Journal of Computational Physics 435 (2021) 110266

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Efficient projection kernels for discontinuous Galerkin simulations of disperse multiphase flows on arbitrary curved elements

Eric J. Ching *, Matthias Ihme

Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA

ARTICLE INFO

Article history:
Available online 9 March 2021

Keywords:
Discontinuous Galerkin method
Lagrangian particle tracking
Particle-laden flow
Curved elements
Euler-Lagrange

ABSTRACT

In this work, we develop projection kernels for Euler-Lagrange point-particle simulations of disperse multiphase flows on arbitrary curved elements. These kernels are employed in a high-order discontinuous Galerkin framework for projecting the action of the particles to the Eulerian mesh. Instead of commonly used isotropic kernels, such as a Gaussian-type kernel, we construct an anisotropic polynomial-based smoothing function that preserves the compactness of the discontinuous Galerkin method on high-aspect-ratio elements and maintains an acceptable computational cost. At the same time, it mitigates inaccuracies associated with larger particles and numerical instabilities arising from the Dirac delta function and low-order kernels. Specifically, the geometric mapping of a physical element to the reference element is exploited to construct a kernel that is elliptical in 2D and ellipsoidal in 3D. We also develop a strategy to conserve interphase transfer near boundaries, particularly on curved elements. This strategy employs a high-order polynomial approximation to appropriately rescale the interphase source terms in an efficient manner. The compatibility of the proposed methodology with different types of meshes is investigated. We then apply it to a number of particle-laden flow configurations, including supersonic dusty flow over a flat plate, moving shocks interacting with clouds of particles, and hypersonic dusty flows over blunt bodies.

1. Introduction

Disperse multiphase flows comprise an important area of science and engineering. A common strategy to numerically simulate these flows is the Euler-Lagrange approach, in which the fluid, or carrier phase, is solved in an Eulerian frame while the particles, or disperse phase, is treated in a Lagrangian manner. At low volume fractions, the particles have very little influence on the carrier fluid and can thus be ignored. This is known as one-way coupling since only the effect of the fluid on the particles is accounted for. Higher volume fractions necessitate a two-way-coupled approach, in which the effect of the particles on the fluid is taken into consideration.

In two-way coupled simulations, the reverse coupling of the particles to the gas is handled via a projection step that translates disperse phase information to the Eulerian frame. Specifically, a projection kernel is employed that distributes the influence of each particle to the Eulerian solver. The projection kernel has also been referred to in the literature using other

* Corresponding author.
E-mail address: eching@stanford.edu (E.J. Ching).

https://doi.org/10.1016/j.jcp.2021.110266
0021-9991/

terms such as mollification kernel [1], weighing function [2], shape function [3], and smoothing function [4]. Euler-Lagrange simulations are commonly performed within a finite-volume framework. In this context, a very simple kernel is the box kernel, in which the back-coupling of a given particle is fully transferred to a single cell [5]. This approach is local, efficient, and easy to implement. However, the piecewise-constant representation can lead to numerical noise and instabilities, especially in high-order simulations, due to the jumps at element interfaces [3]. Inaccuracies can also occur since the influence of a particle should decrease with distance from the particle. A more sophisticated strategy is to project particle information to neighboring points based on a volume-weighted [6] or distance-weighted scheme [7]. Smooth isotropic kernels, such as a Gaussian smoothing function [1,3,4,8], have been adopted as of late. The kernel shape is circular in 2D and spherical in 3D. Although this can expand the numerical stencil and therefore increase computational cost, it has a maximum at its center and smoothly decays away from the particle.

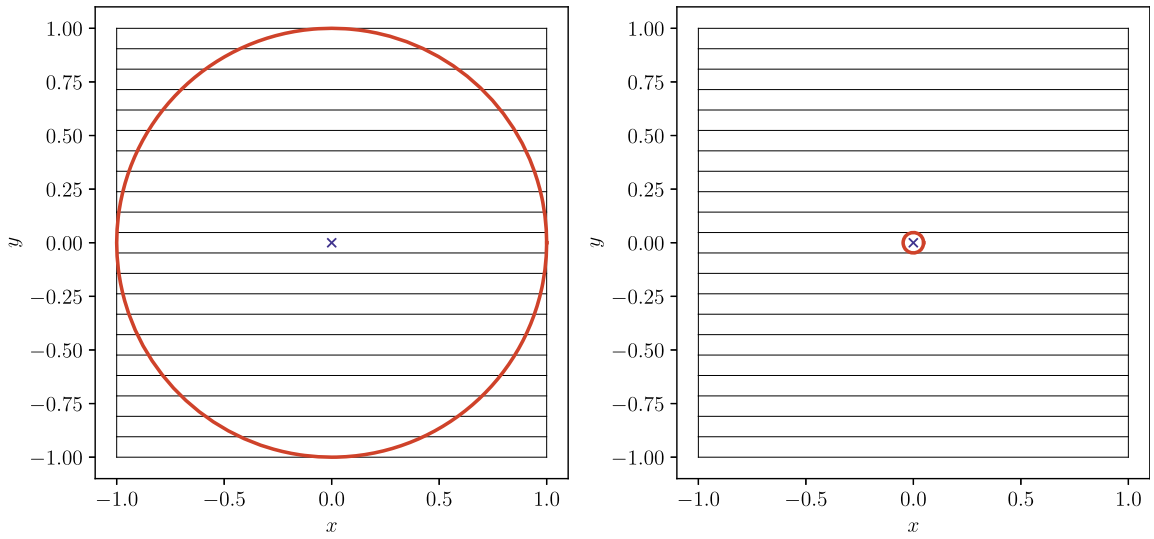
Recently, high-order schemes (with an order of accuracy greater than two) have been employed to solve the carrier fluid. For instance, high-order finite-difference [2] and spectral-based methods [9,10] have been successfully used in Euler-Lagrange point-particle simulations. An increasingly popular high-order scheme is the discontinuous Galerkin (DG) method [11], which benefits from not only arbitrarily high order of accuracy but also geometric flexibility, *hp*-adaptation (where h refers to the mesh and p refers to the order of the solution approximation), and a compact stencil. Only information from face-sharing neighboring elements needs to be transferred, regardless of order of accuracy. This makes the DG method suited for large-scale parallel simulations. In previous work, we developed a two-way coupled Euler-Lagrange methodology for DG schemes. One of our major objectives was the ability to accurately track particles on curved elements [12], which are necessary for representing curved geometries in DG simulations. In particular, a major challenge when treating hard-sphere particle-wall collisions on curved, high-aspect-ratio elements is that simple geometric arguments for localizing particles on planar elements can no longer be applied. In addition, pathological situations can occur that are not observed on straight-sided elements. We proposed algorithms to efficiently and accurately compute particle-wall collisions on arbitrary curved elements, as well as deal with the aberrant situations. To treat the reverse coupling of particles to the fluid, we employed Dirac delta weighing functions. This type of kernel is simple to implement and computationally efficient since the source term integral can then be analytically evaluated. Furthermore, unlike the aforementioned box kernel, the kernel has a maximum at its center and varies in space within the element. While appropriate for flows in which the back-coupled effect is weak to moderate, the lack of smoothness can result in noise as well as numerical instabilities when the particles significantly affect the carrier fluid, especially in high-order simulations. Furthermore, for large particles, approximating their region of influence as points can lead to inaccuracies.

Other researchers have recently performed DG simulations of particle-laden flows as well. Zwick and Balachandar [13] applied the DG method to simulate a rapidly depressurized gas-solid flow in a shock tube, and Huang et al. [8] combined it with an implicit-explicit Runge-Kutta scheme to compute particle-vortex interaction. In both of these studies, isotropic Gaussian kernels were employed [4,8]. Jacobs and Hesthaven [3] investigated isotropic Gaussian-, polynomial-, and cosine-based shape functions in the context of a DG Maxwell field solver for simulating 2D plasma dynamics. Stock et al. [14] and Pfeiffer et al. [15] computed complex 3D plasma applications by making use of the aforementioned polynomial-based shape function.

To deal with flows with strong reverse coupling and/or large particles, a smooth isotropic kernel may seem a natural choice. However, such a kernel can cause issues when using high-aspect-ratio elements, which are commonly employed in high-gradient regions such as boundary layers. For instance, consider the kernel displayed in Fig. 1(a), which nearly covers the entirety of the host element. Due to the high aspect ratio of the elements, the kernel extends into a very large number of elements, which not only significantly increases computational cost and implementation complexity but also eliminates the favorable compactness of the DG method. Furthermore, for a highly stretched and/or unstructured mesh, it is not clear how many elements with which such a kernel overlaps. It is even possible for the number of overlapping elements to vary significantly as a function of the precise location of the kernel center inside a given host element. Note that these problems are exacerbated in 3D. On the other hand, consider the kernel shown in Fig. 1(b), which spreads over only the host element. Due to its small size, it is similar to a delta function and is therefore susceptible to the same issues discussed previously. However, there is an additional disadvantage: given that the resulting source term integral cannot be easily evaluated in analytical form, an exceedingly high number of quadrature points may be required to adequately capture the kernel shape.

The difficulties described above are observed on all high-aspect-ratio elements, irrespective of whether the elements are planar or curved. However, an additional major challenge distinct from those discussed earlier and specific to curved elements is near-wall treatment. On straight-sided elements, a common approach to account for the portion of the kernel that lies outside the domain is to mirror the given particle across the boundary and superimpose the kernel of the image particle [1,3]. However, for a curved element, it is not straightforward to determine the correct position of the image particle such that interphase transfer is conserved. Doing so would likely require an expensive iterative procedure that must be executed for every single particle located in a near-wall element at every time step.

In light of the above, our main objective is to develop a stable, accurate methodology for projecting the effect of the disperse phase to the Eulerian frame that can be efficiently applied to arbitrary curved, high-aspect-ratio elements. In particular, the proposed methodology consists of the following two components, one to address high-aspect-ratio elements and the other for handling curved elements near boundaries:



(a) Large isotropic kernel (b) Small isotropic kernel

Fig. 1. Examples of large and small isotropic kernels on high-aspect-ratio elements. The kernel center is indicated by the blue cross, and the kernel boundary is in red. Element boundaries are drawn with black lines. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

1. *Smooth anisotropic projection kernels*, constructed such that the influence of a given kernel generally does not extend past the node-sharing elements. Although some accuracy is likely lost due to the forced mesh-dependent anisotropy, which itself can be mollified via an extra diffusion step [1], this represents a good compromise among fidelity, mitigation of numerical instabilities and noise, and computational cost. In this way, the overall compactness of the DG method is maintained. Instead of a Gaussian-type kernel, we employ a polynomial-based function that gives lower projection error and zero cutoff error. The size and orientation of the kernels are computed from the mesh-implied metric [16,17], commonly employed in mesh adaptation strategies, which relies on the geometric mapping of a physical element to the reference element. The resulting kernel shape is elliptical in 2D and ellipsoidal in 3D.
2. *Efficient kernel rescaling* in elements near walls in order to conserve interphase transfer. However, instead of calculating the exact rescaling factor using a brute-force approach, which would need to be done for all particles in near-wall elements at every time step, we employ high-order polynomials to provide a continuous approximation of the rescaling factor within each element. The polynomial approximation can be obtained in the preprocessing stage at the beginning of the simulation. This yields an efficient strategy to treat curved walls discretized with high-order elements.

The remaining sections are organized as follows. Section 2 briefly discusses the governing equations of the two phases. Section 3 details the proposed methodology, particularly the motivation for the polynomial-based smoothing function, the construction of smooth anisotropic kernels, and the near-wall treatment. The basic performance of the methodology is investigated on different types of meshes. In Section 4, we apply it to various test cases, including supersonic dusty flow over a flat plate, moving shocks interacting with clouds of particles, and hypersonic dusty flows over blunt bodies.

2. Governing equations

This section describes the governing equations for the carrier and disperse phases. We also summarize the DG discretization, as well as the algorithmic details of the particle solver introduced in Ref. [12].

2.1. Fluid equations

The carrier phase is described by the compressible Navier-Stokes equations, given as

$$\partial_t \boldsymbol{U} + \nabla \cdot \boldsymbol{F}_s = \nabla \cdot \boldsymbol{F}_v + \boldsymbol{S}, \tag{1}$$

where $\boldsymbol{U}(\boldsymbol{x}, t)$ is the conservative state vector, which varies with spatial position, $\boldsymbol{x}$, and time, t, $\boldsymbol{F}_s(\boldsymbol{U})$ is the inviscid flux, $\boldsymbol{F}_v(\boldsymbol{U}, \nabla\boldsymbol{U})$ is the viscous flux, and $\boldsymbol{S}(\boldsymbol{U}, \nabla\boldsymbol{U})$ is the source term vector accounting for the effect of the disperse phase projected onto the Eulerian mesh. The projection step will be discussed in Section 3. These can be expanded as

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho \boldsymbol{u} \\ \rho E \end{bmatrix}, \quad \boldsymbol{F}_s = \begin{bmatrix} \rho \boldsymbol{u} \\ \rho \boldsymbol{u} \otimes \boldsymbol{u} + P\mathbb{I} \\ \boldsymbol{u}(\rho E + P) \end{bmatrix}, \quad \boldsymbol{F}_v = \begin{bmatrix} 0 \\ \boldsymbol{\tau} \\ \boldsymbol{u} \cdot \boldsymbol{\tau} - \boldsymbol{q} \end{bmatrix}, \quad \boldsymbol{S} = \begin{bmatrix} S_\rho \\ \boldsymbol{S}_m \\ S_E \end{bmatrix}, \tag{2}$$

where ρ is the fluid density, $\boldsymbol{u}$ is the velocity, P is the pressure, E is the total energy per unit mass, and $\mathbb{I}$ is the identity matrix. S_ρ, $\boldsymbol{S}_m$, and S_E are the back-coupled mass, momentum, and energy transfer. Pressure can be computed from the total energy as

$$P = (\gamma - 1)\left(\rho E - \frac{\rho}{2}|\boldsymbol{u}|^2\right), \tag{3}$$

where γ is the ratio of specific heats, set to a value of 1.4 in this study, and $c_p = R\gamma/(\gamma - 1)$ is the specific heat capacity at constant pressure, with R denoting the specific gas constant. The dynamic viscosity, μ, is computed using Sutherland's law, and κ, the thermal conductivity, is obtained from the Prandtl number, Pr.

2.2. Discontinuous Galerkin discretization

Equation (2) is discretized using the DG method. The computational domain, Ω, is subdivided into N_e elements such that $\Omega = \cup_{e=1}^{N_e} \Omega_e$, and $\partial\Omega_e$ denotes the boundary of element Ω_e. The local solution is approximated by a polynomial of order p as

$$\boldsymbol{U}_h^e(t, \boldsymbol{x}) = \sum_{n=1}^{N_b} \widetilde{\boldsymbol{U}}_n^e(t)\phi_n(\boldsymbol{x}), \tag{4}$$

where ϕ_n is the nth polynomial basis function and $\widetilde{\boldsymbol{U}}_n^e$ is the vector of coefficients for the nth basis function. In this work, we employ a Lagrange basis. To solve for the polynomial coefficients, we require $\boldsymbol{U}_h^e$ to satisfy

$$\sum_{n=1}^{N_b} d_t \widetilde{\boldsymbol{U}}_n^e(t) \int_{\Omega_e} \phi_m \phi_n d\Omega + \int_{\Omega_e} \phi_m \nabla \cdot \boldsymbol{F}_{\boldsymbol{s}} d\Omega = \int_{\Omega_e} \phi_m \nabla \cdot \boldsymbol{F}_{\boldsymbol{v}} d\Omega + \int_{\Omega_e} \phi_m \boldsymbol{S} d\Omega \;\; \forall \, \phi_m, \tag{5}$$

where $m = 1, ..., N_b$. The global solution approximation is then given as

$$\boldsymbol{U}_h(t, \boldsymbol{x}) = \bigoplus_{e=1}^{N_e} \boldsymbol{U}_h^e(t, \boldsymbol{x}). \tag{6}$$

To solve Eq. (5), integration by parts is performed on the convective and viscous terms, and numerical fluxes are applied. In this study, we employ the Roe inviscid flux function [18] and the BR2 scheme [19]. To compute the integrals (apart from the source term integral, which will be discussed in Section 3), numerical quadrature with an order of accuracy no less than $2p + 1$ is used. To capture shocks, we use intraelement variations for detection and smooth artificial viscosity for stabilization [20]. No additional limiting is performed.

