\mathbf{x}}) := \underbrace{\begin{bmatrix} l_0''(x_0) & l_0''(x_1) & l_0''(x_2) & l_0''(x_3) & l_0''(x_4) \\ l_1''(x_0) & l_1''(x_1) & l_1''(x_2) & l_1''(x_3) & l_1''(x_4) \\ l_2''(x_0) & l_2''(x_1) & l_2''(x_2) & l_2''(x_3) & l_2''(x_4) \\ l_3''(x_0) & l_3''(x_1) & l_3''(x_2) & l_3''(x_3) & l_3''(x_4) \\ l_4''(x_0) & l_4''(x_1) & l_4''(x_2) & l_4''(x_3) & l_4''(x_4) \end{bmatrix}}_{D^2} \begin{bmatrix} f_n(x_0) \\ f_n(x_1) \\ f_n(x_2) \\ f_n(x_3) \\ f_n(x_4) \end{bmatrix} = \begin{bmatrix} b(x_0) \\ b(x_1) \\ b(x_2) \\ b(x_3) \\ b(x_4) \end{bmatrix} =: b(\vec{\mathbf{x}}). \tag{9}$$

In Eq. (9), the left-hand side matrix, called D^2, provides the spectral collocation discretization of the second derivative operator d^2/dx^2. The entries of such a matrix are constructed by evaluating the second derivatives of each element of the Lagrange basis at the collocation points. For a general n, we have that $D^2 = L_n = \{l_i''(x_j)\}$, $i, j = 0, 1, \dots, n$, is a $(n+1) \times (n+1)$ matrix. To impose Dirichlet boundary conditions, the first and the last row of the above matrix will be forced to be the corresponding first and last row of the $(n+1) \times (n+1)$ identity matrix. Furthermore, we impose the proper boundary conditions by suitably replacing $b(x_0)$ and $b(x_4) = b(x_n)$. For example, we set $b(x_0) = b(x_n) = 0$ in the homogeneous case.

The theory of stability and convergence of the spectral collocation method dates back to the 70s[19,55]. We briefly recall that stability requires the boundedness of the approximate solution in some suitable norm. The

bound must be uniform versus the discretization parameter n. As the consistency property is satisfied, i.e., exact and discretized operators coincide in a suitable finite dimensional space, convergence of f_n to f is obtained in certain normed spaces[19]. The convergence, for spectral type approximations, displays an exponential rate under the hypotheses of high regularity of the solution[56]. This makes this numerical technique very competitive with other known methods, as it requires far fewer degrees of freedom if the regularity assumptions on the exact solution are fulfilled.

Superconsistency to define a superconsistent method, we introduce a second grid of $n-1$ points, e.g., $\vec{\mathbf{z}} = \{z_j\}_{j=1,\dots,n-1}$, within the computational domain, according to the following recipe. Hereafter, we will refer to the points x_i introduced in the previous section as the *representation points*, and the points z_i as the *collocation points*. Accordingly, we say that $\vec{\mathbf{x}}$ is the *representation grid*, and $\vec{\mathbf{z}}$ is the *collocation grid*. We use the representation points to express $f_n = \{f_n(x_i)\}$. Instead, we use the collocation points to derive the numerical method. The approximation still keeps the form introduced above, but now we require that $L_n(f_n(\vec{\mathbf{x}})) = \big(L_n f_n\big)(\vec{\mathbf{z}}) = b_n(\vec{\mathbf{z}})$. Now L_n transforms a vector built at the nodes $\vec{\mathbf{x}}$ into a vector built at the nodes $\vec{\mathbf{z}}$. Thus, at the interior of the domain, the new set of points replaces the old one, which is only used to express the approximated solution. In other words, we look for the values $f_n(\vec{\mathbf{x}})$, but we require the problem to be satisfied at the points $\vec{\mathbf{z}}$. The two sets of points may coincide, but this depends on the operator L, to realize a specific property. The method is said to be *superconsistent* if the following consistency condition holds for a functional space $\widetilde{\mathscr{X}}_n$ larger than $\mathscr{X}_n$:

$$\big(Lq\big)(\vec{\mathbf{z}}) = L_n\big(q(\vec{\mathbf{x}})\big) \qquad \forall q \in \widetilde{\mathscr{X}}_n. \tag{10}$$

This means that the exact operator applied to q and successively evaluated at $\vec{\mathbf{z}}$, equals the discrete operator applied to the same q evaluated at the nodes $\vec{\mathbf{x}}$. We want this to be true for functions that are outside the approximation space $\mathscr{X}_n$, realizing in this way a higher degree of consistency. This construction can be done, in principle, for a large range of problems, as, for example, the approximation of integral equations[57].

Let $\widetilde{\mathscr{X}}_n = \mathscr{X}_n \cup \mathrm{span}(\chi)$ and consider the one-dimensional convection-diffusion operator

$$L := \beta \cdot d/dx - \varepsilon \cdot d^2/dx^2.$$

We consider the function $\chi := \chi_{n+1} = (1-x^2)\,P_n'$, where P_n' is the first derivative of the Legendre polynomial P_n of degree n[58]. Now, imposing the superconsistent condition amounts to determining the points $\vec{\mathbf{z}}$ in such a way that:

$$n(n+1)\left[\varepsilon P_n' - \beta P_n\right](z_j) = 0 \qquad \text{for } j = 1, \dots, n-1.$$

These points are located between the zeros of P_n and P_n' and their location depends on the ratio between ε and β, as shown in Fig. 1. We denote S_n the $(n+1)\times(n+1)$ interpolation matrix that maps values at points $\vec{\mathbf{x}}$ to values at points $\vec{\mathbf{z}}$. If L_n is the discrete operator evaluated at the representation points, $S_n \cdot L_n$ is the operator evaluated at the collocation points.

TT representation

We first review some basic notation and definitions about tensors[59] and, in particular, the TT format[13,60]. Then, we review some important basic facts about the TT decomposition.

Notation and generalities.

A d-dimensional tensor $\mathscr{A} = \big\{\mathscr{A}(i_1, i_2, \dots, i_d)\big\}$, $i_\ell = 1, 2, \dots, n_\ell$, $\ell = 1, 2, \dots, d$ is a multi-dimensional array whose entries are indexed by d indices. We denote d-dimensional tensors using calligraphic, upper case fonts, such as $\mathscr{A}, \mathscr{B}, \mathscr{C}$. For scalars, vectors, and matrices, which are formally d-way tensors with, respectively, a number of dimensions $d = 0, 1, 2$, we adopt a different notation for clarity. Precisely, we denote scalar quantities using normal, lower-case fonts as in a, b, c; vector fields using bold, lower-case fonts as in $\mathbf{a}, \mathbf{b}, \mathbf{c}$; matrices using normal, upper case fonts as in A, B, C. Furthermore, we denote the entries of a vector $\mathbf{a}$ as $a(i)$ and the entries of a matrix A as $A(i,j)$. We may occasionally denote the entries of three-dimensional tensor as $\mathscr{A}(i,j,k)$ instead of $\mathscr{A}(i_1,i_2,i_3)$, but we prefer using the notation $\mathscr{A}(i_1,i_2,\dots,i_d)$ when $d > 3$. For completeness, we anticipate that when we consider a three-dimensional tensor that is also a core of a TT decomposition (see below for the definition), the first and last indices are a Greek letter, e.g., α, β etc, and the intermediate index is always a Roman letter, e.g., *i*, *j* etc.