2.3. Disperse phase governing equations

We now describe the governing equations for the disperse phase. For simplicity, we make the following assumptions: each particle is considered to be spherical, non-rotating, and mass-preserving ($S_\rho = 0$); particle-particle interactions are neglected; the temperature of a given particle is uniform; the flow is sufficiently dilute that the particle volume fraction can be ignored. Each particle is governed by the following equations for position, momentum, and energy:

$$\frac{d\boldsymbol{x}_d}{dt} = \boldsymbol{u}_d, \tag{7a}$$

$$m_d \frac{d\boldsymbol{u}_d}{dt} = \boldsymbol{F}, \tag{7b}$$

$$m_d c_d \frac{dT_d}{dt} = Q, \tag{7c}$$

where the subscript "d" denotes the disperse phase. $\boldsymbol{x}_d$ is the position of the centroid of the particle, $\boldsymbol{u}_d$ is the particle velocity, c_d is the specific heat of the particle, and T_d is the particle temperature. $m_d = \rho_d \frac{\pi}{6} D^3$ is the particle mass, where ρ_d is the particle density and D is the particle diameter. $\boldsymbol{F}$ and Q are the overall drag and heating rate on the particle. Unless otherwise specified, contributions to interphase momentum and energy transfer other than the quasi-steady terms are neglected [21]. These quasi-steady contributions are written as

$$\boldsymbol{F}_{qs} = \frac{1}{8}\pi D^2 \rho_c (\boldsymbol{u}_c - \boldsymbol{u}_d)|\boldsymbol{u}_c - \boldsymbol{u}_d| C_D, \tag{8a}$$

$$Q_{qs} = \pi D \kappa_c (T_c - T_d)\mathrm{Nu}, \tag{8b}$$

where the subscript "c" indicates the carrier fluid. C_D and Nu are the drag coefficient and Nusselt number, respectively, which in general are functions of the relative particle Reynolds number and the relative particle Mach number, given by

$$\mathrm{Re}_d = \frac{\rho_c |\boldsymbol{u}_c - \boldsymbol{u}_d| D}{\mu_c}, \tag{9a}$$

$$\mathrm{Ma}_d = \frac{|\boldsymbol{u}_c - \boldsymbol{u}_d|}{\sqrt{\gamma R T_c}} \tag{9b}$$

The specific drag and Nusselt number correlations employed for each test case will be described in the corresponding sections.

2.4. Particle tracking algorithm

At each time step, the host element of a particle must be identified. An efficient algorithm for particle search and localization is that by Allievi and Bermejo [22], which is applicable to arbitrary curved elements on unstructured meshes. The algorithm exploits the geometric mapping of a physical element to a reference element. An output of the algorithm is the position of the particle in reference space. This is used to interpolate the state of the carrier fluid to the particle location using the polynomial approximation of the carrier fluid (Eq. (4)). This maintains the spatial order of accuracy of the solution of the carrier phase. $\boldsymbol{F}$ and Q in Eqs. (8) are then computed, and the particle is advanced in time. To reduce cost, each computational particle is used to represent multiple physical particles [21].

As discussed in Section 1, a major difficulty associated with the tracking of particles on curved, high-aspect-ratio elements is the treatment of particle-wall collisions. In previous work [12], we developed algorithms to compute such collisions in an efficient and accurate manner. The proposed algorithms can also deal with pathological scenarios that can occur when using curved, high-aspect-ratio elements. The use of curved elements can significantly improve predictions of particle trajectories in problems involving particle-wall collisions [12].

3. Reverse coupling

In this section, we detail the proposed methodology for projecting the action of the disperse phase to the Eulerian frame. We begin with an overview of the basic equations and a comparison between two types of kernels: a Gaussian function and a polynomial-based function. Next, we discuss the construction of an appropriate anisotropic kernel, followed by the treatment of curved elements near walls. Several types of example meshes are employed to illustrate these two components of the formulation.

3.1. Back-coupling source term

The effect of the disperse phase on the carrier phase is considered via the source term vector, $\boldsymbol{S} = [S_\rho, \boldsymbol{S}_m, S_E]^T$, on the RHS of Eq. (1). The individual terms can be expanded as

$$S_\rho = 0, \tag{10a}$$

$$\boldsymbol{S}_m = -\sum_{i=1}^{N_p} \boldsymbol{F}_i \chi(\boldsymbol{x}; \boldsymbol{x}_{d,i}), \tag{10b}$$

$$S_E = -\sum_{i=1}^{N_p} \left(Q_i + \boldsymbol{u}_{d,i} \cdot \boldsymbol{F}_i \right) \chi(\boldsymbol{x}; \boldsymbol{x}_{d,i}), \tag{10c}$$

where i indexes the particles, N_p is the total number of particles in the domain, and the second term in Eq. (10c) is the work done by the overall particle drag. $\chi(\boldsymbol{x}; \boldsymbol{x}_{d,i})$ is the projection kernel that distributes the particle influence to the Eulerian mesh and satisfies the following property:

$$\int_\Omega \chi \, d\Omega = 1. \tag{11}$$

The requirement that the integral of the kernel over the computational domain is unity (and dimensionless) ensures conservation of information transfer between the two phases. Another desirable property is that the kernel maximum is located at the particle centroid [2], i.e.

$$\arg\max_{\boldsymbol{x} \in \Omega} \chi(\boldsymbol{x}; \boldsymbol{x}_d) = \boldsymbol{x}_d. \tag{12}$$

3.2. Dirac delta shape function

Previously, we employed the shifted Dirac delta function, $\chi(\boldsymbol{x};\boldsymbol{x}_{d,i}) = \delta(\boldsymbol{x}-\boldsymbol{x}_{d,i})$ [12]. By using this choice of shape function, the source term in Eq. (5) can be evaluated analytically as

$$\int_{\Omega_e} \phi_m \boldsymbol{S} d\Omega = \sum_{i=1}^{N_p^e} \phi_m(\boldsymbol{x}_{d,i})[0, \boldsymbol{F}_i, Q_i + \boldsymbol{u}_{d,i}\cdot\boldsymbol{F}_i]^T, \tag{13}$$

where N_p^e is the number of particles whose centroids lie in element Ω_e. Since quadrature is not needed, this kernel benefits from extremely low computational cost and straightforward implementation. Furthermore, there is no additional computational or implementation overhead associated with running parallel simulations, unlike for finite-size kernels (as will be discussed later in this section). Note that in $p=0$ simulations, for which $N_b = 1$ and $\phi_1 = 1$, this shape function gives the same results as the box kernel (assuming appropriate normalization). In this case, the RHS of Eq. (13) is independent of $\{\boldsymbol{x}_{d,i}\}_{i=1}^{N_p^e}$. For $p>0$, on the other hand, it is indeed a function of the specific locations of the particles inside the element, which for the box kernel remains the opposite. Coupled with the fact that it also satisfies Eq. (12), the Dirac delta function is higher-fidelity than the box kernel and is a good choice for flow configurations with small particles and moderate back-coupling. However, in high-order simulations involving large particles and/or significant influence of the disperse phase, the delta shape function can lead to reduced accuracy and Gibbs-type instabilities.

3.3. Finite-size, isotropic kernels

We now consider two smoothly varying isotropic projection kernels. For illustration, we focus primarily on the 2D case, but the 1D and 3D cases follow directly. The first kernel is the popular Gaussian kernel,

$$\chi_g(\boldsymbol{x};\boldsymbol{x}_d) = \begin{cases} \dfrac{1}{2\pi\sigma^2}\exp\left[-\dfrac{1}{2}\dfrac{r^2}{\sigma^2}\right], & r \le R_c, \\ 0, & r > R_c, \end{cases} \tag{14}$$

where σ^2 is the variance and $r = |\boldsymbol{r}| = |\boldsymbol{x}-\boldsymbol{x}_d|$ is distance from the particle centroid. The characteristic width of the kernel, i.e. the full width at half the maximum height, is $\delta_f = 2\sqrt{2\ln 2}\sigma$. To enforce compact support, the kernel is truncated at the cutoff distance, R_c, which defines the *extent* of the kernel. Ideally, the kernel should be close to zero at the cutoff to minimize the cutoff error. Kernel shapes for different values of δ_f are shown in Fig. 2(a). For smaller δ_f, the peak is higher and the profile is steeper near the center.

The second kernel is the polynomial function presented in Ref. [3],

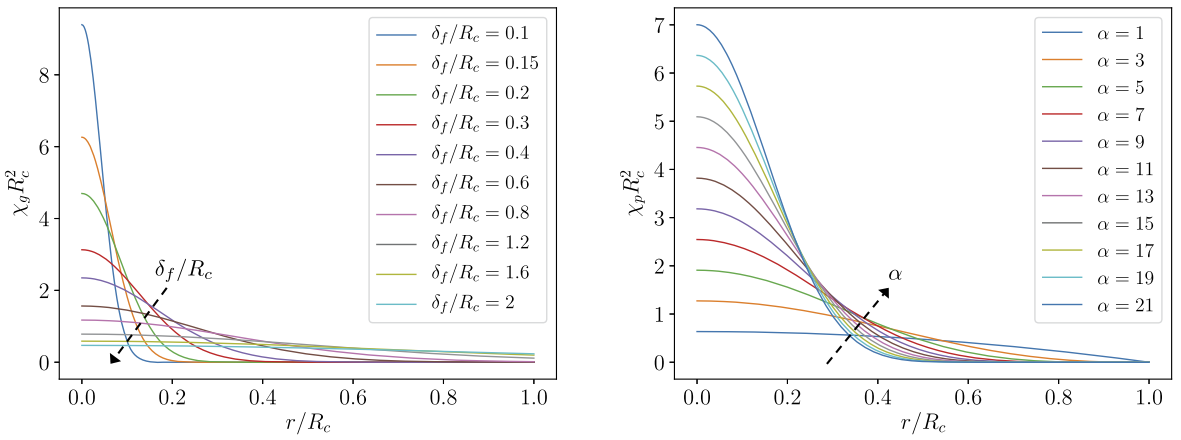
$$\chi_p(\boldsymbol{x};\boldsymbol{x}_d) = \begin{cases} \dfrac{\alpha+1}{\pi R_c^2}\left[1-\left(\dfrac{r}{R_c}\right)^2\right]^\alpha, & r \le R_c, \\ 0, & r > R_c, \end{cases} \tag{15}$$

where α is an integer that controls the order of the polynomial and the distribution of the kernel. The characteristic width is then $R_c\sqrt{1-(1/2)^{1/\alpha}}$. At the cutoff distance, R_c, the kernel is exactly zero. Therefore, the integral of the kernel over its extent is exactly unity, i.e. $\int_\Omega \chi_p d\Omega = \int_0^{2\pi}\int_0^{R_c} \chi_p r dr d\theta = 1$. Fig. 2(b) displays the profiles for this kernel at different values of α. Higher α results in greater kernel density near its center, whereas lower α more evenly distributes the kernel over its extent.

One advantage of the polynomial kernel is that if its influence covers an entire element, its integral over the element can be integrated exactly. To demonstrate this, the Gaussian and polynomial kernels are integrated in 1D over an element, Ω_e, of size $\Delta x = 1$. To maintain compactness, we set R_c to $\Delta x/2$. The kernels are centered at the element centroid. The projection error, defined as $\left|\int_\Omega \chi d\Omega - 1\right|$, is displayed in Fig. 3. Given that the kernels do not extend into other elements, $\int_\Omega \chi d\Omega = \int_{\Omega_e} \chi d\Omega$. The integral is approximated using a Qth-order Gauss-Legendre quadrature rule (which can exactly integrate a polynomial of order less than or equal to Q) as

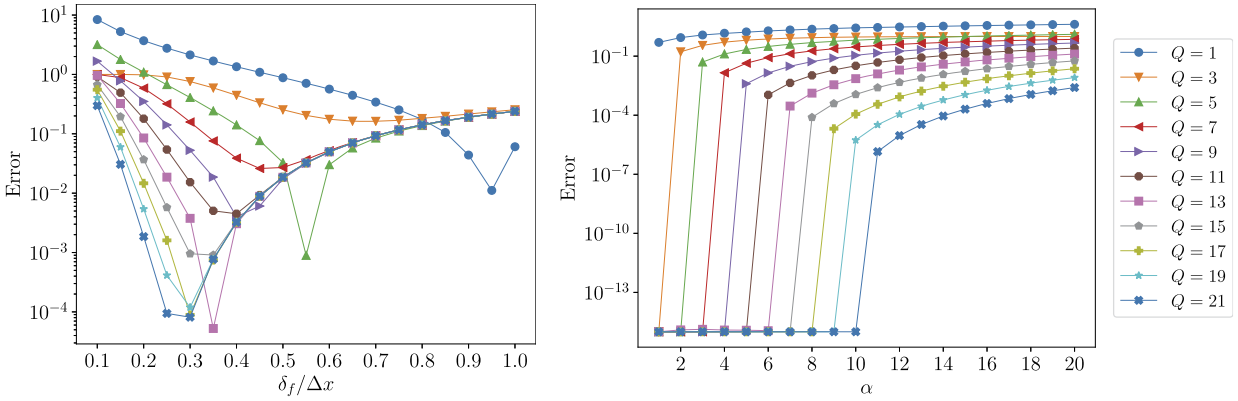
$$\int_{\Omega_e} \chi d\Omega \approx \sum_{k=1}^{N_Q} w_k \chi(x_k) j_e(\xi_k), \tag{16}$$

where N_Q is the number of quadrature points, w_k is the kth quadrature weight, x_k and ξ_k are the coordinates of the kth quadrature point in physical and reference space, respectively, and j_e is the Jacobian determinant of the geometric mapping of Ω_e. Gauss-Legendre quadrature rules of different Q are employed. For adequate Q, the polynomial kernel can be integrated with machine zero error, unlike the Gaussian kernel. Integrating the Gaussian kernel with low Q can result in large error.



(a) Gaussian profiles (b) Polynomial profiles

Fig. 2. Profiles of the (a) Gaussian kernel defined in Eq. (14) for different values of δ_f/R_c and (b) polynomial kernel defined in Eq. (15) for different values of α.



(a) Gaussian kernel (b) Polynomial kernel

Fig. 3. Projection errors in 1D of the (a) Gaussian kernel defined in Eq. (14) for different values of $\delta_f/\Delta x$ and (b) polynomial kernel defined in Eq. (15) for different values of α. The kernel boundary coincides with element boundary.

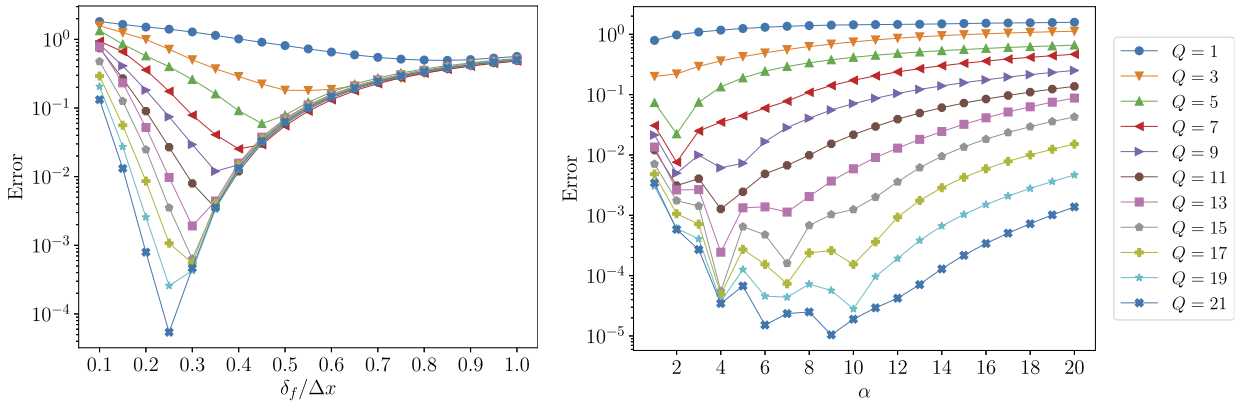
In general, though, it is rare for the kernel boundary to coincide exactly with the boundary of an element, especially in 2D and 3D. If a kernel fully covers an entire element, then the integral of the polynomial kernel over said element can be integrated exactly with numerical quadrature. However, the kernel will likely spread into neighboring elements as well. The integration of the polynomial kernel in those elements will not be exact due to its piecewise nature. To further evaluate the effectiveness of the two projection kernels, we consider a mesh with 3×3 square elements, each with side length $\Delta x = 1$. 121 particles are uniformly placed at equidistant locations within the center element. The projection error is now evaluated as

$$\frac{1}{N_p}\sum_{i=1}^{N_p}\left|\int_\Omega \chi_i d\Omega - 1\right|, \tag{17}$$

with $N_p = 121$, and the domain integral is computed as

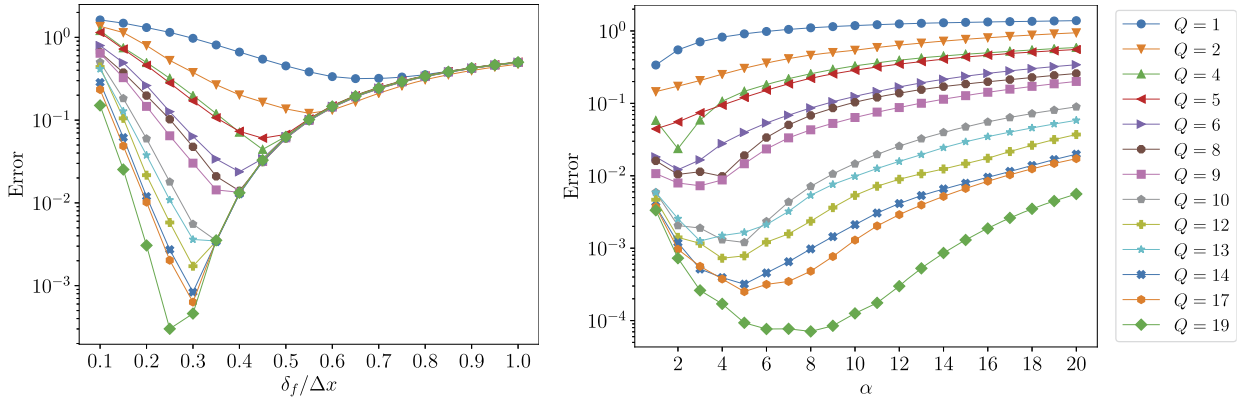
$$\int_\Omega \chi_i d\Omega = \sum_{e=1}^{N_v}\int_{\Omega_e} \chi_i d\Omega, \tag{18}$$

where N_v is the number of elements in the neighborhood of the particle consisting of the host element and the node-sharing elements. Each of these element integrals is numerically evaluated using Gauss-Legendre quadrature. For compactness, the cutoff distance is again set to $R_c = \Delta x/2$. The errors are shown in Fig. 4.



(a) Gaussian kernel (b) Polynomial kernel

Fig. 4. Projection errors in 2D of the (a) Gaussian kernel defined in Eq. (14) for different values of $\delta_f/\Delta x$ and (b) polynomial kernel defined in Eq. (15) for different values of α. 121 particles are placed at equidistant locations within the center element of a mesh with 3×3 square elements.

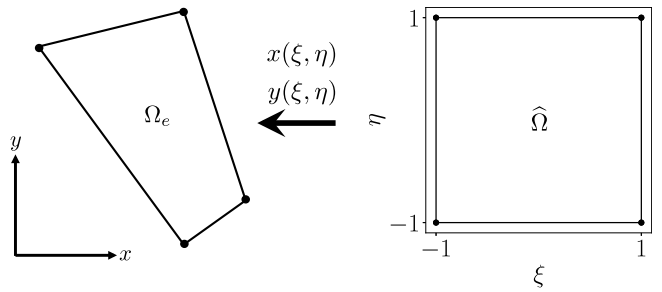


(a) Gaussian kernel (b) Polynomial kernel

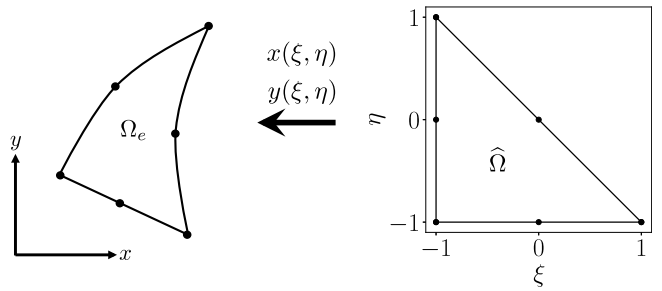
Fig. 5. Projection errors in 2D of the (a) Gaussian kernel defined in Eq. (14) for different values of $\delta_f/\Delta x$ and (b) polynomial kernel defined in Eq. (15) for different values of α. Each square element of the previous mesh (Fig. 4) is split into two right triangles of unit side length. Only the quadrature rules up to $Q = 19$ whose points are inside the triangle and whose weights are positive are employed.

Overall, the projection errors associated with the Gaussian kernel are similar to those in the previous 1D test (Fig. 3(a)). Greatest accuracy is achieved with high Q and small (but not exceedingly small) kernel widths. For low Q, intermediate kernel widths ($\delta_f/\Delta x \approx 0.5$) give the lowest errors. Regardless of Q, the Gaussian kernel cannot be integrated accurately for extremely small and large values of $\delta_f/\Delta x$. In the former case, the kernel approaches a delta function, while in the latter, the weight of the kernel is distributed more uniformly, resulting in large cutoff errors. For the polynomial kernel, the errors are much larger than in the 1D test (Fig. 3(b)) since the integrals can no longer be evaluated analytically using Gauss-Legendre quadrature. Nevertheless, for fixed Q, the errors are generally lower for the polynomial kernel than for the Gaussian kernel. In general, as Q increases, the optimal value of α increases. For instance, $\alpha = 2$ and $\alpha = 10$ yield the lowest error for $Q = 5$ and $Q = 19$, respectively. Over all considered values of Q and α, greatest accuracy is obtained with $Q = 21$ and $\alpha = 9$. However, if α is excessively large, then the kernel approaches a delta function, causing high projection errors. We note that although the precise details and quantities observed here will differ if more complex configurations are considered, the qualitative results remain overall consistent.

The above calculation is repeated on triangular elements using the quadrature rules proposed by Dunavant [23]. Each square element is divided into two right triangles of unit side length ($\Delta x = 1$). Only the quadrature rules up to $Q = 19$ whose points are inside the triangle and whose weights are positive are employed. The particles are placed at the same 121 equidistant locations, with $R_c = \Delta x/2$. The results are displayed in Fig. 5. The trends here are broadly the same as on the quadrilateral elements (with Gauss-Legendre quadrature). However, one unexpected result is that $Q = 12$ mostly gives lower errors than $Q = 13$ for both kernels. For the polynomial kernel, the differences with the quadrilateral elements are more noticeable. For a given Q, the error vs. α profile is smoother than on the quadrilateral elements, and the optimal value of α is different.



(a) Geometric mapping for $q = 1$ quadrilateral elements



(b) Geometric mapping for $q = 2$ triangular elements

Fig. 6. Geometric mapping for (a) $q = 1$ quadrilateral elements and (b) $q = 2$ triangular elements.

Due to the greater accuracy and zero cutoff error, we elect to employ the polynomial shape function, χ_p, instead of the Gaussian function. The circular/spherical region of influence of the polynomial function will be projected to an ellipse/ellipsoid to build an appropriate anisotropic kernel. This will be discussed in Section 3.5. We remark that the results in this section can be used as guidelines for selecting α and Q according to user preferences regarding accuracy, computational cost, and stability. Note that lower α can more effectively mitigate noise and instabilities since the kernel is more uniformly distributed over its extent. It is also certainly possible for this study, i.e. computing projection error as a function of α and Q, to be repeated for other meshes and flow configurations of interest, but this is not considered essential.

3.4. Geometric mapping

Before we detail how to construct a suitable anisotropic projection kernel, we briefly review the 2D geometric mapping from reference space to physical space. The mapping $\boldsymbol{x} : \widehat{\Omega} \rightarrow \Omega_e$, with $\widehat{\Omega}$ denoting the reference element, is given as

$$x(\xi,\eta) = \sum_{i=1}^{N_n} \psi_i(\xi,\eta) x_i^e, \tag{19a}$$

$$y(\xi,\eta) = \sum_{i=1}^{N_n} \psi_i(\xi,\eta) y_i^e, \tag{19b}$$

where ξ and η are the reference space coordinates, N_n is the number of nodes defining the element geometry, ψ_i is the ith basis function for the geometry interpolation, and (x_i^e, y_i^e) is the physical position of the ith node. Fig. 6 shows the geometric mapping for a $q = 1$ quadrilateral and a $q = 2$ triangle, where q refers to the order of the geometry interpolation. The reference element for quadrilaterals is a square with a side length of two, and that for triangles is an isosceles right triangle with a leg length of two. The geometric Jacobian is given by

$$\boldsymbol{J}_e = \begin{bmatrix} \dfrac{\partial x(\xi,\eta)}{\partial \xi} & \dfrac{\partial x(\xi,\eta)}{\partial \eta} \\ \dfrac{\partial y(\xi,\eta)}{\partial \xi} & \dfrac{\partial y(\xi,\eta)}{\partial \eta} \end{bmatrix}. \tag{20}$$

3.5. Finite-size, anisotropic kernel

As detailed in Section 1, using an isotropic kernel on very high-aspect-ratio elements can be problematic. An anisotropic kernel can mitigate many of these issues while maintaining a good balance among accuracy, stability, and computational efficiency. To illustrate the construction of a suitable anisotropic shape function, we again consider the 2D case, although a brief discussion of the 3D extension will be provided. Note that in 1D, there is no distinction between isotropy and anisotropy under the considered framework.

3.5.1. 2D kernel

Instead of using a circle to represent the kernel shape, we consider an ellipse. To construct an appropriate ellipse, information about the size and orientation of a given element is required. This can be obtained from the mesh-implied metric, a key ingredient of recent mesh adaptation strategies [16,17]. Specifically, consider the following 2×2 symmetric positive definite matrix:

$$\boldsymbol{\mathcal{M}} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T = \begin{bmatrix} | & | \\ \widehat{\boldsymbol{e}}_1 & \widehat{\boldsymbol{e}}_2 \\ | & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} — & \widehat{\boldsymbol{e}}_1^T & — \\ — & \widehat{\boldsymbol{e}}_2^T & — \end{bmatrix}. \tag{21}$$

The columns of $\boldsymbol{V}$ are the orthonormal eigenvectors, $\widehat{\boldsymbol{e}}_1$ and $\widehat{\boldsymbol{e}}_2$, and $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues, λ_1 and λ_2, along the main diagonal. Following Refs. [16] and [17], the eigenvectors are the left singular vectors of $\boldsymbol{J}_e$. The eigenvalues are related to the singular values, h_1 and h_2, of $\boldsymbol{J}_e$ as

$$\lambda_1 = \frac{1}{h_1^2}, \quad \lambda_2 = \frac{1}{h_2^2}. \tag{22}$$

$\boldsymbol{\mathcal{M}}$ can then be used to define an ellipse in physical space centered at $\boldsymbol{x}_0$ using the following implicit equation:

$$(\boldsymbol{x}_l - \boldsymbol{x}_0)^T \boldsymbol{\mathcal{M}} (\boldsymbol{x}_l - \boldsymbol{x}_0) = 1, \tag{23}$$

which can be solved for $\boldsymbol{x}_l$ to obtain the points defining the surface of the ellipse. $\widehat{\boldsymbol{e}}_1$ and $\widehat{\boldsymbol{e}}_2$ in Eq. (21) define the principal directions of the ellipse; h_1 and h_2 are the principal stretching magnitudes, or the lengths of the semi-axes.

An alternative way to compute $\boldsymbol{x}_l$ is to first construct the following matrix:

$$\boldsymbol{\mathcal{H}} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{bmatrix} | & | \\ \widehat{\boldsymbol{e}}_1 & \widehat{\boldsymbol{e}}_2 \\ | & | \end{bmatrix} \begin{bmatrix} h_1 & 0 \\ 0 & h_2 \end{bmatrix} \begin{bmatrix} — & \widehat{\boldsymbol{e}}_1^T & — \\ — & \widehat{\boldsymbol{e}}_2^T & — \end{bmatrix}, \tag{24}$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix with h_i along the main diagonal. $\boldsymbol{x}_l$ can then be obtained as

$$\boldsymbol{x}_l = \boldsymbol{\mathcal{H}}\widehat{\boldsymbol{t}} + \boldsymbol{x}_0, \tag{25}$$

where $\widehat{\boldsymbol{t}}$ is a unit vector. Eq. (25) projects a unit circle to the given ellipse by means of three consecutive geometrical transformations: a rotation via $\boldsymbol{V}^T$, a horizontal and vertical scaling through $\boldsymbol{\Sigma}$, and a rotation in the opposite direction by $\boldsymbol{V}$. This is shown schematically in Fig. 7. Note that in the context of triangular mesh adaptation, Fidkowski [16] and Oliver [17] introduce an additional mapping between the reference element and a unit equilateral triangle, which guarantees an isotropic metric for equilateral triangular elements. This is not deemed necessary for our desired application and is not pursued here.

The last step in the construction of the anisotropic polynomial-based kernel is to modify χ_p to ensure that Eq. (11) is satisfied. Specifically, the area of the ellipse, $\pi h_1 h_2$, is accounted for, instead of, as in the original isotropic version (Eq. (15)), the area of the circle. The final form of the kernel is expressed mathematically as

$$\chi_p(\boldsymbol{x}; \boldsymbol{x}_d) = \begin{cases} \dfrac{\alpha+1}{\pi h_1 h_2}\left(1 - s^2\right)^{\alpha}, & s \leq 1, \\ 0, & s > 1, \end{cases} \tag{26}$$

where $s = |\boldsymbol{s}|$, with

$$\boldsymbol{s} = \boldsymbol{\mathcal{H}}^{-1}(\boldsymbol{x}_d)\boldsymbol{r}. \tag{27}$$

$s < 1$, $s = 1$, and $s > 1$ correspond to the ellipse interior, boundary, and exterior, respectively. Note that $\boldsymbol{\mathcal{H}}^{-1}$ is evaluated at $\boldsymbol{x}_d$ (the location of the given particle) on the host element.

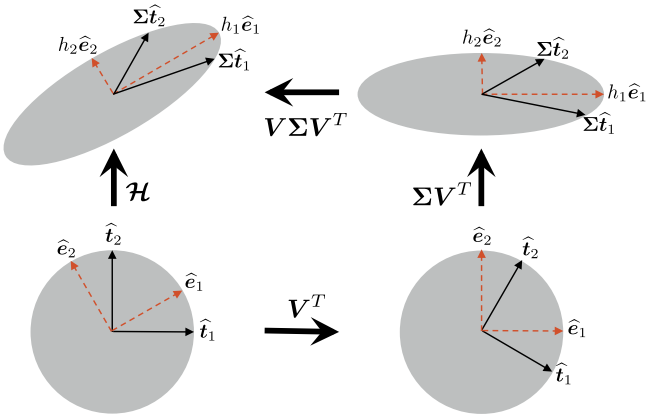


Fig. 7. Geometrical interpretation of the linear transformation $\hat{\boldsymbol{t}} \rightarrow \mathcal{H}\hat{\boldsymbol{t}}$ (see Eqs. (24) and (25)). The unit circle (bottom left) is first rotated via $\boldsymbol{V}^T$ (bottom right), then scaled horizontally and vertically through $\boldsymbol{\Sigma}$ (top right), and finally rotated in the opposite direction by $\boldsymbol{V}$ (top left).

3.5.2. Remarks

With the ellipse defined as in Eqs. (24) and (25), the kernel will generally reach no further than the node-sharing elements. The stencil of the DG method is therefore slightly expanded from solely the face-sharing elements. In certain cases, a different kernel size or aspect ratio may be desired. This can be easily attained by modifying h_1 and/or h_2 in Eq. (24). For example, if a uniform mesh is used, then h_1 and h_2 can be doubled without further expanding the stencil. The recommended prescription of h_1 and h_2 in the previous section represents a good default choice, especially for highly stretched and/or unstructured meshes in which it is not clear beforehand the number of elements that overlap with an ellipse of arbitrary size in any given part of the domain. An isotropic kernel is recovered if $h_1 = h_2$. Note that in Refs. [16,17], the reference triangle was an isosceles right triangle with unit leg length. In this case, the ellipse construction procedure described above would generate an ellipse twice the size as that obtained with the reference triangle in Fig. 6(b). Therefore, the recommended ellipse size can be obtained by simply halving h_1 and h_2. This similarly applies if the reference quadrilateral is a unit square, instead of the bi-unit square in Fig. 6(a). We also note that provided the mesh is valid, i.e. the determinant of the geometric Jacobian is positive (even if the mesh quality is not considered high according to classical mesh quality measures based on the geometric Jacobian [24]), an appropriate kernel can be constructed.