Following a MATLAB-style notation, we denote free indices using the symbol " : ". This notation makes it possible to represent the two important structural concepts of *fiber* and *slice* of a tensor. A fiber is a vector obtained by fixing all but one index, and a slice is a matrix obtained by fixing all but two indices. For example, a 3D tensor $\mathscr{A}(i,j,k)$ has three different types of fibers, i.e., $\mathscr{A}(:,j,k)$, $\mathscr{A}(i,:,k)$, $\mathscr{A}(i,j,:)$, and three different types of slices $\mathscr{A}(i,:,:)$, $\mathscr{A}(:,j,:)$, $\mathscr{A}(:,:,k)$. Finally, always according with the MATLAB-style notation, a repetition of the free index symbol " : " denotes an index contraction. For example, let A, B, C be three matrices with compatible sizes such that $C = AB$. Then, the notation "$C(i,j) = A(i,:)B(:,j)$" is equivalent to $C(i,j) = \sum_\alpha A(i,\alpha)B(\alpha,j)$, where the summation over the index α is taken, as usual, over all the elements of the *i*-th rows of A and *j*-th columns of B.

TT format.

The TT format provides an efficient representation of high-dimensional tensors by decomposing them into a product of much smaller, three-dimensional tensors called the "TT cores". Specifically, a d-dimensional tensor

$\mathscr{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is said to be in TT format if there exist d cores $\mathscr{A}_\ell \in \mathbb{R}^{r_{\ell-1} \times n_\ell \times r_\ell}$, for $\ell = 1, \ldots, d$, with $r_0 = r_d = 1$, such that each entry of $\mathscr{A}$ can be computed as:

$$\mathscr{A}(i_1, \ldots, i_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \mathscr{A}_1(1, i_1, \alpha_1) \mathscr{A}_2(\alpha_1, i_2, \alpha_2) \cdots \mathscr{A}_d(\alpha_{d-1}, i_d, 1). \tag{11}$$

In some sense, Eq. (11) is a multi-dimensional generalization of the (reduced) Singular Value Decomposition (SVD) of matrices. Note, however, that we do not require any specific orthogonality property in the above definition (more details on the SVD connection will be given in the next example and later in this section).

To fix ideas, consider the following two dimensional case. A matrix $A \in \mathbb{R}^{n_1 \times n_2}$ is decomposed in the product of two matrices $A_1 \in \mathbb{R}^{n_1 \times r_1}$ and $A_2 \in \mathbb{R}^{r_1 \times n_2}$, so that $A(i_1, i_2) = \sum_{\alpha=1}^{r_1} A_1(i_1, \alpha) A_2(\alpha, i_2)$. We can easily construct such a decomposition through the SVD of A, which we write as usual as $A = U \Sigma V^T$, where $U \in \mathbb{R}^{n_1 \times r_1}$ and $V \in \mathbb{R}^{n_2 \times r_1}$ are two orthogonal-by-column matrices, and $\Sigma \in \mathbb{R}^{r_1 \times r_1}$ is the diagonal matrix with r_1 strictly positive singular values, so that r_1 is the rank of matrix A. Then, take, for instance, $A_1 = U$, $A_2 = \Sigma V^T$, or $A_1 = U\Sigma$, $A_2 = V^T$. We can also use the tensor notation introduced above and specialized to the case $d = 2$, so that we can write $\mathscr{A}(i_1, i_2) = \sum_{\alpha=1}^{r_1} \mathscr{A}_1(1, i_1, \alpha) \mathscr{A}_2(\alpha, i_2, 1)$, where $\mathscr{A}_1 \in \mathbb{R}^{r_0 \times n_1 \times r_1}$, $\mathscr{A}_2 \in \mathbb{R}^{r_1 \times n_2 \times r_2}$, with $r_0 = r_2 = 1$. The decomposition process can be easily extended to $d = 3$ by redefining $\mathscr{A}_2$ as the intermediate core $\mathscr{A}_2 \in \mathbb{R}^{r_1 \times n_2 \times r_2}$ and introducing a final core $\mathscr{A}_3 \in \mathbb{R}^{r_2 \times n_3 \times r_3}$ with $r_3 = 1$, so that $\mathscr{A}(i_1, i_2, i_3) = \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \mathscr{A}_1(1, i_1, \alpha_1) \mathscr{A}_2(\alpha_1, i_2, \alpha_2) \mathscr{A}_2(\alpha_2, i_3, 1)$. This process can be continued by adding one new dimension at each step as it is illustrated in Fig. 11 for the construction of the TT representation of a tensor up to $d = 6$.

Remark. The computational effectiveness of the TT representation of multi-dimensional arrays is related to controlling the core ranks. To exemplify this fact, consider a six-dimensional array with mode sizes $n = n_1 = \ldots = n_6 = 10^3$ and assume that the ranks are all equal to $r = r_1 = \ldots = r_{d-1} = 10$. In such a case, the total storage required to represent such a tensor in TT format is proportional to $\mathscr{O}(nr^2 d) = 6\,10^5$ against a full grid storage requirement of $\mathscr{O}(n^d) = 10^{18}$.

The TT representation can equivalently be reformulated in a more compact form by representing each core as a set of slice matrices parametrized with the spatial index i_ℓ, e.g., $A_\ell(i_\ell) = \mathscr{A}_\ell(:, i_\ell, :) \in \mathbb{R}^{r_{\ell-1} \times r_\ell}$, yielding:

$$\mathscr{A}(i_1, \ldots, i_d) = A_1(i_1) A_2(i_2) \cdots A_d(i_d). \tag{12}$$

The internal sizes of the decomposition, i.e., the set of $(d-1)$ integers $\{r_\ell\}_{\ell=1}^{d-1}$, are the the *TT-ranks*, or, simply, the *ranks*, of the TT decomposition. A convenient upper bound on them is their maximum value, which we denote as $r = \max_\ell \{r_\ell\}$. We also introduce an upper bound on the dimensions, denoted as $n = \max_\ell n_\ell$. The TT-ranks determine the storage complexity of the representation, which scales linearly with the dimension d as $\mathscr{O}(dnr^2)$, in contrast to the exponential growth $\prod_{\ell=1}^{d} n_\ell = \mathscr{O}(n^d)$ required for the full tensor representation. The linear dependence on d of the TT-representation of a tensor is very convenient, reflecting the compression efficiency of such a representation, whenever $r \ll n$. In such a case, we say that (11) and (12) are a *low-rank representation* of tensor $\mathscr{A}$.

If we introduce the multi-index $i = (i_1, i_2, \ldots, i_d)$, we can think that Eq. (11) defines the TT format representation for a sort of "*multi-dimensional vector*", i.e., $\mathscr{A}(i)$. It is also possible to adopt a similar *TT format for linear operators*, i.e., "*multi-dimensional matrices*" $\mathscr{L}(i, j)$ that depend on two d-dimensional multi-indices $i = (i_1, i_2, \ldots, i_d)$ and $j = (j_1, j_2, \ldots, j_d)$. The formal definition reads as

$$\begin{aligned} \mathscr{L}\Big((i_1, i_2, \ldots, i_d), (j_1, j_2, \ldots, j_d)\Big) &= \mathscr{L}\Big((i_1, j_1), (i_2, j_2), \ldots, (i_d, j_d)\Big) \\ &= \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} L_1\big(1, (i_1, j_1), \alpha_1\big)\, L_2\big(\alpha_2, (i_2, j_2), \alpha_3\big)\, \ldots L_d\big(\alpha_{d-1}, (i_d, j_d), 1\big), \end{aligned} \tag{13}$$

where the first equality follows from a convenient index permutation to form the index pairs (i_ℓ, j_ℓ), $\ell = 1, 2, \ldots, d$.