For meshes with rapidly varying element sizes and orientations, the kernel may extend past the node-sharing elements. Furthermore, $\mathcal{H} = \bigoplus_{e=1}^{N_e} \mathcal{H}^e$ is discontinuous across elements and, inside a given element, dependent only on the geometric mapping of said element. Therefore, ellipse size and orientation can change abruptly across elements, which may lead to inaccuracies and noise. To mitigate these issues, we employ an averaged, globally continuous representation of $\mathcal{H}$ using a first-order Lagrange basis. With $\mathcal{H}_s^e$ denoting the local representation on Ω_e, the nth set of coefficients is computed via a simple node-averaging procedure as

$$\widetilde{\mathcal{H}}_n^e = \mathcal{H}_s^e(\boldsymbol{x}_n^e) = \frac{1}{N_{v,n}^e} \sum_{l=1}^{N_{v,n}^e} \mathcal{H}^l(\boldsymbol{x}_n^e), \tag{28}$$

where $N_{v,n}^e$ is the number of elements connected to the nth node of Ω_e and $\mathcal{H}^l$ is the local distribution of $\mathcal{H}$ in the lth of those elements. $\mathcal{H}$ in Eq. (27) is then replaced by $\mathcal{H}_s$. More sophisticated averaging procedures can be employed instead [25,26], but that in Eq. (28) is sufficient for our purposes. Note that even with the averaged representation of $\mathcal{H}$, it is still possible for the kernel to extend past the node-sharing elements. However, this scenario can be addressed using the same approach proposed for treating walls, which will be introduced later in this section.

To minimize the mesh dependence of the kernel shapes (and potentially further suppress numerical instabilities), an extra diffusion operation can be applied [1]. This involves solving the following diffusion equation for $\boldsymbol{S}$:

$$\frac{\partial \boldsymbol{S}}{\partial \mathrm{t}} = \mathrm{D}\nabla^2 \boldsymbol{S}, \tag{29}$$

where t is a fictitious time, D is a diffusion coefficient, and the components of $\boldsymbol{S}$ at $\mathrm{t} = 0$ are defined as in Eqs. (10a). $\sqrt{\mathrm{D}\mathrm{t}_f}$, where t_f is the final time, represents a diffusion length scale.

We draw from Ref. [1] to provide approximate guidelines for choosing between delta functions and smooth kernels, as well as whether to apply the above diffusion step. If the maximum particle radius is less than $h_{\min}/10$, where $h_{\min} = \min\{h_1, h_2\}$, then delta functions are likely to be sufficient since the influence of each particle is generally small relative to the element. Otherwise, the use of smooth kernels is suggested. We find this to be particularly important when the local

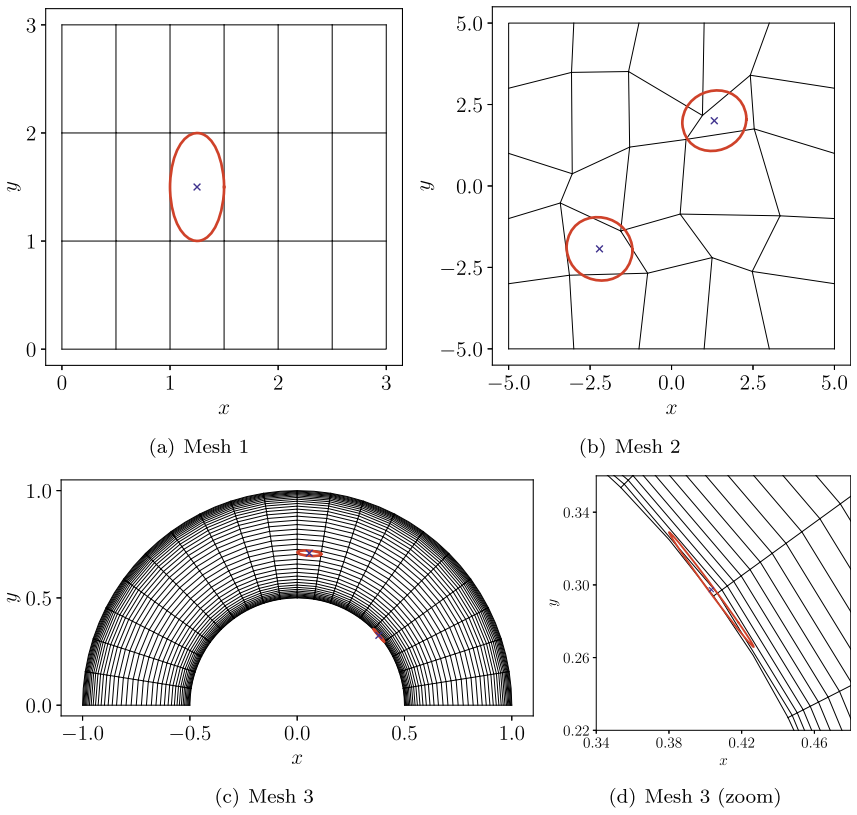


Fig. 8. Example ellipses on three different quadrilateral meshes: (a) Mesh 1 is a structured $q = 1$ mesh; (b) Mesh 2 is an unstructured $q = 1$ mesh; (c) Mesh 3 is a $q = 2$ mesh with high-aspect-ratio elements. In (d), we zoom in on the high-aspect-ratio elements near the wall for Mesh 3. Blue crosses indicate the ellipse center, and the ellipse boundary is in red.

volume fraction exceeds approximately 1%, in which case noise and numerical instabilities are more likely to occur. If the maximum particle radius is furthermore greater than h_{min}, then we propose applying the diffusion process. Guidelines for prescribing the diffusion length scale as a function of the characteristic kernel width, mesh size, and particle diameter can be found in Ref. [1]. For the test cases considered in this study, since the particle radii do not exceed h_{min}, the diffusion operation is not applied. Nevertheless, this step can be crucial for flows in which particles are large relative to the elements of the mesh. In such cases, volume filtering of the Navier-Stokes equations may be important as well [1,27].

Finally, computational cost can be reduced by means of an adaptive strategy to switch between delta shape functions and smooth shape functions. For instance, the former can be used in regions where the back-coupling is not important, while the latter can be employed in areas with strong back-coupling. Additionally, for meshes with a wide range of element sizes, delta functions can be applied on large elements, with smooth kernels engaged on the smaller elements. In this paper, however, only one type of shape function is employed in a given simulation.

3.5.3. 2D example kernels

We proceed to show example ellipses constructed using the above procedures on different types of meshes. Fig. 8 gives results on three quadrilateral meshes: a structured $q = 1$ mesh, an unstructured $q = 1$ mesh, and a $q = 2$ mesh with high-aspect-ratio elements near the annular boundaries. For this last mesh, a zoomed-in view of a region near the inner wall is provided. In Fig. 8(a), the ellipse perfectly captures the size and orientation of the host element. In Fig. 8(b), the elements are more isotropic, resulting in more isotropic ellipses as well. The ellipses in Figs. 8(c) and 8(d) accurately account for the size and orientation of the high-order, high-aspect-ratio elements. In all cases, the ellipse spreads over only the host element and the node-sharing elements, ensuring a compact stencil.

This exercise is repeated on triangular elements. Each element of the three quadrilateral meshes is split into two triangles. The resulting meshes and ellipses are displayed in Fig. 9. In general, the ellipses here are slightly longer and more anisotropic than the corresponding ellipses in Fig. 8. Nevertheless, the size and orientation of the elements are well-captured. Note that in Fig. 9(b), the upper-right ellipse does extend past the node-sharing elements due to the abrupt variation in element size. This also occurs in Fig. 9(d), a consequence of the high-aspect-ratio elements and curved annular boundary. As previously mentioned, this scenario will be addressed in Section 3.6.

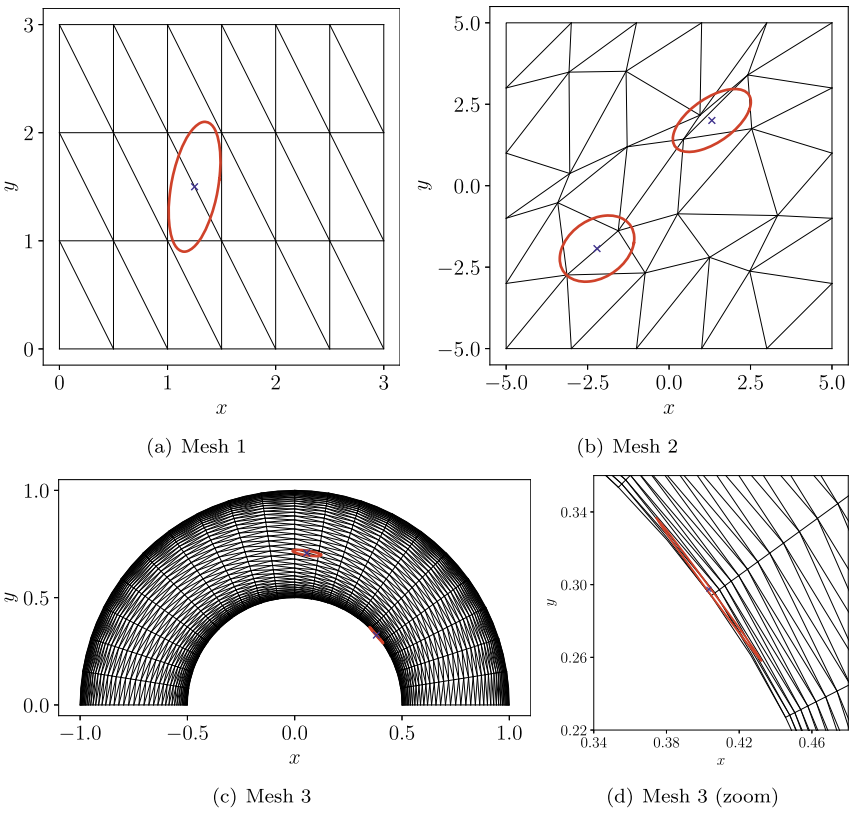


Fig. 9. Example ellipses on three different triangular meshes: (a) Mesh 1 is a structured $q = 1$ mesh; (b) Mesh 2 is an unstructured $q = 1$ mesh; (c) Mesh 3 is a $q = 2$ mesh with high-aspect-ratio elements. In (d), we zoom in on the high-aspect-ratio elements near the wall for Mesh 3. Blue crosses indicate the center of the ellipse, and the ellipse boundary is drawn in red. These meshes are created by splitting the quadrilateral elements of the meshes in Fig. 8 into triangles.

3.5.4. 3D kernel

In this section, the extension of the anisotropic polynomial kernel to 3D is discussed. The kernel shape changes from an ellipse to an ellipsoid, with $\mathcal{H}$ now given as

$$\mathcal{H} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{bmatrix} | & | & | \\ \widehat{\boldsymbol{e}}_1 & \widehat{\boldsymbol{e}}_2 & \widehat{\boldsymbol{e}}_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} h_1 & 0 & 0 \\ 0 & h_2 & 0 \\ 0 & 0 & h_3 \end{bmatrix} \begin{bmatrix} — & \widehat{\boldsymbol{e}}_1^T & — \\ — & \widehat{\boldsymbol{e}}_2^T & — \\ — & \widehat{\boldsymbol{e}}_3^T & — \end{bmatrix}, \tag{30}$$

where h_1, h_2, and h_3 are the singular values of $\boldsymbol{J}_e$ and $\widehat{\boldsymbol{e}}_1$, $\widehat{\boldsymbol{e}}_2$, and $\widehat{\boldsymbol{e}}_3$ are the left singular vectors of $\boldsymbol{J}_e$. With the appropriate normalization (Eq. (11)), the final form of the kernel is defined as

$$\chi_p(\boldsymbol{x}; \boldsymbol{x}_d) = \begin{cases} \dfrac{\Gamma\left(\alpha + \frac{5}{2}\right)}{\pi^{3/2}\Gamma(\alpha+1)h_1h_2h_3}\left(1 - s^2\right)^{\alpha}, & s \leq 1, \\ 0, & s > 1, \end{cases} \tag{31}$$

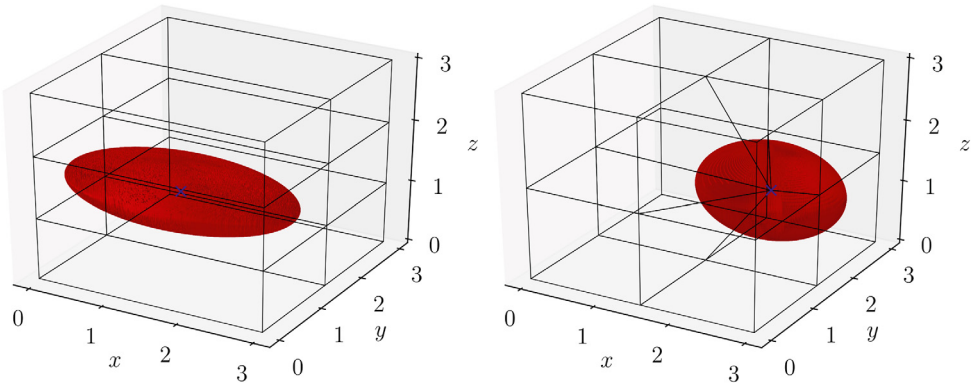
where Γ is the gamma function, written as

$$\Gamma(z) = \int_0^{\infty} \zeta^{z-1} e^{-\zeta} d\zeta. \tag{32}$$

Many standard math libraries provide built-in routines to evaluate the gamma function. Note that Stock et al. [14] and Pfeiffer et al. [15] employed the isotropic version of this polynomial-based kernel in plasma dynamics simulations.

3.5.5. 3D example kernels

Just as in Section 3.5.3, we show sample ellipsoids constructed using the proposed approach on two hexahedral meshes. The computational domain is $[0, 3] \times [0, 3] \times [0, 3]$ for both meshes. The first is a structured mesh with $1 \times 2 \times 3$ elements;



(a) Mesh 1 (b) Mesh 2

Fig. 10. Example ellipsoids on two hexahedral meshes: (a) Mesh 1 is a structured mesh with $1 \times 2 \times 3$ elements; (b) Mesh 2 consists of $2 \times 2 \times 2$ elements, with the center node perturbed. Blue crosses indicate the center of the ellipse, and the ellipse boundary is drawn in red.

the ellipsoid center is located at $\boldsymbol{x}_0 = [1.5, 0.75, 1.5]^T$. The second mesh consists of $2 \times 2 \times 2$ elements, with the center node perturbed; the ellipsoid is centered at $\boldsymbol{x}_0 = [2.1, 2, 1.1]^T$. In Fig. 10(a), the ellipsoid perfectly captures the size and shape of the host element. In Fig. 10(b), the ellipsoid is suitably constructed in spite of the apparent perturbation. These results, along with those in Section 3.5.3, demonstrate that the mesh-implied metric can accurately encode the size and orientation of many types of elements, even those that are highly anisotropic, distorted, and/or curved.

3.5.6. Parallelization

This section explains how parallel simulations are handled in the context of a standard Message Passing Interface (MPI) implementation. Each processor stores in memory the particles whose centroids are located inside the elements comprising its subdomain. Since the projection kernel of a particle near a subdomain boundary can cross into other subdomains, we make use of *ghost particles* [1], which are copies of physical particles sent to neighboring processors. That the kernel generally does not extend past the node-sharing elements maintains a reasonable implementation and interprocessor communication overhead. If a particle is located in an element that shares a node with an element stored on another processor, then it is communicated to the processor as a ghost particle belonging to said element. To illustrate this, consider the example subdomain partitioning in Fig. 11, where the elements are colored by host processor. Particles in Element 1 on Proc 1 are communicated to Element 7, its face-sharing neighbor, on Proc 2 as ghost particles. This is also the case for particles in Element 2 since it shares a node with Element 7. Particles in Element 10, on the other hand, are copied to both of the other processors: on Proc 2, Elements 15, 16, and 17 are looped through to evaluate the projection kernel; on Proc 3, the kernel is evaluated in Elements 4, 5, and 11.

An alternative parallelization approach is to circumvent ghost particles and store on each processor the necessary geometry and basis information of all elements *outside* its subdomain that share a node with elements *inside* its subdomain. For instance, Proc 1 in Fig. 11 stores geometry and basis information of Elements 7, 13, 14, 15, 16, and 17 from Proc 2 and Elements 4, 5, and 11 from Proc 3. For illustration purposes, we focus on the particles located in Element 8. The source term integrals (the last term on the RHS of Eq. (5)) associated with these particles are computed for Element 8 and the node-sharing elements. These integral evaluations are all performed on Proc 1. The values of the integrals over the elements outside Proc 1's subdomain are then communicated to the corresponding processors. This approach is potentially simpler, especially since many DG codes already store this information for the face-sharing elements (a subset of the node-sharing elements) outside a given subdomain. However, Euler-Lagrange solvers that can handle interparticle collisions likely already rely on ghost particles, which would make the first approach more natural.

In this study, the domain is partitioned such that all processors have approximately the same number of elements. However, this can lead to load imbalance, especially if particles are highly concentrated in specific regions of the domain. Since none of the simulations here are considerably large, we do not take explicit measures to address such load imbalance. Nevertheless, future work will entail implementation of dynamic load balancing strategies [28] to enable larger-scale simulations.

3.6. Near-wall treatment

As discussed in Section 1, ensuring conservation of interphase momentum and energy transfer near walls can be challenging on arbitrary curved elements. The classical strategy of creating a mirror particle [1,3] is significantly less straightforward than on straight-sided elements. A general, brute-force approach is to simply rescale the projection kernel such that its integral is numerically evaluated to be unity, i.e.

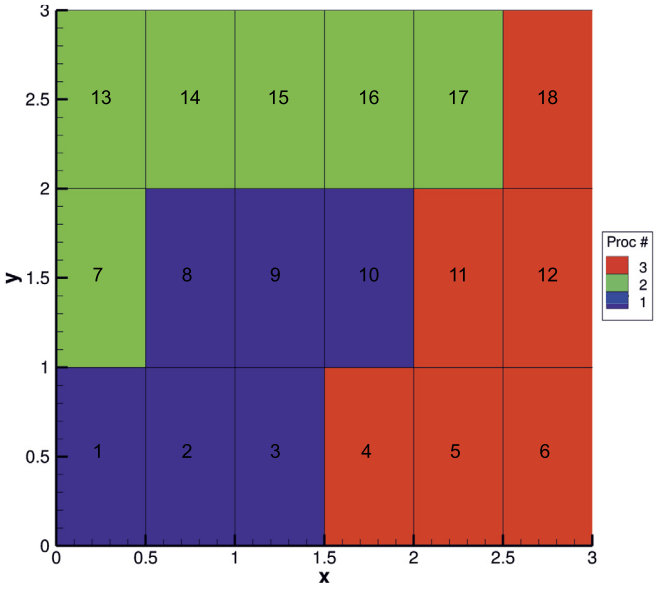


Fig. 11. Example subdomain partitioning to illustrate ghost particles. Elements are indicated by their IDs and colored by host processor.

$$K(\boldsymbol{x}_d) = \frac{1}{\sum_{e=1}^{N_v} \int_{\Omega_e} \chi_p(\boldsymbol{x}; \boldsymbol{x}_d) d\Omega}, \tag{33}$$

where K is the scaling factor. However, this can be very expensive since K must be calculated for each particle at every time step. Furthermore, depending on the parallelization strategy, this can result in excessive interprocessor communication.

Nevertheless, the generality of the rescaling approach is desirable. To reduce computational overhead, instead of calculating a new scaling factor for each particle at every time step, we employ a polynomial representation of K, with the local element approximation given as

$$K_h^e(\boldsymbol{x}_d) = \sum_{n=1}^{N_K} \widetilde{K}_n^e \Phi_n(\boldsymbol{x}_d), \tag{34}$$

where Φ_n is the nth basis function, N_K is the number of basis functions, and $\widetilde{K}_n^e$ is the nth coefficient. K_h^e is obtained at the beginning of the simulation by means of L_2 projection, given in vector form as

$$\widetilde{\boldsymbol{K}}^e = \mathbf{M}^{-1} \int_{\Omega_e} K(\boldsymbol{x}) \boldsymbol{\Phi}(\boldsymbol{x}) d\Omega, \tag{35}$$

where $\mathbf{M}$ is the mass matrix, with $\mathsf{M}_{mn} = \int_{\Omega_e} \Phi_m \Phi_n d\Omega$. The integral on the RHS of Eq. (35) is calculated using high-order numerical quadrature, which involves computing K as in Eq. (33) at the quadrature points. An alternative way to obtain K_h^e, assuming a Lagrange basis, is via interpolation to the nodes as

$$K_h^e(\boldsymbol{x}_n) = \widetilde{K}_n^e = K(\boldsymbol{x}_n). \tag{36}$$

Although interpolation is cheaper than L_2 projection, we recommend using the latter to initialize K_h^e, especially since this only needs to be performed once as a preprocessing step.

With K_h^e, the integral of the reverse coupling source term over Ω_e (the last term on the RHS of Eq. (5)) for the ith particle is evaluated as

$$\int_{\Omega_e} \phi_m \boldsymbol{S}_i(\boldsymbol{x}; \boldsymbol{x}_{d,i}) d\Omega = [0, \boldsymbol{F}_i, Q_i + \boldsymbol{u}_{d,i} \cdot \boldsymbol{F}_i]^T K_h^o(\boldsymbol{x}_{d,i}) \int_{\Omega_e} \chi_p(\boldsymbol{x}; \boldsymbol{x}_{d,i}) \phi_m(\boldsymbol{x}) d\Omega, \tag{37}$$

where $(\cdot)^o$ denotes the host element of the particle, which is not necessarily Ω_e. Note that although K_h varies in space, it is computed at the particle centroid in the host element and can be moved outside of the integral. Recall that in evaluating χ_p, the ellipse is similarly obtained by computing $\boldsymbol{\mathcal{H}}_s$ at the particle centroid in the host element.

3.6.1. Remarks

Given that the rescaling factor is not exact under this approach, we propose employing very high-order polynomials, e.g., sixth-order or higher, for K_h in order to maintain a high level of accuracy. If desired, memory requirements and initialization cost can be reduced by storing and accounting for K_h only in certain regions, such as near walls. In elements where rescaling is not necessary, K_h can be excluded from Eq. (37) and therefore need not be stored.

We note that this rescaling approach further motivates the use of the polynomial kernel, instead of the Gaussian function. Given its nonzero cutoff error, rescaling the Gaussian kernel would result in even greater cutoff error (for $K_h > 1$). On the other hand, for the polynomial kernel, there is zero cutoff error regardless of the magnitude of the rescaling factor.

Finally, a secondary advantage of the developed strategy is that it implicitly addresses the scenarios shown in Figs. 9(b) and 9(d), in which the kernel extends past the node-sharing elements. Accounting for K_h enables looping over only the host element and node-sharing elements while maintaining low projection error. Specific examples will be illustrated in Section 3.6.3. However, it is still preferred, for the purposes of numerical stability and low projection error, to minimize the likelihood of the kernel spreading past the node-sharing elements. If large kernels that extend past this neighborhood are consistently used, but only the elements in said neighborhood are looped over, this can lead to a less accurate polynomial representation of the rescaling factor (K_h) and therefore higher projection error. Furthermore, this will force abrupt truncation of the kernel at a nonzero value, potentially causing Gibbs-type noise. Nevertheless, with the construction of the ellipse as in Sec 3.5.1, this scenario is typically infrequent, and the proposed rescaling approach represents an appropriate remedy.

3.6.2. Parallelization

To aid with implementation, we outline how to construct K_h in parallel simulations based on a standard MPI implementation. For illustration purposes, we focus on L_2 projection, but interpolation can be done in a similar manner. Recall that this is performed at the start of the simulation. The details are given in Algorithm 1 from the perspective of a single processor. Note that Algorithm 1 draws from the ghost-particle approach, which many Euler-Lagrange solvers already rely on, but, as discussed in Section 3.5.6, parallelization can alternatively be done by storing on each processor additional information about node-sharing elements outside its subdomain. Though not pursued here, this alternative approach can simplify some steps in Algorithm 1.

Algorithm 1: Procedure to construct K_h in parallel simulations based on a standard MPI implementation, from the perspective of Processor P. We focus on L_2 projection, but interpolation can be done in a similar manner. This procedure is executed at the start of the simulation.

for *each element e* **do**
  for *each quadrature point k on element e, located at* $\boldsymbol{x}_k^e$ **do**
    $I_k^e \leftarrow \int_{\Omega_e} \chi_p(\boldsymbol{x};\boldsymbol{x}_k^e) d\Omega$
    for *each node-sharing element l on Processor* P **do**
      $I_k^e \leftarrow I_k^e + \int_{\Omega_l} \chi_p(\boldsymbol{x};\boldsymbol{x}_k^e) d\Omega$
    end
  end
  if *element e shares a node with an element on another processor* **then**
    Send $\boldsymbol{x}_k^e$, $\mathcal{H}_s^{-1,e}(\boldsymbol{x}_k^e)$, and eigenvalues of $\mathcal{H}_s^e(\boldsymbol{x}_k^e)$ to node-sharing processors (necessary for computing χ_p)
  end
end
for *each* $\boldsymbol{x}_k^e$ *received by Processor* P **do**
  for *each element l on Processor* P *sharing a node with element e (on a different processor)* **do**
    Compute $\int_{\Omega_l} \chi_p(\boldsymbol{x};\boldsymbol{x}_k^e) d\Omega$ and send to processor storing element e
  end
end
for *each* $\int_{\Omega_l} \chi_p(\boldsymbol{x};\boldsymbol{x}_k^e) d\Omega$ *received by Processor* P **do**
  $I_k^e \leftarrow I_k^e + \int_{\Omega_l} \chi_p(\boldsymbol{x};\boldsymbol{x}_k^e) d\Omega$
end
for *each element e* **do**
  $K_k^e \leftarrow \dfrac{1}{I_k^e}$
  $\widetilde{\boldsymbol{K}}^e \leftarrow \boldsymbol{\mathcal{M}}^{-1} \sum_{k=1}^{N_Q} K_k^e \boldsymbol{\Phi}(\boldsymbol{x}_k^e) w_k j_e(\boldsymbol{x}_k^e)$ (Eq. (35), with quadrature used to evaluate the integral)
end

3.6.3. Examples

Fig. 12 displays example distributions of K, with $\alpha = 4$, for three of the 2D meshes from Section 3.5.3: Mesh 1 with structured quadrilateral elements (Fig. 8(a)), Mesh 2 with unstructured triangular elements (Fig. 9(b)), and Mesh 3 with $q = 2$ quadrilateral elements (Figs. 8(c) and 8(d)). K_h is obtained using the procedure described in Algorithm 1.

The K-distribution for the first of these is given in Fig. 12(a). Since the kernels on this uniform mesh never extend past the neighborhood of node-sharing elements, K is essentially unity in the majority of the domain. At walls, but away

from the corners, $K \approx 2$ since approximately half of the kernel lies outside the domain. At the corners of the domain, the rescaling factor approaches a value of four since only a quarter of the kernel overlaps with the domain interior.

The second example K-distribution is displayed in Fig. 12(b). Recall that in Fig. 9(b), one of the sample ellipses extends past the node-sharing elements of the host element. It can be observed that K is noticeably greater than unity in the vicinity of said ellipse. Note that the contour levels are prescribed in such a way that this can be more easily visualized. As expected, K is large near the boundaries.

For the final mesh, two different views of K are provided. In Fig. 12(c), we zoom in on a section of the inner annular boundary. The $K > 1$ region closely follows the profile of the boundary. Note that $K < 2$ along the wall since more than half of the kernel lies inside the domain, given the curvature of the boundary face. Fig. 12(d) zooms in on the rightmost corner of the outer annular boundary, located at $\boldsymbol{x} = [1, 0]^T$. $K \approx 2$ along the portion of the $y = 0$ line away from the boundary. $K > 4$ at the corner, again due to the curvature of the boundary face, which in this case leads to less than a quarter of the kernel overlapping with the domain.

With this rescaling approach, along with the proposed formulation for constructing smooth anisotropic kernels, low projection errors can be achieved on all 2D and 3D meshes in Section 3.5. Specifically, for each of these meshes, 10,000 particles are initialized at random locations throughout the respective domains. With $\alpha = 4$ and $Q = 12$, projection errors less than 0.1% are obtained on all meshes. This combination of α and Q is chosen based on Figs. 4(b) and 5(b) as an appropriate balance between accuracy and computational cost. The performance of the developed methodology will be further evaluated in particle-laden flow simulations in Section 4.

3.7. Numerical implementation

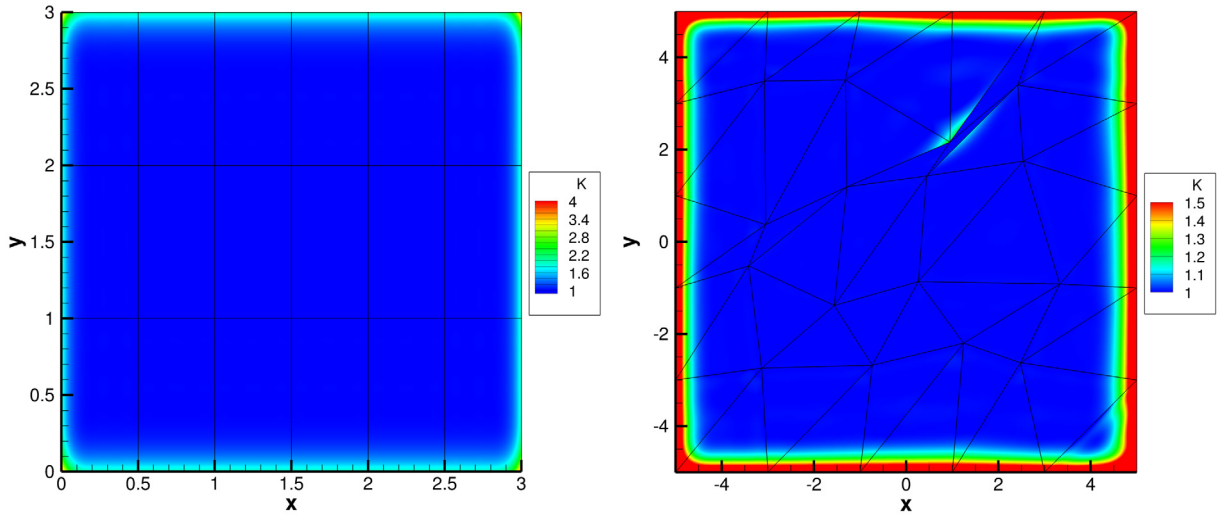
Before proceeding to the results of numerical simulations, we conclude this section with the implementation details of the entire solution procedure for the carrier and disperse phases at each time step. These are provided in Algorithm 2, in which Forward Euler time stepping is assumed for simplicity. If a multi-stage method, such as Runge-Kutta schemes, is employed, then the solution procedure is performed at each stage. Note that since the focus here is on the disperse phase, only a broad overview of the carrier phase solution procedure is provided.

Algorithm 2: Solution procedure for both the carrier and disperse phases at each time step from the perspective of Processor P. Forward Euler time stepping is assumed for both phases for simplicity. Note that the exact enumeration of the steps here does not necessarily give the most efficient implementation; they are simply presented as such for clarity. Since the focus is on the disperse phase, only a broad overview of the carrier phase solution procedure is provided.

```
for each particle i do
    Interpolate state of carrier fluid to particle centroid
    Compute F_i and Q_i (RHS of Eqs. (7))
    Compute quantities for kernel construction: K_h^o(x_{d,i}), H_s^{-1,o}(x_{d,i}), and eigenvalues of H_s^o(x_{d,i}), where the superscript "o" denotes the host element
    if host element shares a node with an element on another processor then
        Copy particle to node-sharing processor(s) as ghost particle (including kernel quantities)
    end
end
for each particle i (including ghost particles) do
    Compute source term contribution on host element (Eq. (37))
    for each element l on Processor P sharing a node with host element do
        Compute source term contribution on element l (Eq. (37))
    end
end
Delete ghost particles
for each particle i do
    Advance particle in time by solving Eqs. (7)
    Identify new host element and compute position in reference space, ξ_{d,i}, using search-locate procedure by Allievi and Bermejo [22]
end
for each element e do
    Advance local solution, U_h^e, in time by solving Eq. (5)
end
```

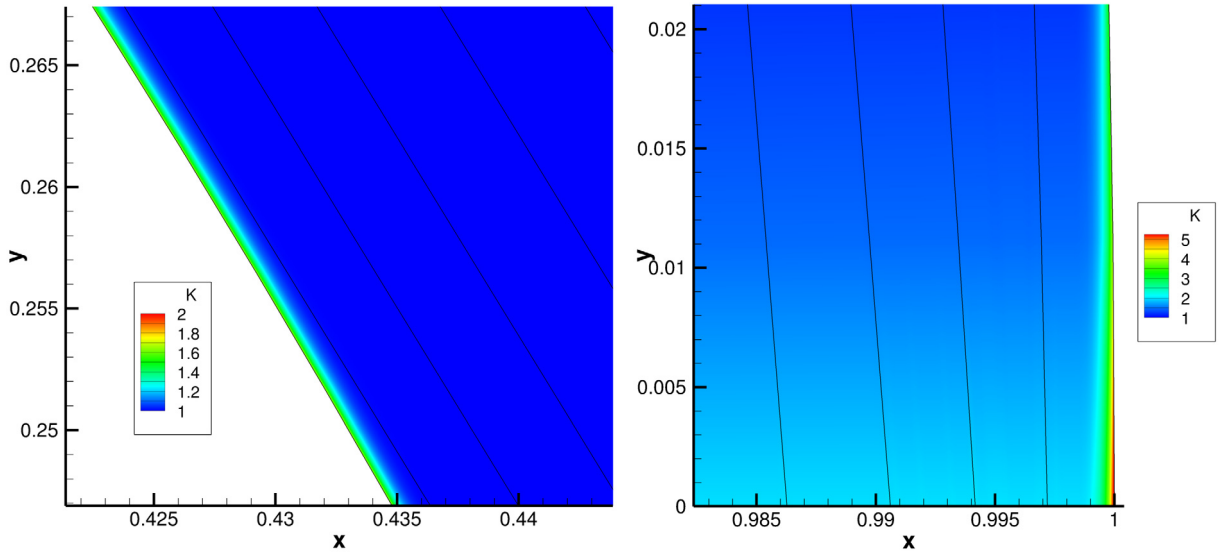
4. Results

In this section, we present results obtained with the developed reverse-coupling methodology. First, we apply it to supersonic dusty flow over a flat plate. Next, we compute the interaction between a moving shock and a spanwise-homogeneous cloud of particles. The third test case entails a hypersonic flow laden with small dust particles over a sphere. In these first three test cases, the reverse coupling is not significant and the particles are small relative to the elements of the mesh;