The so-called "*core notation*" gives a more intuitive insight into the structure of the cores:

$$A_\ell = \begin{bmatrix} A_\ell(1, :, 1) & A_\ell(1, :, 2) & \ldots & A_\ell(1, :, r_\ell) \\ A_\ell(2, :, 1) & A_\ell(2, :, 2) & \ldots & A_\ell(2, :, r_\ell) \\ \vdots & \vdots & \vdots & \vdots \\ A_\ell(r_{\ell-1}, :, 1) & A_\ell(r_{\ell-1}, :, 2) & \ldots & A_\ell(r_{\ell-1}, :, r_\ell) \end{bmatrix},$$

$$L_\ell = \begin{bmatrix} L_\ell(1, :, :, 1) & L_\ell(1, :, :, 2) & \ldots & L_\ell(1, :, :, r_\ell) \\ L_\ell(2, :, :, 1) & L_\ell(2, :, :, 2) & \ldots & L_\ell(2, :, :, r_\ell) \\ \vdots & \vdots & \vdots & \vdots \\ L_\ell(r_{\ell-1}, :, :, 1) & L_\ell(r_{\ell-1}, :, :, 2) & \ldots & L_\ell(r_{\ell-1}, :, :, r_\ell) \end{bmatrix}.$$

With such notation and using the definition of the strong Kroncker product[61] "$\bowtie$", we write the TT decomposition in the more compact form:

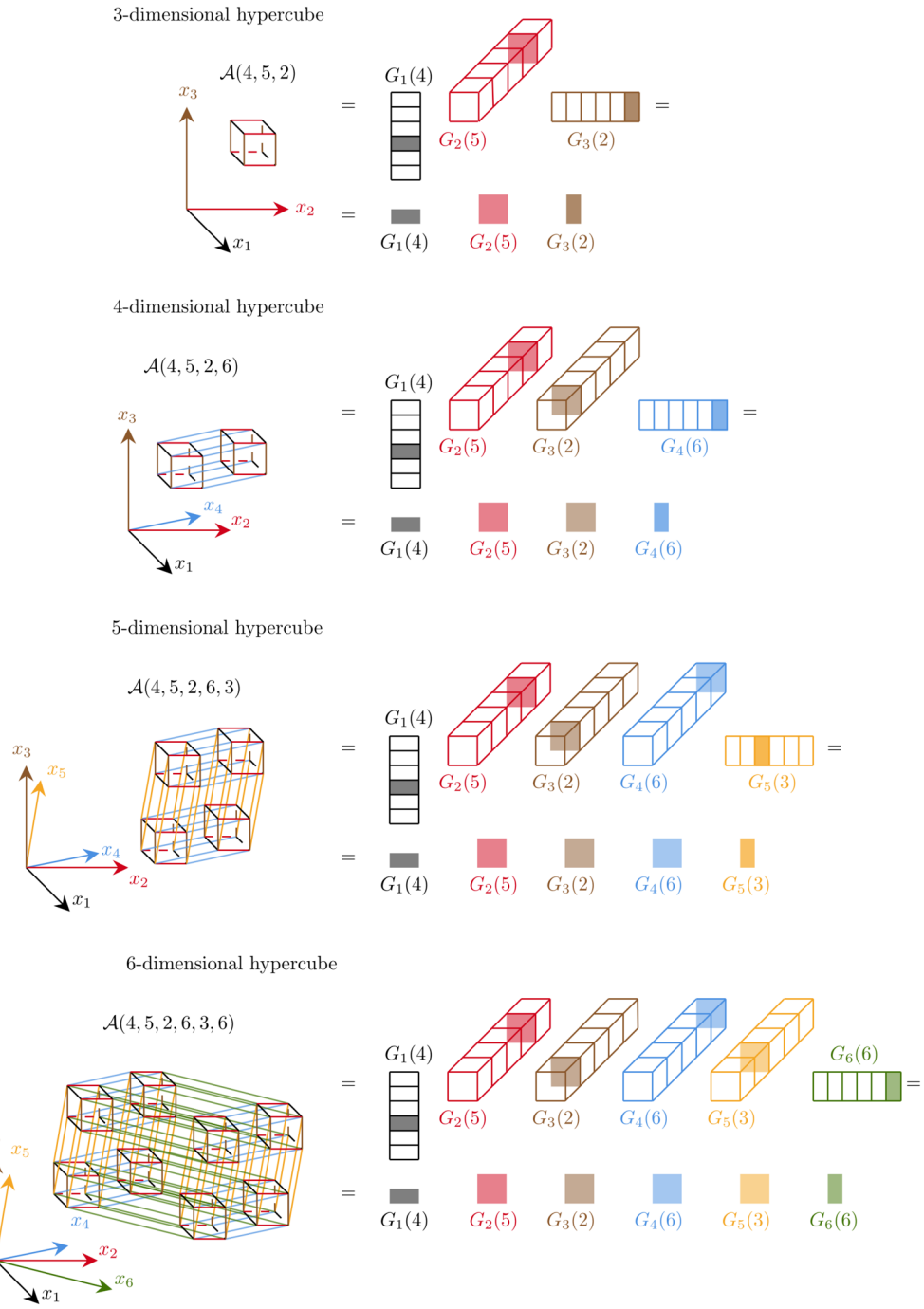


Fig. 11. In this picture we present the progression from a three to a six-dimensional array in full and TT format.

$$\mathscr{A} = A_1 \bowtie A_2 \bowtie \ldots \bowtie A_d, \qquad \mathscr{L} = L_1 \bowtie L_2 \bowtie \ldots \bowtie L_d.$$

We will make use of this mathematical formalism to describe the different parts of $\mathrm{T}^2\mathrm{S}^2$.

TT decomposition. The construction of an optimal TT representation for a d-dimensional tensor can be achieved through the TT-SVD algorithm[13]. This algorithm builds the TT decomposition through a systematic application of singular value decompositions on specific matrix representations of the tensor, known as *unfoldings*. For a d-dimensional tensor $\mathscr{A}$, we consider $d-1$ "fundamental" matrix unfoldings. The ℓ-th unfolding, denoted as A_ℓ, reorganizes the tensor entries into a matrix of size $(\prod_{k=1}^{\ell} n_k) \times (\prod_{k=\ell+1}^{d} n_k)$ through the relation:

$$A_\ell(\eta_\ell(i_1,\ldots,i_\ell), \eta'_\ell(i_{\ell+1},\ldots,i_d)) = \mathscr{A}(i_1,\ldots,i_d), \tag{14}$$

where η_ℓ and η'_ℓ are two bijective mappings between the respective index tuples, e.g., $(i_1, \ldots, i_\ell)$ and $(i_{\ell+1}, \ldots, i_d)$, often called "long indices", and the row and column indices of matrix A_ℓ. A very strong connection exists between the TT-ranks of tensor $\mathscr{A}$ and the ranks of the unfolding matrices. In fact, the rank of the ℓ-th matrix unfolding is a lower bound for the TT-rank r_ℓ, and a constructive proof shows that TT-SVD can compute an *exact* TT-representation with TT-ranks r_ℓ equal to the ranks of the matrix unfoldings A_ℓ[13], Theorem 2.1, i.e., $r_\ell = \operatorname{rank}(A_\ell)$ for $\ell = 1, \ldots, d-1$.