(a) Mesh 1, quadrilaterals

(b) Mesh 2, triangles



(c) Mesh 3, quadrilaterals (first view)

(d) Mesh 3, quadrilaterals (second view)

Fig. 12. Example distributions of K for three of the 2D meshes from Section 3.5.3: (a) Mesh 1 with structured quadrilateral elements (Fig. 8(a), (b) Mesh 2 with unstructured triangular elements (Fig. 9(b)), and (c) Mesh 3 with $q = 2$ quadrilateral elements (Figs. 8(c) and 8(d)). In all cases, $\alpha = 4$. In (b), the contour levels are prescribed so as to emphasize how the rescaling approach addresses the scenario in which the kernel extends past the neighborhood of node-sharing elements.

therefore, results obtained with delta shape functions are a good benchmark for verifying correct implementation and testing the accuracy of the proposed methodology. The fourth test case comprises the interaction between a moving shock and a spanwise-inhomogeneous cloud of particles, and the final case consists of a hypersonic flow laden with medium-sized dust particles over a sphere. Under the conditions considered, the use of delta functions leads to noise and instabilities that in certain cases result in solver divergence; conversely, the use of smooth anisotropic shape functions suppresses instabilities while maintaining good accuracy and computational cost. This is consistent with the guidelines proposed in Section 3.5.2 regarding the importance of using smooth kernels, given the larger particles relative to the mesh elements. In each of the five test cases, we compare our results with those from theory, experiments, and/or other flow solvers.

In these test cases, the time step sizes of both phases are equal and simply chosen such that the CFL of the carrier phase is approximately unity. Although not a concern here, substepping of the disperse phase can be employed in the case of strict stiffness constraints [1]. Using smaller time step sizes does not noticeably influence on the solution.

Table 1
Flow parameters for supersonic dusty flow over a flat plate. Ma_∞ is the freestream Mach number, Re_∞ is the freestream Reynolds number, T_{wall} is the temperature of the isothermal wall, T_∞ is the free-stream temperature, and β is the mass loading ratio between the disperse and carrier phases. Re_∞ is based on the particle momentum relaxation length.

Ma_∞	Re_∞	$T_{\mathrm{wall}}/T_\infty$	β	Pr	D (m)	ρ_d (kg/m^3)	c_d (J/kg/K)
1.5	10^4	0.5	1.0	1.0	10^{-6}	2700	1040

4.1. Supersonic dusty flow over a flat plate

We consider the problem analytically investigated by Wang and Glass [29] of steady compressible laminar boundary-layer flow of a dilute dusty gas over a semi-infinite flat plate. We previously simulated this flow in Ref. [12] using delta functions, the results of which are used here to assess the accuracy of our proposed methodology.

Table 1 summarizes the flow conditions. The specific heat at constant pressure of the carrier gas is equal to the specific heat of the particle phase. To remain consistent with the theoretical results, we employ a constant Nusselt number, $\mathrm{Nu} = 2$, and Stokes drag, given as

$$C_D = \frac{24}{\mathrm{Re}_d}, \tag{38}$$

to compute the drag coefficient. The DG solutions are computed with $p = 2$. The mesh has 56 elements in the streamwise direction and 80 elements in the wall-normal direction, with high-aspect-ratio elements near the wall. Implicit third-order backward-differencing and the third-order Adams-Bashforth scheme are used to evolve the carrier and disperse phases in time, respectively. After a steady-state pure-gas solution is achieved, particles are injected at each time step along the inflow boundary. Further details on the setup can be found in Refs. [12] and [29].

Fig. 13 shows distributions of streamwise velocity and temperature in a selected portion of the near-wall region for both the pure gas (without particles) and dusty gas (with particles). $(\cdot)^*$ denotes a nondimensionalized quantity; additional details about the nondimensionalization can be found in Ref. [29]. The mesh is superimposed to illustrate the high aspect ratios of the near-wall elements (note, however, that the nondimensionalization of y is a nonlinear transformation). For both the pure and dusty gases, the thermal boundary layer is thinner than the velocity boundary layer. The presence of dust particles gives rise to slightly thinner boundary layers due to the transfer of momentum and energy to the carrier gas. This leads to an increase in the surface heat flux, displayed in Fig. 14. The theoretical and DG results are provided for both the pure and dusty gases. Good agreement between the numerical and theoretical solutions is observed. At higher x^*, slightly greater differences are observed since some of the assumptions made in the theoretical investigation break down. There are no noticeable discrepancies between the results computed with the two shape functions.

4.2. Moving shock interacting with a spanwise-homogeneous particle cloud

Here, we compute the 1D interaction between a moving shock and a spanwise-homogeneous cloud of particles. This problem was studied both experimentally and computationally by Boiko et al. [30]. In their simulations, they employed a third-order finite-difference scheme for the carrier phase and solved a collisionless kinetic equation in Lagrangian variables to model the disperse phase. Jacobs and Don simulated this flow as well using a high-order WENO-Z Euler-Lagrange method [2]. Comparisons between our results and those of the experiment and the other computations will be provided.

Table 2 shows parameters for the disperse phase as well as the pre-shock state of the carrier gas (air). The right-moving shock is initialized at $x = 0$ with a Mach number of 2.8. The post-shock state can be computed using the Rankine-Hugoniot jump conditions. The monodisperse particle cloud is located between $x = 0$ and $x = 0.0156$ m, initialized with a volume fraction of 3%. Viscous effects are ignored, i.e. $\boldsymbol{F}_v = 0$, except for interphase drag and heating. We employ the drag and Nusselt number correlations used by Boiko et al. [30], given as

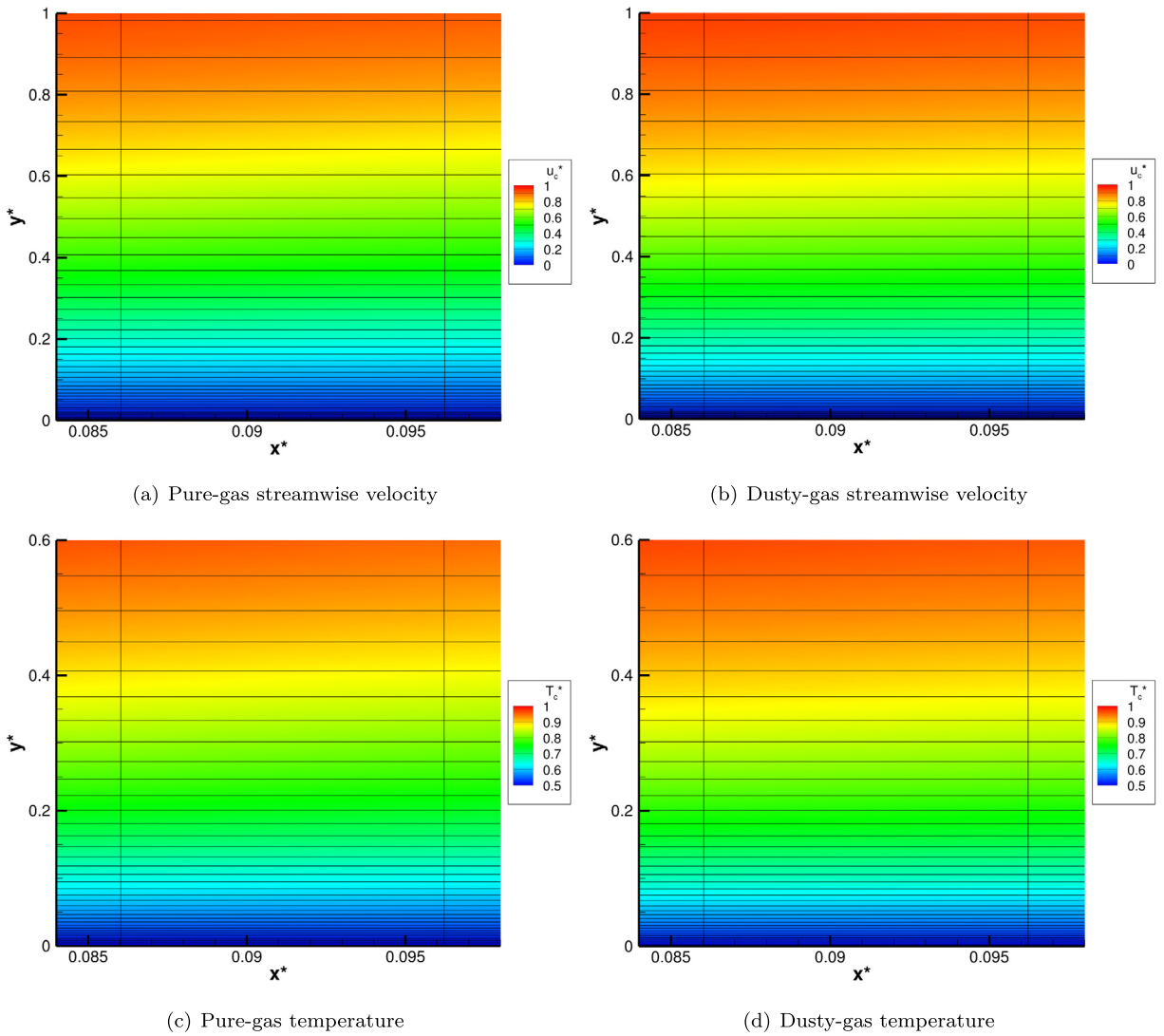
$$C_D = \left(0.38 + \frac{24}{\mathrm{Re}_d} + \frac{4}{\mathrm{Re}_d^{0.5}}\right)\left[1 + \exp\left(\frac{-0.43}{\mathrm{Ma}_d^{4.67}}\right)\right], \tag{39a}$$

$$\mathrm{Nu} = 2 + 0.6\mathrm{Re}_d^{0.5}\mathrm{Pr}^{0.33}. \tag{39b}$$

Just as in Refs. [2] and [30], we also account for the drag force induced by the pressure gradient, written as

$$\boldsymbol{F}_p = -\frac{1}{V_d}\nabla P. \tag{40}$$

We consider two mesh spacings: $\Delta x = 0.0312$ m and $\Delta x = 0.0156$ m. Both meshes are extruded one element in the spanwise direction. The domain is $[-0.04, 0.18] \times [0, 0.01]$ m^2. Second-order backwards differencing and the second-order



(a) Pure-gas streamwise velocity

(b) Dusty-gas streamwise velocity

(c) Pure-gas temperature

(d) Dusty-gas temperature

Fig. 13. Pure-gas and dusty-gas distributions of nondimensionalized streamwise velocity and temperature for supersonic dusty flow over a flat plate. The mesh is superimposed.

Table 2
Disperse phase parameters and pre-shock state of the carrier gas for shock interaction with a spanwise-homogeneous particle cloud.

ρ_d (kg/m^3)	D (μm)	c_d (J/kg)	ρ_c (kg/m^3)	u_c (m/s)	P_c (bar)
1200	300	1400	1.2	0	1

Adams-Bashforth method are employed to temporally advance the carrier and disperse phases, respectively. Post-shock conditions are prescribed at the left end of the domain, while pre-shock conditions are prescribed at the right end. Symmetry is imposed at the top and bottom boundaries. We compute this problem with $p = 1$, 2, and 3 on each mesh.

Figs. 15 and 16 shows the computed pressure and Mach number profiles, respectively, at $t = 50$ μs, $t = 100$ μs, $t = 150$ μs. Fig. 17 displays the streamwise velocity distributions, superimposed with particle locations, at the same times for $p = 3$, $\Delta x = 0.0156$ m. At $t = 50$ μs, the rightmost pressure jump in Figs. 15(a) and 15(f) is due to the initial shock, which has traveled past the particle cloud. This pressure jump is slightly smaller than in the absence of particles since some of the shock energy is lost to the particles. The interaction between the shock and particles produces a reflected shock as well as an expansion fan that forms inside the particle cloud. Across the expansion fan, the carrier gas accelerates and the pressure decreases continuously to the post-shock value of the rightmost pressure jump. At $t = 50$ μs, the variation of Mach number inside the expansion fan is non-monotonic. As time advances, it becomes monotonic and less oscillatory. Additionally, the

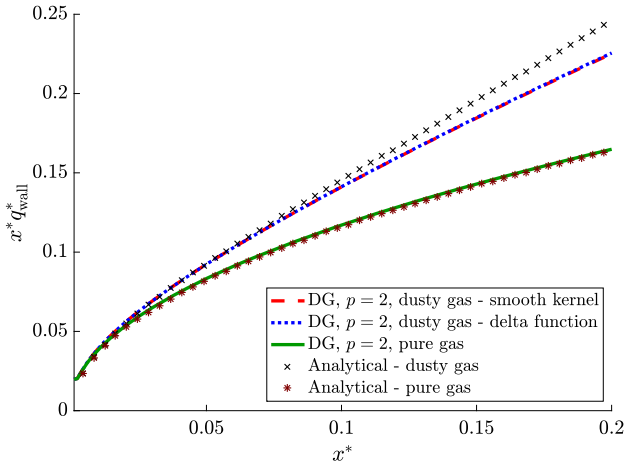


Fig. 14. Nondimensionalized heat flux profiles along the flat plate for both the pure and dusty gases obtained in the simulation of supersonic dusty flow over a flat plate. Theoretical and numerical results are provided. A comparison between the delta shape function and the smooth polynomial kernel is also provided.

reflected shock travels further leftward and the size of the particle cloud increases. The tail of the particle cloud becomes more dilute than the head.

The overall flow topology is well-captured even in the coarsest case ($p = 1$, $\Delta x = 0.0312$ m). However, increasing p and decreasing the mesh spacing enables sharper representation of the shocks and the expansion wave. The improvement in the solution between $p = 1$ and $p = 2$ is greater than that between $p = 2$ and $p = 3$. These improvements are less noticeable on the finer mesh. The mesh convergence in the pressure profiles observed in Fig. 15 reinforces the notion that the extra diffusion step discussed in Section 3.5.2 is not necessary for the small particles (relative to the element size) considered here, despite the mesh dependence of the kernel size.

Good agreement with the results by Jacobs and Don [2] is observed, especially for the finer runs. Note that comparisons with their results are provided only in the pressure profiles. Discrepancies with the results by Boiko et al. [30] are greater, potentially due to the different modeling approach and less mesh resolution in their simulations.

Fig. 18 gives the temporal evolution of $x_s(t)$, the streamwise position of the head of the particle cloud. Comparisons with the experimental and computational results by Boiko et al. [30] are provided. Higher resolution in the DG simulations yields slightly lower x_s. At early and late times, good agreement is observed with the experimental results. At intermediate times, there are small discrepancies between the DG simulations and the experiment. These discrepancies are reduced for the finer DG results. Overall, the DG simulations agree more closely with the experiment than do the computations by Boiko et al. [30].