Of particular practical importance is the existence of low-rank approximations. The sequential nature of the TT-SVD algorithm ensures that the TT ranks are optimal in the sense that they provide the best possible approximation for a given accuracy threshold. If we truncate the singular values in the TT-SVD algorithm below a given threshold τ, the resulting approximation error in the Frobenius norm is bounded by $\tau\sqrt{d-1}\|\mathscr{A}\|_F$. Such an approximate TT decomposition keeps the TT ranks minimal while achieving the accuracy prescribed by τ. Therefore, when computational constraints or data compression requirements need smaller ranks $r'_\ell \leq \operatorname{rank}(A_\ell)$, a best approximation $\mathscr{A}^{TT}_{\text{best}}$ in the TT format is guaranteed to exist.

The TT-SVD algorithm efficiently provides a quasi-optimal approximation $\mathscr{A}^{TT}$ in the sense that:

$$\|\mathscr{A} - \mathscr{A}^{TT}\|_{\mathscr{F}} \leq \sqrt{d-1} \inf_{\mathscr{B}^{TT} \in \mathbb{TT}(\mathbf{r}'_\ell)} \|\mathscr{A} - \mathscr{B}^{TT}\|_{\mathscr{F}}. \tag{15}$$

Here, $\mathbb{TT}(\mathbf{r}'_\ell)$ denotes the manifold of TT representations with prescribed rank $\mathbf{r}'_\ell = (r'_1, \ldots, r'_{d-1})$, and the multiplicative factor $\sqrt{d-1}$ represents the stability constant of the algorithm[60]. This quasi-optimality result establishes TT-SVD as a practical tool for tensor approximation, offering a controlled trade-off between computational efficiency and approximation accuracy.

The TT format supports the efficient implementation of fundamental multi-linear algebraic operations such as scalar multiplication, tensor addition, element-wise (Hadamard) products, and contractions with matrices and vectors[13]. These operations are designed to preserve the TT structure but they generally lead to an increase in the TT ranks of the resulting tensor. A critical step is the so-called rounding procedure, denoted as $\operatorname{rndg}(\cdot)$, which "recompresses" the tensor by computing a new TT representation with smaller ranks while controlling the approximation error[13]. Given a tensor $\mathscr{A}$ in TT-format, the "rounded" tensor $\operatorname{rndg}(\mathscr{A})$ satisfies the inequality $\|\mathscr{A} - \operatorname{rndg}(\mathscr{A})\|_{\mathscr{F}} \leq \tau\|\mathscr{A}\|_{\mathscr{F}}$, where τ is a threshold parameter that makes a trade-off between accuracy and computational costs possible. On the one hand, smaller values of τ can preserve a better accuracy but implies higher TT ranks, hence increasing the computational costs and storage requirements. On the other hand, larger values of τ lead to a better compression with lower ranks, thus reducing computational complexity at the expense of accuracy. This balance between accuracy and efficiency must be carefully calibrated according to the specific requirements of the application at hand.

Finally, it is worth mentioning the cross-approximation technique[60], which allows us to compute the TT representation of a tensor by sampling only a small subset of its entries. This algorithm makes the calculation of a TT decomposition very efficient since in many applications, mainly when dealing with function-generated tensors, computing and storing the full tensor is prohibitively expensive or impossible. Cross-interpolation typically employs the MaxVol algorithm[62], a sophisticated sampling strategy that selectively chooses optimal rows and columns from the tensor to construct its TT representation. In our context, cross-interpolation proves particularly valuable for efficiently representing the right-hand side term f, allowing us to work with high-dimensional data while maintaining computational tractability.

$\mathrm{T^2S^2}$ algorithm for convection-diffusion-reaction equations

We consider the problem defined by (1), (2), and (3) defined on the d-dimensional hypercube $\Omega \subset \mathbb{R}^d$, with boundary conditions $\partial\Omega$. Therein, ε is the (scalar) diffusion coefficient, $\boldsymbol{\beta} = (\beta_\ell)_{\ell=1,2,\ldots,d}$ is the d-sized vector-valued convection field, ρ is the scalar reaction coefficient, b is the right-hand side source term. The coefficients ε, $\boldsymbol{\beta}$, and ρ can be constant or varying in space as a function of the multidimensional position vector $\mathbf{x} \in \mathbb{R}^d$. In the last case, we assume that these coefficients satisfy appropriate regularity conditions so that problem (1)–(3) is well-posed. For example, we can assume that (*i*) $\varepsilon(\mathbf{x})$ is *uniformly positive*, i.e., there exists a strictly positive constant ϵ_0 such that $\varepsilon(\mathbf{x}) \geq \epsilon_0$ for all $\mathbf{x} \in \Omega$; (*ii*) the components of $\boldsymbol{\beta}$ are smooth (e.g., at least $\beta_\ell \in C^1(\Omega)$) and bounded; (*iii*) ρ is bounded and non-negative, i.e., $\rho \geq 0$. In such a case, we also assume that a sufficiently accurate, low-rank representation in TT format exists for all non-constant coefficients. If these coefficients are low-rank, we can obtain their TT representation through an application of the cross-interpolation algorithm.

The discrete diffusion operator. The low-rank, TT representation of ε reads as $\varepsilon^{TT} = \varepsilon_1 \bowtie \varepsilon_2 \bowtie \ldots \bowtie \varepsilon_d$, where the cores are given in the core notation by

$$\varepsilon_\ell = \begin{bmatrix} \varepsilon_\ell(1,:,1) & \varepsilon_\ell(1,:,2) & \cdots & \varepsilon_\ell(1,:,r_\ell) \\ \varepsilon_\ell(2,:,1) & \varepsilon_\ell(2,:,2) & \cdots & \varepsilon_\ell(2,:,r_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon_\ell(r_{\ell-1},:,1) & \varepsilon_\ell(r_{\ell-1},:,2) & \cdots & \varepsilon_\ell(r_{\ell-1},:,r_\ell) \end{bmatrix}, \ \ell = 1, 2, \ldots, d.$$

Then, for $\alpha_{\ell-1} = 1, 2, \ldots, r_{\ell-1}$ and $\alpha_\ell = 1, 2, \ldots, r_\ell$, we set $E_\ell(\alpha_{\ell-1},:,:,\alpha_\ell) = \operatorname{diag}\big(\varepsilon_\ell(\alpha_{\ell-1},:,\alpha_\ell)\big)$, so that every $E_\ell(\alpha_{\ell-1},:,:,\alpha_\ell)$ is a two-dimensional, diagonal matrix with $\varepsilon_\ell(\alpha_{\ell-1},:,\alpha_\ell)$ along the diagonal. Using such a notation, we rewrite ε^{TT} as $\mathscr{E}^{TT} = \mathscr{E}_1 \bowtie \mathscr{E}_2 \bowtie \ldots \bowtie \mathscr{E}_d$, where each core of this TT representation is the block matrix

$$\mathscr{E}_\ell = \begin{bmatrix} E_\ell(1,:,:,1) & E_\ell(1,:,:,2) & \cdots & E_\ell(1,:,:,r_\ell) \\ E_\ell(2,:,:,1) & E_\ell(2,:,:,2) & \cdots & E_\ell(2,:,:,r_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ E_\ell(r_{\ell-1},:,:,1) & E_\ell(r_{\ell-1},:,:,2) & \cdots & E_\ell(r_{\ell-1},:,:,r_\ell) \end{bmatrix}, \ \ell = 1,2,\ldots,d.$$

For $\ell = 1,2,\ldots,d$, we let $I_\ell(\alpha_{\ell-1},:,:,\alpha_\ell) \in \mathbb{R}^{n_\ell \times n_\ell}$ be the $n_\ell \times n_\ell$-sized identity matrix for all $\alpha_{\ell-1} = 1,2,\ldots,r_{\ell-1}$ and $\alpha_\ell = 1,2,\ldots,r_\ell$. Finally, let Δ_ℓ^n be the one-dimensional discretization of the partial second derivative $\partial^2/\partial x_\ell$ along the direction ℓ. Precisely, $\Delta_\ell^n = S_n\, D^2$ is the $\big((n+1)\times(n+1)\big)$ -sized matrix operator D^2 introduced in the left-hand side of Eq. (9), premultiplied for the matrix that shifts the evaluation to the collocation grid. We recall that n is the polynomial degree of the spectral collocation approximation. The second order operator can be represented exactly as $\Delta = \mathscr{D} = \mathscr{D}_1 \bowtie \mathscr{D}_2 \bowtie \ldots \bowtie \mathscr{D}_d$, where each $\mathscr{D}_\ell$ contains Δ_ℓ^n. The cores $\mathscr{D}_\ell$ are given by:

$$\mathscr{D}_1 = \begin{bmatrix} \Delta_1^n(1,:,:,1) & I_1(1,:,:,2) & \cdots & I_1(1,:,:,d-1) & I_1(1,:,:,d) \end{bmatrix},$$

$$\mathscr{D}_\ell = \begin{bmatrix} I_\ell(1,:,:,1) & & & & & & \\ & I_\ell(2,:,:,2) & & & & & \\ & & \ddots & & & & \\ & & & \Delta_\ell^n(\ell,:,:,\ell) & & & \\ & & & & \ddots & & \\ & & & & & I_\ell(d-1,:,:,d-1) & \\ & & & & & & I_\ell(d,:,:,d) \end{bmatrix},$$

$$\mathscr{D}_d = \begin{bmatrix} I_d(1,:,:,1) \\ I_d(2,:,:,1) \\ \vdots \\ I_d(d-1,:,:,1) \\ \Delta_d^n(d,:,:,1) \end{bmatrix}.$$

Finally, we consider the approximation $\varepsilon\Delta \approx \mathscr{E} \cdot \mathscr{D}$, where the dot product is defined at algorithm 4 in[13], and the superscript TT is dropped.

The discrete convection operator.

A similar reasoning can be applied to the first-order differential operator $\boldsymbol{\beta}\cdot\nabla$, where $\boldsymbol{\beta}$ is a vector coefficient with d components, i.e., $\boldsymbol{\beta} = \big(\beta_1, \beta_2, \ldots, \beta_d\big)^T$, and the partial differential operator ∇ has also d components, i.e., $\nabla = (\partial_1, \partial_2, \ldots, \partial_d)^T$ with the abbreviation $\partial_k = \partial/\partial x_k$, for $k = 1,2,\ldots,d$. We construct a TT representation of $\boldsymbol{\beta}\cdot\nabla$ by splitting the directional derivative along $\boldsymbol{\beta}$ into its directional components and then introducing the TT representation of each term:

$$\boldsymbol{\beta}\cdot\nabla = \sum_{k=1}^{d} \beta_k \partial_k \approx \sum_{k=1}^{d} \beta_k^{TT} \partial_k^{TT} = \boldsymbol{\beta}^{TT} \cdot \nabla^{TT},$$

or, equivalently, by saying that $(\boldsymbol{\beta}\cdot\nabla)^{TT} = \boldsymbol{\beta}^{TT}\cdot\nabla^{TT}$, where $\boldsymbol{\beta}^{TT} = \big(\beta_1^{TT}, \beta_2^{TT}, \ldots, \beta_d^{TT}\big)$ and $\nabla^{TT} = (\partial_1^{TT}, \partial_2^{TT}, \ldots, \partial_d^{TT})$. The TT decomposition of the k component $\beta_k^{TT} = \beta_1^{(k)} \bowtie \beta_2^{(k)} \bowtie \ldots \bowtie \beta_d^{(k)}$ is written using the cores $\beta_\ell^{(k)}$. As for the diffusion operator, we set $B_\ell^{(k)}(\alpha_{\ell-1},:,:,\alpha_\ell) = \mathrm{diag}\left(\beta_\ell^{(k)}(\alpha_{\ell-1},:,\alpha_\ell)\right)$; we rewrite β_k^{TT} as $\mathscr{B}_k^{TT} = \mathscr{B}_1^{(k)} \bowtie \mathscr{B}_2^{(k)} \bowtie \ldots \bowtie \mathscr{B}_d^{(k)}$ where:

$$\mathscr{B}_\ell^{(k)} = \begin{bmatrix} B_\ell^{(k)}(1,:,:,1) & B_\ell^{(k)}(1,:,:,2) & \cdots & B_\ell^{(k)}(1,:,:,r_\ell^{(k)}) \\ B_\ell^{(k)}(2,:,:,1) & B_\ell^{(k)}(2,:,:,2) & \cdots & B_\ell^{(k)}(2,:,:,r_\ell^{(k)}) \\ \vdots & \vdots & \ddots & \vdots \\ B_\ell^{(k)}(r_{\ell-1}^{(k)},:,:,1) & B_\ell^{(k)}(r_{\ell-1}^{(k)},:,:,2) & \cdots & B_\ell^{(k)}(r_{\ell-1}^{(k)},:,:,r_\ell^{(k)}) \end{bmatrix}.$$

The TT representation of the k-th directional derivative $\partial_k^{TT} = \mathscr{N}_1^{(k)} \bowtie \mathscr{N}_2^{(k)} \bowtie \ldots \bowtie \mathscr{N}_d^{(k)}$ follows the same principals as the second order operator. We denote by ∂_k^n the (matrix) operator representing the one-dimensional partial derivative along the direction k, premultiplied by the interpolation matrix S_n. The resulting cores $\mathscr{N}_\ell^{(k)}$ for all k are given by

$$\mathscr{N}_1^{(k)} = \begin{bmatrix} \partial_1^n(1,:,:,1) & I_1(1,:,:,2) & \cdots & I_1(1,:,:,d-1) & I_1(1,:,:,d) \end{bmatrix},$$

$$\mathscr{N}_\ell^{(k)} = \begin{bmatrix} I_\ell(1,:,:,1) & & & & & & \\ & I_\ell(2,:,:,2) & & & & & \\ & & \ddots & & & & \\ & & & \partial_k^n(\ell,:,:,\ell) & & & \\ & & & & \ddots & & \\ & & & & & I_\ell(d-1,:,:,d-1) & \\ & & & & & & I_\ell(d,:,:,d) \end{bmatrix}, \ \ell = 2,\dots,d-1,$$

$$\mathscr{N}_d^{(k)} = \begin{bmatrix} I_d(1,:,:,1) \\ I_d(2,:,:,1) \\ \vdots \\ I_d(d-1,:,:,1) \\ \partial_d^n(d,:,:,1) \end{bmatrix}.$$

With this notation, the advection operator can be written as

$$\boldsymbol{\beta}\cdot\nabla \approx \sum_{k=1}^{d} \mathscr{B}_k^{TT} \cdot \partial_k^{TT} = \sum_{k=1}^{d} \mathscr{D}_k \cdot \partial_k$$

where the $\cdot$ on the right hand side is the matrix-by-matrix product defined at algorithm 4 in[13], and the superscript TT is dropped.