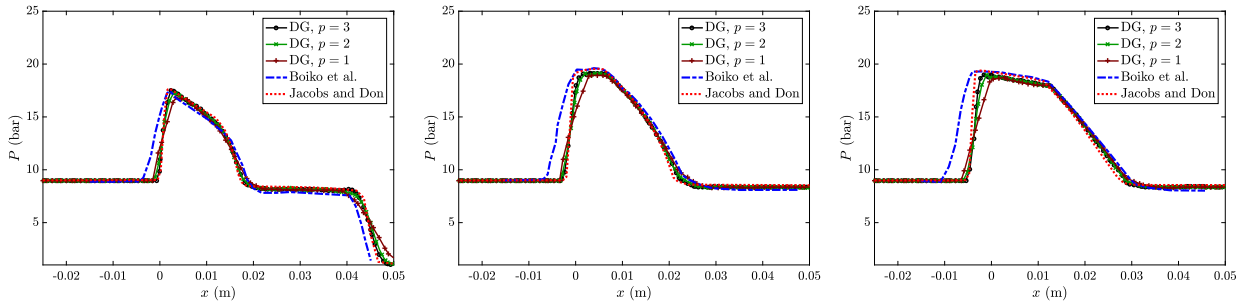
To test robustness of the developed reverse-coupling methodology across different types of meshes, we recompute this problem on the following additional meshes: the first is a hexahedral mesh obtained by extruding the fine quadrilateral mesh ($\Delta x = 0.0156$ m) one element in the z-direction, and the second is obtained by splitting each quadrilateral in the fine mesh into two triangles. The pressure and Mach number profiles are given in Fig. 19. We also benchmark these results against those computed with delta shape functions. The solution procedure for the disperse phase (not including the standard carrier phase solver) is approximately 90% more expensive with smooth kernels ($\alpha = 4$, $Q = 12$) than with delta functions, largely due to the use of numerical quadrature to evaluate the back-coupling source term. Apart from a slight discrepancy in the right-moving incident shock, there are essentially no differences among all simulations, demonstrating insensitivity to mesh topology.

4.3. Hypersonic flow laden with small particles over a blunt body

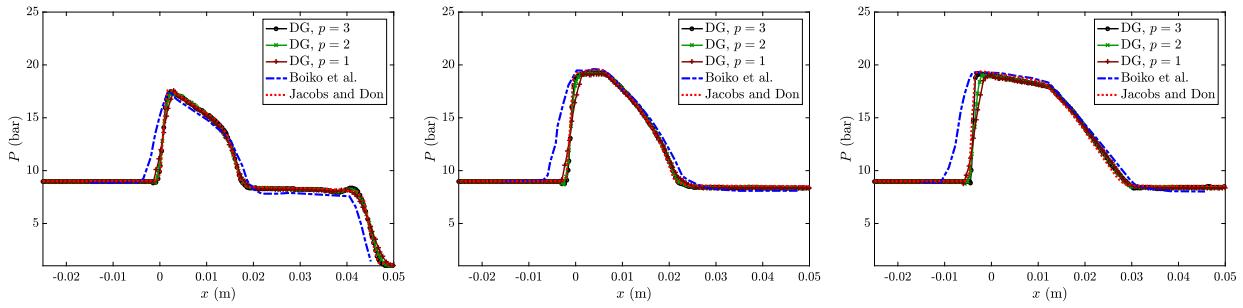
In this section, we revisit a problem computed in Ref. [12] using delta shape functions. Good agreement with experiments was obtained. Our goal here is to ensure consistency of the proposed reverse-coupling methodology with delta functions in a test case that comprises all of the following:

- Moderate back-coupling of the particles to the carrier gas, such that delta functions achieve stable and accurate results
- Curved boundaries
- High-aspect-ratio elements

This flow configuration, which satisfies all of the above, entails a hypersonic flow laden with small particles over a sphere forebody. This was experimentally studied by Vasilevksii et al. [31,32] in the U1-1M shock tunnel at the Central Aerohydro-

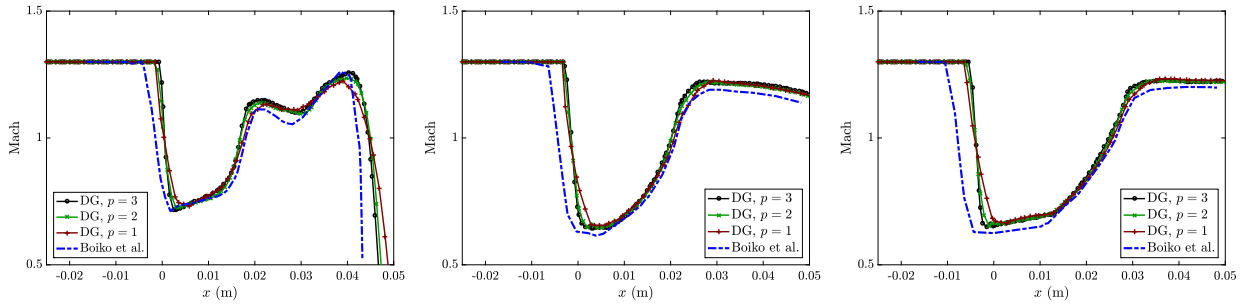


(a) $p = 1$, $\Delta x = 0.0312$ m, $t = 50$ μs (b) $p = 2$, $\Delta x = 0.0312$ m, $t = 100$ μs (c) $p = 3$, $\Delta x = 0.0312$ m, $t = 150$ μs

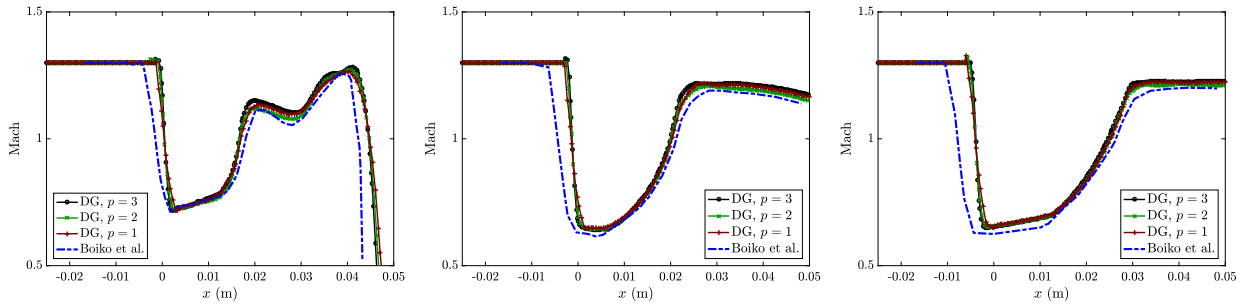


(d) $p = 1$, $\Delta x = 0.0156$ m, $t = 50$ μs (e) $p = 2$, $\Delta x = 0.0156$ m, $t = 100$ μs (f) $p = 3$, $\Delta x = 0.0156$ m, $t = 150$ μs

Fig. 15. Pressure profiles at $t = 50$ μs, $t = 100$ μs, $t = 150$ μs for shock interaction with a spanwise-homogeneous cloud of particles. Results from the computations by Boiko et al. [30] and Jacobs and Don [2] are provided for comparison.

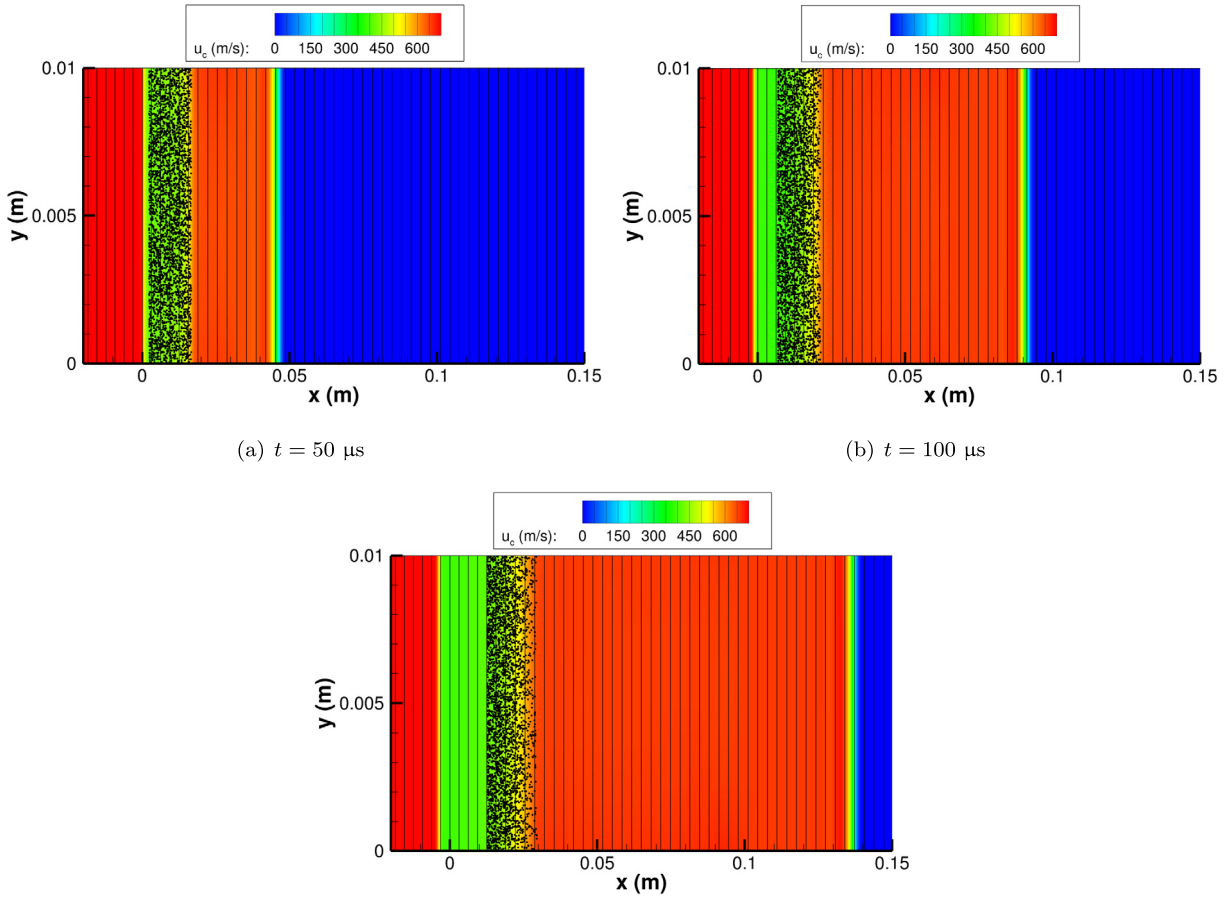


(a) $p = 1$, $\Delta x = 0.0312$ m, $t = 50$ μs (b) $p = 2$, $\Delta x = 0.0312$ m, $t = 100$ μs (c) $p = 3$, $\Delta x = 0.0312$ m, $t = 150$ μs



(d) $p = 1$, $\Delta x = 0.0156$ m, $t = 50$ μs (e) $p = 2$, $\Delta x = 0.0156$ m, $t = 100$ μs (f) $p = 3$, $\Delta x = 0.0156$ m, $t = 150$ μs

Fig. 16. Mach number profiles at $t = 50$ μs, $t = 100$ μs, $t = 150$ μs for shock interaction with a spanwise-homogeneous cloud of particles. Results from the computations by Boiko et al. [30] and Jacobs and Don [2] are provided for comparison.



(a) $t = 50$ μs (b) $t = 100$ μs

(c) $t = 150$ μs

Fig. 17. Streamwise velocity distributions, superimposed with particle locations, at $t = 50$ μs, $t = 100$ μs, $t = 150$ μs for shock interaction with a spanwise-homogeneous cloud of particles. Results are given for $p = 3$, $\Delta x = 0.0156$ m. Only 1% of all particles is shown. For visual clarity, every other mesh line is drawn.

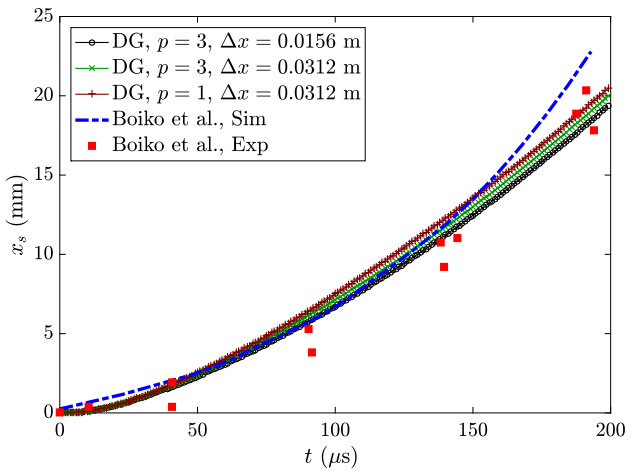
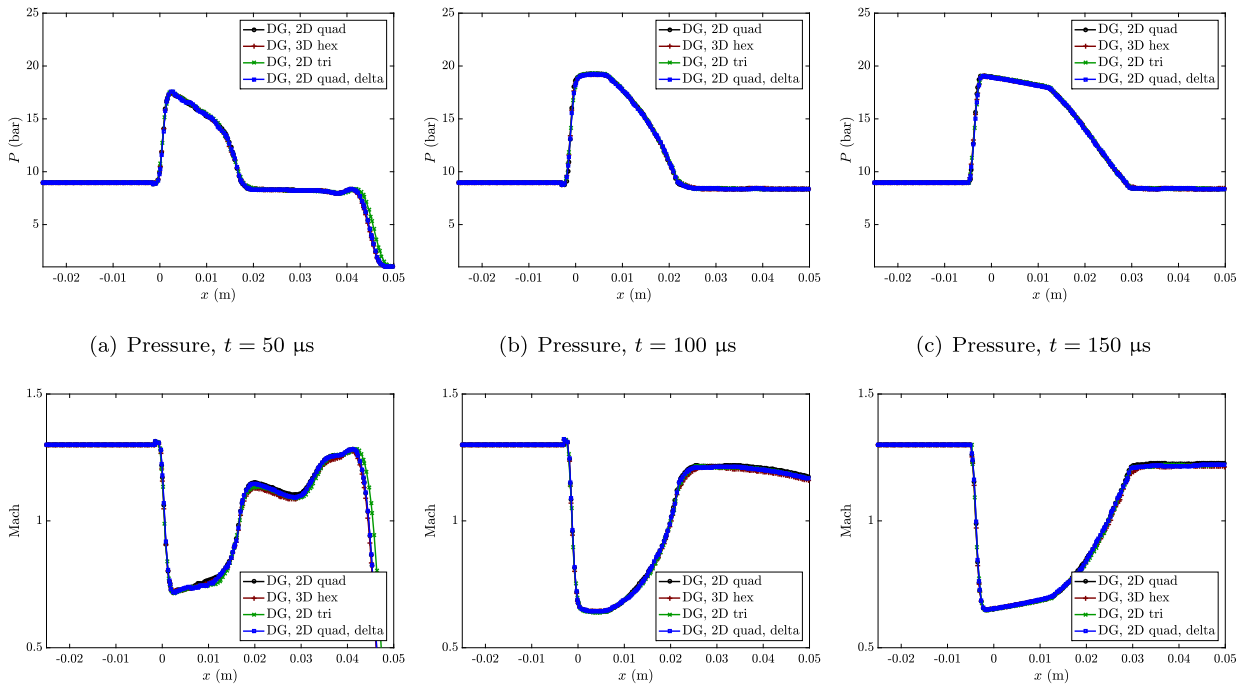


Fig. 18. Temporal evolution of $x_s(t)$, the streamwise position of the head of the particle cloud. Comparisons with the experimental and computational results by Boiko et al. [30] are provided.



(a) Pressure, $t = 50$ μs (b) Pressure, $t = 100$ μs (c) Pressure, $t = 150$ μs

(d) Mach number, $t = 50$ μs (e) Mach number, $t = 100$ μs (f) Mach number, $t = 150$ μs

Fig. 19. Pressure and Mach number profiles at $t = 50$ μs, $t = 100$ μs, $t = 150$ μs for shock interaction with a spanwise-homogeneous cloud of particles computed with different types of meshes. Results obtained with delta shape functions are provided for benchmark comparisons.

dynamics Institute (CAHI). Since this problem was already detailed in-depth in Refs. [12] and [33], the discussion here is condensed.