The discrete reaction operator. Following the same approach used for the diffusion and convection operators, we now construct the discrete reaction operator in TT format. The low-rank, TT representation of ρ reads as $\rho^{TT} = \rho_1 \bowtie \rho_2 \bowtie \dots \bowtie \rho_d$, where the cores are given in the core notation by

$$\rho_\ell = \begin{bmatrix} \rho_\ell(1,:,1) & \rho_\ell(1,:,2) & \cdots & \rho_\ell(1,:,r_\ell) \\ \rho_\ell(2,:,1) & \rho_\ell(2,:,2) & \cdots & \rho_\ell(2,:,r_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ \rho_\ell(r_{\ell-1},:,1) & \rho_\ell(r_{\ell-1},:,2) & \cdots & \rho_\ell(r_{\ell-1},:,r_\ell) \end{bmatrix}, \ \ell = 1,2,\dots,d.$$

Then, we introduce the reaction term $\mathscr{R}$, which is represented through the core sequence $\mathscr{R}^{TT} = \mathscr{R}_1 \bowtie \mathscr{R}_2 \bowtie \dots \bowtie \mathscr{R}_d$, where each core corresponds to a spatial dimension. The ℓ-th core is given by

$$\mathscr{R}_\ell = \begin{bmatrix} R_\ell(1,:,:,1) & R_\ell(1,:,:,2) & \cdots & R_\ell(1,:,:,r_\ell) \\ R_\ell(2,:,:,1) & R_\ell(2,:,:,2) & \cdots & R_\ell(2,:,:,r_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ R_\ell(r_{\ell-1},:,:,1) & R_\ell(r_{\ell-1},:,:,2) & \cdots & R_\ell(r_{\ell-1},:,:,r_\ell) \end{bmatrix}, \ \ell = 1,2,\dots,d,$$

where, for $\alpha_{\ell-1} = 1,2,\dots,r_{\ell-1}$ and $\alpha_\ell = 1,2,\dots,r_\ell$, we set $R_\ell(\alpha_{\ell-1},:,:,\alpha_\ell) = \mathrm{diag}(\rho_\ell(\alpha_{\ell-1},:,\alpha_\ell))$, so that every $R_\ell(\alpha_{\ell-1},:,:,\alpha_\ell)$ is a two-dimensional, diagonal matrix with $\rho_\ell(\alpha_{\ell-1},:,\alpha_\ell)$ along the diagonal. Finally, we note that the diagonal structure of the matrices reflects the multiplicative nature of the reaction term.

The space–time formulation.

The final space–time differential operator includes the time derivative term:

$$L_n := D_t \otimes \bigotimes_{\ell=1}^{d} I_\ell + I_t \otimes \left[-\ \mathscr{E}\cdot\mathscr{D} + \sum_{k=1}^{d} \mathscr{B}_k \cdot \mathscr{N}_k + \mathscr{R} \right],$$

where D_t e I_t are the discrete derivative and the identity matrix relative to the time discretization, where the time operator is the approximation of $\partial/\partial t$. $\otimes$ is the tensor product of two arrays, and $\bigotimes$ is the tensor product of multiple arrays. $\mathscr{E}\cdot\mathscr{D}$ is the TT approximation of the discrete diffusion operator $\varepsilon\Delta$. $\sum_{k=1}^{d} \mathscr{B}_k \cdot \mathscr{N}_k$ is the TT approximation of the discrete advection operator $\boldsymbol{\beta}\cdot\nabla$, and $\mathscr{R}$ is the TT approximation of the discrete reaction operator. The superconsistent version of the scheme is achieved by evaluating the terms $\mathscr{B}$, $\mathscr{E}$ and $\mathscr{R}$ on the superconsistent grid.

Boundary conditions. To apply Dirichelet boundary conditions in full format we set to zero the rows that correspond to the boundary points. In TT format that is achieved by setting to zero the first and the last rows of the one dimensional discrete operators described in the previous paragraphs. In full format we need to add a diagonal matrix that is one corresponding to the boundary nodes and zero elsewhere, namely $\mathcal{I}_{\mathrm{bc}}$, in Fig. 12.

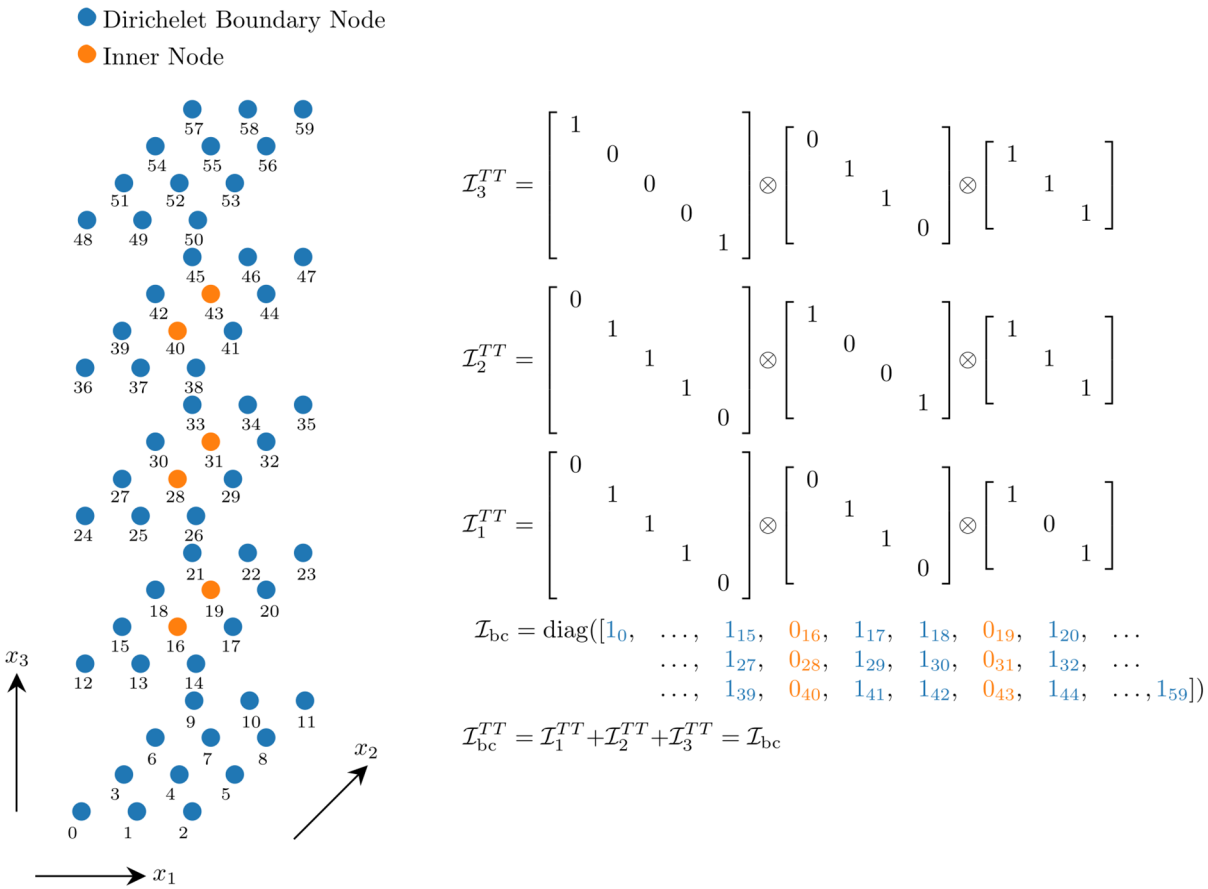


Fig. 12. In this picture, a three dimensional example illustrates the procedure to represent the diagonal matrix with ones corresponding to the boundary nodes and zeros elsewhere. The matrix $\mathcal{I}_{\mathrm{bc}}^{TT}$ is added to the overall system matrix with zero rows corresponding to the boundary nodes.

Figure 12 presents a simple three dimensional grid with highlighted the boundary and the inner nodes. $\mathcal{I}_i^{TT}$ sets to one the elements corresponding to the x_i direction. $\mathcal{I}_{\mathrm{bc}}^{TT}$ is the sum of $\mathcal{I}_i^{TT}$ with $i = 1, 2, 3$, and exactly corresponds to $\mathcal{I}_{\mathrm{bc}}$.

Data availability

Data is provided within the manuscript or supplementary information files.

Received: 15 May 2025; Accepted: 20 August 2025
Published online: 06 January 2026

References

1. Zank, G. P. *Transport Processes in Space Physics and Astrophysics* (Springer, 2014).
2. Williams, M. M. R. *Random Processes in Nuclear Reactors* (Pergamon, 2013).
3. Jüngel, A. Transport equations for semiconductors. In *Lecture Notes in Physics* (Springer, 2009).
4. Cercignani, C. *The Boltzmann Equation and Its Applications. Applied Mathematical Sciences* (Springer, 2012).
5. Bell, G. I. & Glasstone, S. *Nuclear Reactor Theory* (US Atomic Energy Commission (AEC), US Atomic Energy Commission, 1970). (TID-25606, osti_id 4074688, NSA-25-010317).
6. Graziani, F. (ed.) *Computational Methods in Transport* (Springer, 2006).
7. Graziani, F. R. & Olson, G. L. Transport methods conquering the seven-dimensional mountain. In *Technical Report*, UCRL-JC-154432, Lawrence Livermore National Laboratory (2003).
8. Bellman, R. *Dynamic Programming* (Princeton University Press, 1957).
9. Einkemmer, L. & Moriggl, A. Semi-Lagrangian 4D, 5D, and 6D kinetic plasma simulation on large-scale GPU-equipped supercomputers. *Int. J. High Perform. Comput. Appl.* **37**, 180–196. https://doi.org/10.1177/10943420221137599 (2023).
10. Zhou, J., Li, R. & Luo, T. Physics-informed neural networks for solving time-dependent mode-resolved phonon Boltzmann transport equation. *npj Comput. Mater.* **9**, 212. https://doi.org/10.1038/s41524-023-01165-7 (2023).
11. Ruthotto, L., Osher, S. J., Li, W., Nurbekyan, L. & Fung, S. W. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proc. Natl. Acad. Sci.* **117**, 9183–9193. https://doi.org/10.1073/pnas.1922204117 (2020).
12. Einkemmer, L., Kormann, K., Kusch, J., McClarren, R. G. & Qiu, J.-M. A review of low-rank methods for time-dependent kinetic simulations (2024). ArXiv:2412.05912.
13. Oseledets, I. V. Tensor-train decomposition. *SIAM J. Sci. Comput.* **33**, 2295–2317. https://doi.org/10.1137/090752286 (2011).

14. Dolgov, S. V., Khoromskij, B. N. & Oseledets, I. V. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the fokker-planck equation. *SIAM J. Sci. Comput.* **34**, A3016–A3038. https://doi.org/10.1137/120864210 (2012).
15. Dolgov, S. V. & Savostyanov, D. V. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM J. Sci. Comput.* **36**, A2248–A2271. https://doi.org/10.1137/140953289 (2014).
16. Dolgov, S. V. & Savostyanov, D. V. Alternating minimal energy methods for linear systems in higher dimensions. Part II. Faster algorithm and application to nonsymmetric systems. *SIAM J. Sci. Comput.* **37**, A2456–A2488. https://doi.org/10.1137/140953289 (2015).
17. Dolgov, S. V. & Scheichl, R. A hybrid alternating least squares-TT cross algorithm for parametric PDEs. *SIAM/ASA J. Uncertain. Quantif.* **7**, 260–291. https://doi.org/10.1137/18M1194134 (2019).
18. Kormann, K. A semi-Lagrangian Vlasov solver in tensor train format. *SIAM J. Sci. Comput.* **37**, B613–B632. https://doi.org/10.1137/140971270 (2015).
19. Canuto, C., Hussaini, M. Y., Quarteroni, A. & Zang, T. A. *Spectral Methods: Fundamentals in Single Domains* (Springer, 2007).
20. Hesthaven, J. S., Gottlieb, S. & Gottlieb, D. *Spectral Methods for Time-Dependent Problems*. Vol. 21. *Cambridge Monographs on Applied and Computational Mathematics* (Cambridge University Press, 2007).
21. Funaro, D. *Spectral Elements for Transport-Dominated Equations* (Springer, 1997).
22. Arnold, D. N. Numerical analysis of partial differential equations. In *Lecture Notes, University of Minnesota*. 8445-8446 (2018).
23. Fish, J. & Belytschko, T. *A First Course in Finite Elements* (Wiley, 2007).
24. Funaro, D. Superconsistent discretizations. *J. Sci. Comput.* **17**, 67–79. https://doi.org/10.1023/A:1015136227726 (2002).
25. De l'Isle, F. & Owens, R. G. Superconsistent collocation methods with applications to convection-dominated convection-diffusion equations. *J. Comput. Appl. Math.* **391**, 113367. https://doi.org/10.1016/j.cam.2020.113367 (2021).
26. De l'Isle, F. & Owens, R. G. A superconsistent collocation method for high Reynolds number flows. *Comput. Fluids* **259**, 105897. https://doi.org/10.1016/j.compfluid.2023.105897 (2023).
27. Tal-Ezer, H. Spectral methods in time for hyperbolic equations. *SIAM J. Numer. Anal.* **23**, 11–26 (1986).
28. Schiesser, W. E. *The Numerical Method of Lines: Integration of Partial Differential Equations* (Academic Press, 1991).
29. Hundsdorfer, W. & Verwer, J. G. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*. Vol. 33 (Springer, 2003).
30. Ruuth, S. J. & Spiteri, R. J. A new class of optimal high-order strong-stability-preserving time discretization methods. *SIAM J. Numer. Anal.* **40**, 469–491 (2003).
31. Glowinski, R. Finite element methods for incompressible viscous flow. In *Numerical Methods for Fluids (Part 3)*. Vol. 9. *Handbook of Numerical Analysis*. 3–1176. https://doi.org/10.1016/S1570-8659(03)09003-3 (Elsevier, 2003).
32. Hughes, T. J. R. & Mallet, M. A new finite element formulation for computational fluid dynamics: III. The generalized streamline operator for multidimensional advective-diffusive systems. *Comput. Methods Appl. Mech. Eng.* **58**, 305–328. https://doi.org/10.1016/0045-7825(86)90152-0 (1986).
33. Brooks, A. N. & Hughes, T. J. R. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Eng.* **32**, 199–259. https://doi.org/10.1016/0045-7825(82)90071-8 (1982).
34. Malaya, N. et al. MASA: A library for verification using manufactured and analytical solutions. *Eng. Comput.* 1–10. https://doi.org/10.1007/s00366-012-0267-9 (2012).
35. Glowinski, R. & Pan, T. W. *Numerical Simulation of Incompressible Viscous Flow* (De Gruyter, 2022).
36. Wienke, B. R., Neutrons and photons. Transport equations in moving material. Part I. *Prog. Nucl. Energy* **46**, 13–55. https://doi.org/10.1016/j.pnucene.2004.09.016 (2005).
37. Meuer, H., Strohmaier, E., Dongarra, J., Simon, H. & Meuer, M. Top500 (2025) (online article).
38. Smith, H. & Jensen, H. H. *Transport Phenomena* (Clarendon Press, 1989).
39. Ackroyd, R. *Finite Element Methods for Particle Transport: Applications to Reactor and Radiation Physics* (Applications to Reactor and Radiation Physics (Research Studies Press, Finite Element Methods for Particle Transport, 1997).
40. Kutz, J. N. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data* (Oxford University Press Inc, 2013).
41. Kutz, J. N., Brunton, S. L., Brunton, B. W. & Proctor, J. L. *Dynamic Mode Decomposition* (Society for Industrial and Applied Mathematics, 2016). https://epubs.siam.org/doi/pdf/10.1137/1.9781611974508.
42. Hesthaven, J. S., Rozza, G. & Stamm, B. Certified reduced basis methods for parametrized partial differential equations. In *Springer Briefs in Mathematics*. 1 Ed (Springer, 2015).
43. Bungartz, H.-J. & Griebel, M. Sparse grids. *Acta Numer.* **13**, 147–269. https://doi.org/10.1017/S0962492904000182 (2004).
44. Peskin, C. S. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.* **10**, 252–271. https://doi.org/10.1016/0021-9991(72)90065-4 (1972).
45. Glowinski, R., Pan, T., Hesla, T., Joseph, D. & Périaux, J. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: Application to particulate flow. *J. Comput. Phys.* **169**, 363–426. https://doi.org/10.1006/jcph.2000.6542 (2001).
46. Boffi, D., Cavallini, N. & Gastaldi, L. The finite element immersed boundary method with distributed Lagrange multiplier. *SIAM J. Numer. Anal.* **53**, 2584–2604. https://doi.org/10.1137/140978399 (2015).
47. Reports-exascale computing project (2025). https://www.exascaleproject.org/reports/ . Accessed 9 Feb 2025 (2025).
48. Advanced Reactor Information System (2025). https://aris.iaea.org/. Accessed 9 Feb 2025 (2025).
49. IAEA World Fusion Outlook 2024 (2024). https://www.iaea.org/publications/15777/iaea-world-fusion-outlook-2024. Accessed 21 Feb 2025 (2025).
50. Geist, W. H. et al. (eds) *Nondestructive Assay of Nuclear Materials for Safeguards and Security* (Springer, 2024).
51. Cavallini, N. et al. Vanquishing the computational cost of passive gamma emission tomography simulations leveraging physics-aware reduced order modeling. *Sci. Rep.* **13**, 15034. https://doi.org/10.1038/s41598-023-41220-3 (2023).
52. Singh, J., Kumar, N. & Jiwari, R. A robust weak Galerkin finite element method for two parameter singularly perturbed parabolic problems on nonuniform meshes. *J. Comput. Sci.* **77**, 102241. https://doi.org/10.1016/j.jocs.2024.102241 (2024).
53. Yadav, O. P. & Jiwari, R. A finite element approach to capture turing patterns of autocatalytic Brusselator model. *J. Math. Chem.* **57**, 769–789. https://doi.org/10.1007/s10910-018-0982-6 (2019).
54. Hadamard, J. *Sur les problèmes aux dérivées partielles et leur signification physique* (Princeton University Bulletin, 1902).
55. Gottlieb, D. & Orszag, S. A. Numerical analysis of spectral methods: Theory and applications. *SIAM* **26** (1977).
56. Maday, Y. & Quarteroni, A. Spectral and pseudo-spectral approximations of the Navier–Stokes equations. *SIAM J. Numer. Anal.* **16**, 573–595 (1979).
57. Funaro, D. Superconsistent discretizations of integral type equations. *Appl. Numer. Math.* **48**, 1–11. https://doi.org/10.1016/j.apnum.2003.05.007 (2004).
58. Fatone, L., Funaro, D. & Scannavini, V. Finite-difference preconditioners for superconsistent pseudospectral approximations.. *ESAIM Modél. Math. Anal. Numér.-PIPPO* **41**, 1021–1039. https://doi.org/10.1051/m2an:2007052 (2007).
59. Kolda, T. & Bader, B. Tensor decompositions and applications. *SIAM Rev.* **51**, 455–500 (2009).
60. Oseledets, I. & Tyrtyshnikov, E. TT-cross approximation for multidimensional arrays. *Linear Algebra Appl.* **432**, 70–88 (2010).

61. De Launey, W. & Seberry, J. The strong Kronecker product. *J. Comb. Theory Ser. A* **66**, 192–213. https://doi.org/10.1016/0097-3165(94)90062-0 (1994).
62. Goreinov, S., Oseledets, I., Savostyanov, D., Tyrtyshnikov, E. & Zamarashkin, N. How to find a good submatrix. In (Olshevsky, V. & Tyrtyshnikov, E. eds.) *Matrix Methods: Theory, Algorithms and Applications*. 247–256. https://doi.org/10.1142/9789812836021_0015 (World Scientific, 2010).

Acknowledgements
A. Favalli and N. Cavallini warmly acknowledge the support of the Joint Research Centre of the European Commission. The LDRD-ER Project #20210485ER and the LDRD-DR Project #20250032DR funded the work of Dr. G. Manzini. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy (Contract No. 89233218CNA000001). D. Funaro and G. Manzini are also members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Author contributions
N.C.: methodology, analysis, and coding, G.M, D.F.: methodology, analysis, validation. A.F.: conceived the concept and the study, analysis, and supervised the project. All authors contributed to the preparation of the manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-16905-6.

Correspondence and requests for materials should be addressed to A.F.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

© European Union, Los Alamos National Laboratory, and Daniele Funaro 2026