Table 3 lists the flow conditions. The carrier gas is N_2, and the dust material is SiO_2. We employ the same polydisperse size distribution as in the experiment [31]. The specific heat of the SiO_2 dust as a function of temperature is obtained from Refs. [34] and [35]. To compute drag coefficients, we employ the Henderson correlation [36], which, for $\mathrm{Ma}_d < 1$, is given as

$$\begin{aligned} C_D = &\frac{24}{\mathrm{Re}_d + S\left(4.33 + \frac{3.65-1.53T_d/T_c}{1+0.353T_d/T_c}\right)\exp\left(-0.247\frac{\mathrm{Re}_d}{S}\right)} \\ &+ \exp\left(-0.5\frac{\mathrm{Ma}_d}{\mathrm{Re}_d}\right)\left[\frac{4.5 + 0.38(0.03\mathrm{Re}_d + 0.48\sqrt{\mathrm{Re}_d})}{1 + 0.03\mathrm{Re}_d + 0.48\sqrt{\mathrm{Re}_d}} + 0.1\mathrm{Ma}_d^2 + 0.2\mathrm{Ma}_d^8\right] \\ &+ 0.6S\left[1 - \exp\left(-\frac{\mathrm{Ma}_d}{\mathrm{Re}_d}\right)\right], \end{aligned} \tag{41}$$

and, for $\mathrm{Ma}_d > 1.75$,

$$C_D = \frac{0.9 + \frac{0.34}{\mathrm{Ma}_d} + 1.86\sqrt{\frac{\mathrm{Ma}_d}{\mathrm{Re}_d}}\left[2 + \frac{2}{S^2} + \frac{1.058}{S}\sqrt{\frac{T_d}{T_c}} - \frac{1}{S^4}\right]}{1 + 1.86\sqrt{\frac{\mathrm{Ma}_d}{\mathrm{Re}_d}}}, \tag{42}$$

where $S = \mathrm{Ma}_d\sqrt{\gamma/2}$ is the molecular speed ratio. Linear interpolation is used for $1 \leq \mathrm{Ma}_d \leq 1.75$, which yields

$$C_D = C_{D,H}\big|_{\mathrm{Ma}_d=1} + \frac{4}{3}(\mathrm{Ma}_d - 1)(C_{D,H}\big|_{\mathrm{Ma}_d=1.75} - C_{D,H}\big|_{\mathrm{Ma}_d=1}). \tag{43}$$

To compute Nusselt numbers, we use the correlation by Fox et al. [37], defined as

$$\mathrm{Nu} = \frac{2\exp(-\mathrm{Ma}_d)}{1 + 17\frac{\mathrm{Ma}_d}{\mathrm{Re}_d}} + 0.459\mathrm{Pr}^{0.33}\mathrm{Re}_d^{0.55}\frac{1 + 0.5\exp(-17\frac{\mathrm{Ma}_d}{\mathrm{Re}_d})}{1.5}. \tag{44}$$

Fig. 20 shows the O-grid mesh, which consists of 60,000 $q = 2$ hexahedral elements. High-aspect-ratio elements are used near the wall in order to resolve the sharp thermal gradients. For the carrier phase, freestream conditions are prescribed at

Table 3
Flow conditions for hypersonic flow laden with small dust particles over a sphere forebody. $(\cdot)_\infty$ denotes freestream conditions, $(\cdot)_t$ indicates total quantities, R_S is the radius of the aluminum sphere, β is the mass loading ratio, and $\overline{(\cdot)}$ indicates the averaging procedure employed by Vasilevskii et al. [31].

Ma_∞	$P_{t,\infty}$ (bar)	$T_{t,\infty}$ (K)	R_S (m)	T_{wall} (K)	ρ_d (kg/m^3)	$\overline{D}$ (μm)	β
6.1	17.5	570	0.006	300	2264	0.19	1%

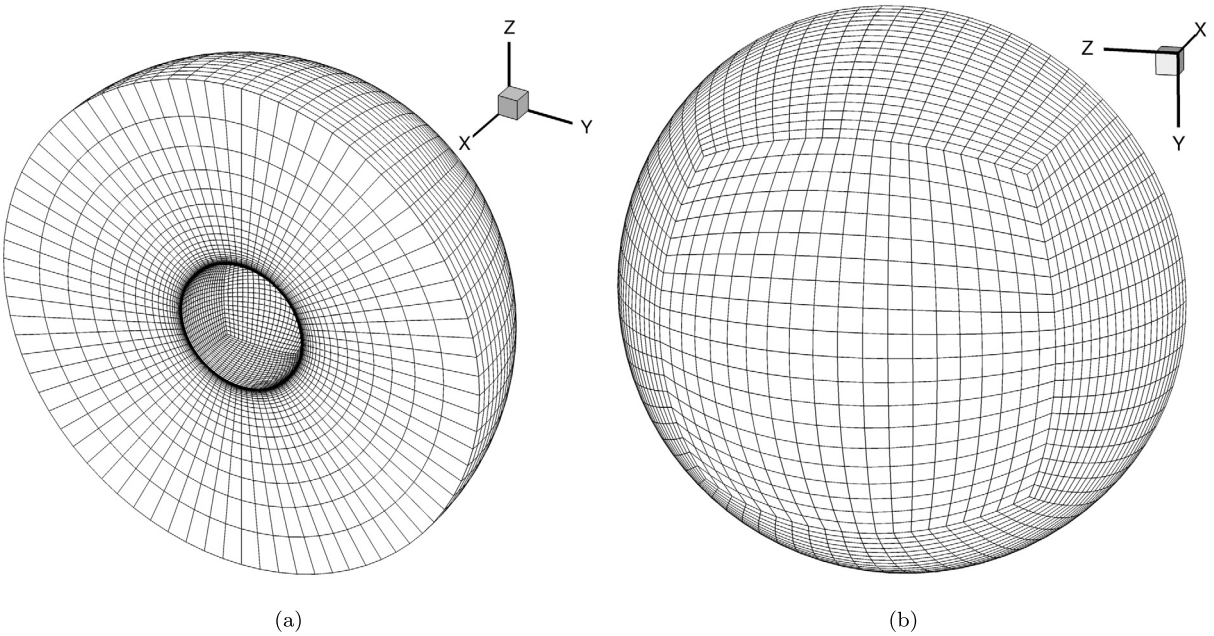
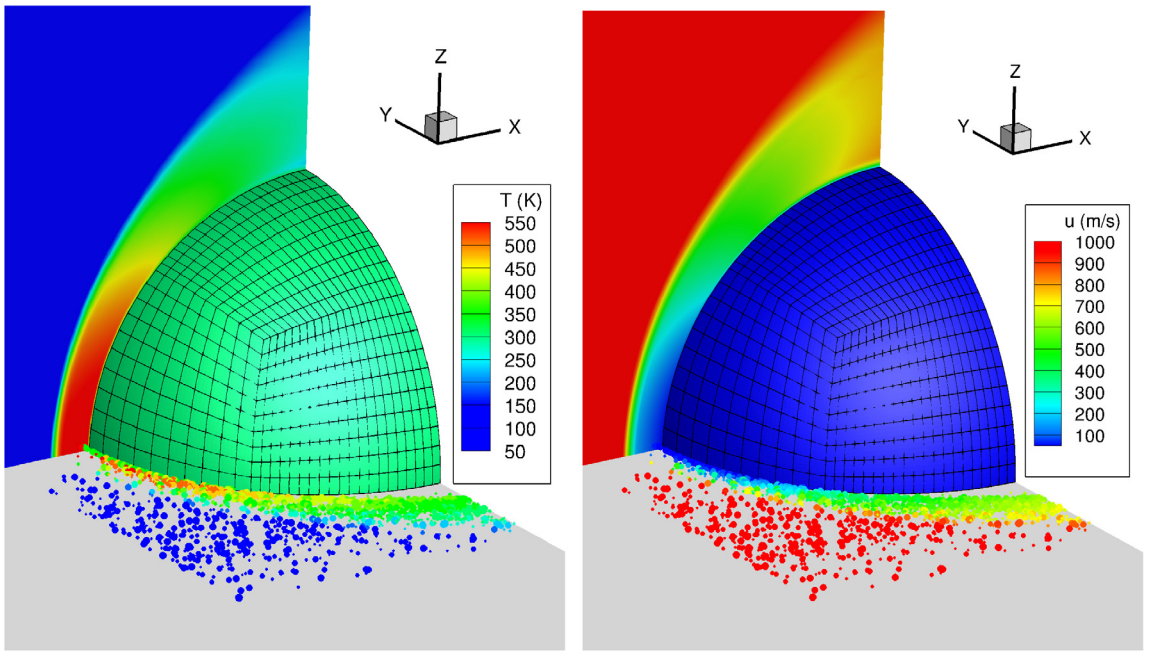


Fig. 20. Two different views of the mesh employed in hypersonic dusty flow over a blunt body. The mesh is made up of 60,000 $q = 2$ hexahedral elements.

the inflow boundary. The sphere surface is an isothermal no-slip wall. Extrapolation is used at the outflow. For the disperse phase, particles are injected along the inflow boundary at freestream conditions. Particles exit the domain at the outflow boundary. For particle-wall collisions, we use the correlations by Stasenko [38] to compute the coefficients of restitution in the normal and tangential directions. We employ third-order backward differencing to integrate the carrier gas in time and the third-order Adams-Bashforth method to integrate the particles in time. The DG solution is computed with $p = 2$ polynomials. Additional details on the setup can be found in Ref. [12].

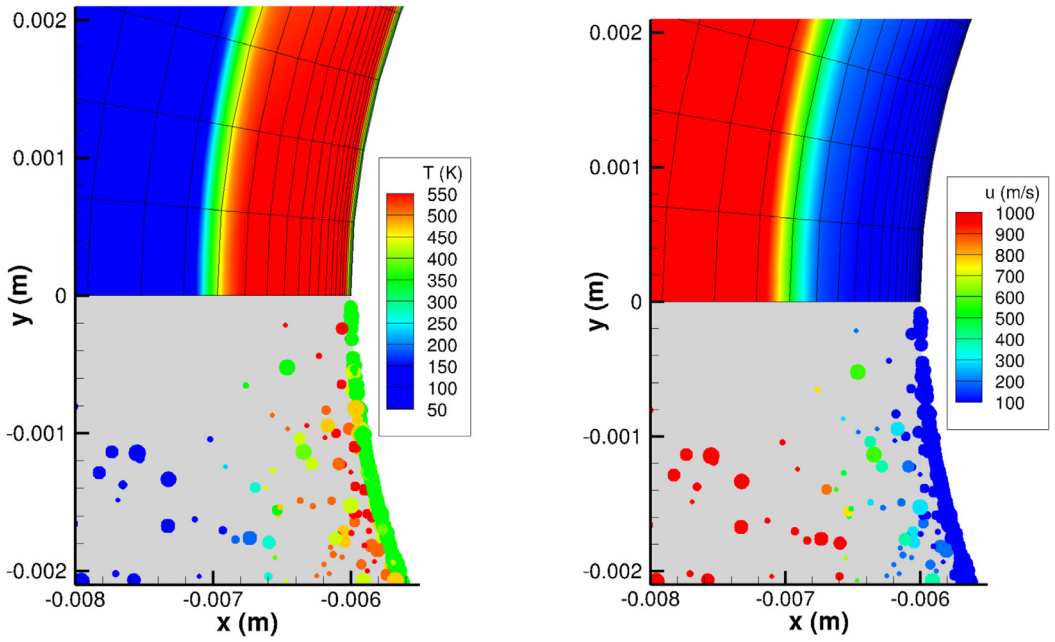
Fig. 21 shows 3D and zoomed-in 2D views of the temperature and streamwise velocity of both phases. The 3D views comprise an xz-slice of the carrier gas solution, a projection of the particles onto the xy-plane, and the sphere wall, superimposed by the surface mesh. The 2D views zoom in on the stagnation region. The top half is an xy-slice of the carrier gas solution, and the bottom half is a projection of the particles onto the xy-plane. Note that the seemingly low concentration of particles near the stagnation point is a direct consequence of the sphere-to-plane projection. The particles are indicated by circles scaled in size by particle diameter and colored by the corresponding quantities. Some smearing of the shock is evident in the carrier gas fields due to the use of artificial viscosity for stabilization; nevertheless, the shock is stably captured and free from apparent oscillations, despite no specific effort to align it with the mesh. The distribution of the disperse phase generally mirrors that of the carrier phase. As particles cross the shock, they transfer momentum to and drain thermal energy from the shock layer, thereby slowing down and heating up. Larger particles respond more slowly to the carrier gas than smaller particles. Since the particles overall have short momentum response times, they are either advected by the carrier gas past the sphere or strike the surface at low velocities. The particles that accumulate near the sphere wall deposit thermal energy to the colder boundary layer, as displayed in Fig. 21(c). This has important implications for surface heating, which will be discussed next.

Fig. 22 shows the surface heat flux profiles for the pure gas and dusty gas. Dusty-gas results obtained with delta shape functions and the proposed methodology are provided. The heat fluxes are averaged temporally, azimuthally, and over each boundary face. The horizontal dashed black line represents the stagnation-point dust-gas heat flux from the corresponding experiment [31]. Good agreement with experiments is observed. The dust-induced heat flux augmentation here is partially a consequence of the inelastic particle-wall collisions. Specifically, the kinetic energy removed from a particle upon collision is converted into heat, which is absorbed by the surface. As previously discussed, however, the impact energies and therefore the collisional energy transfer are relatively small. A greater contribution to the amplified heating is the interphase mo-



(a) 3D view of temperature

(b) 3D view of streamwise velocity



(c) Zoomed-in 2D view of temperature

(d) Zoomed-in 2D view of streamwise velocity

Fig. 21. 3D and zoomed-in 2D views of both the carrier and disperse phases for hypersonic flow laden with small particles over a sphere forebody: (a) 3D view of temperature; (b) 3D view of streamwise velocity; (c) 2D view of temperature; (d) 2D view of streamwise velocity. The particles are indicated by circles scaled in size by particle diameter and colored by the corresponding quantities. Only 6.25e-4% of all particles is shown.

mentum and energy exchange in the shock and boundary layers. Further discussion on the gas-particle interactions, particle trajectories, and mesh convergence can be found in Refs. [12] and [33].

No apparent differences are observed between the solution obtained with delta functions and that computed with smooth anisotropic kernels. These results, along with those for the previous two test cases, demonstrate correct implementation of the proposed methodology and its success in computing a variety of particle-laden flow configurations with

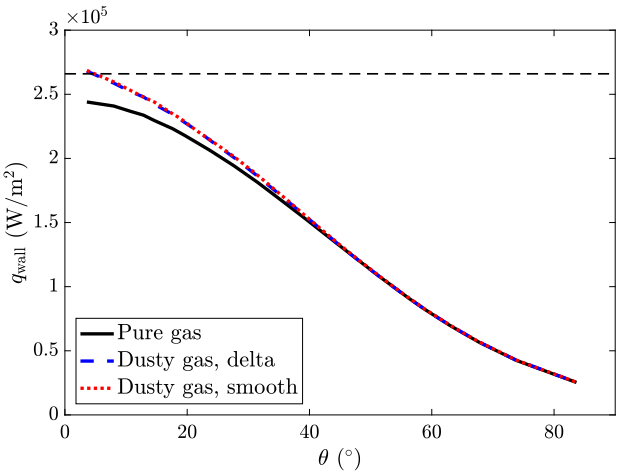


Fig. 22. Pure-gas and dusty-gas surface heat flux profiles for hypersonic flow laden with small particles over a sphere forebody. Dusty-gas heat fluxes computed with delta shape functions and smooth anisotropic kernels are given. The horizontal dashed black line represents the stagnation-point dust-gas heat flux from the corresponding experiment [31]. θ is the polar angle with respect to the $-x$-axis.

Table 4
Disperse phase parameters and pre-shock state of the carrier gas for shock interaction with a spanwise-inhomogeneous particle cloud.

ρ_d (kg/m^3)	D (μm)	c_d (J/kg/K)	ρ_c (kg/m^3)	u_c (m/s)	P_c (bar)
8900	100	435	1.2	0	1

moderate back-coupling. Under the considered conditions for the next two cases, delta functions lead to numerical instabilities and noise due to the much stronger back-coupling, illustrating the benefit of employing smooth projection kernels.

4.4. Moving shock interacting with a spanwise-inhomogeneous particle cloud

In this section, we simulate the interaction between a moving shock and a spanwise-inhomogeneous particle cloud. This builds upon the flow configuration in Section 4.2. The primary difference is that the rectangular cloud of particles does not extend to the top and bottom boundaries, eliminating the spanwise homogeneity. The physical interaction between the shock and the particle cloud is therefore elevated from 1D to 2D.

We consider the same conditions as in the simulation by Kiselev et al. [39]. Table 4 gives the disperse phase parameters and the pre-shock state of the carrier gas. The computational domain is $[0, 5.5] \times [-1.1, 1.1]$ m^2. The Mach number of the right-moving shock is 3, which is initialized at $x = 0.315$ m. The carrier gas is air. The particle cloud is initialized in equilibrium with the pre-shock gas with a volume fraction of 4% in the rectangular region $[0.315, 0.634] \times [-0.16, 0.16]$ m^2. Particles are uniformly spaced in the cloud. Just as in Section 4.2, the only viscous effects taken into account are interphase drag and heating. The same drag and Nusselt number correlations are employed. The computations here are run for over twice as long as those by Kiselev et al. [39]. In their simulations, they used the same modeling strategy as Boiko et al. for the test case in Section 4.2, i.e. a third-order finite difference method to solve the Euler equations and a Lagrangian approach to solve a collisionless kinetic equation for the disperse phase. Note that Jacobs and Don [2] also considered this flow configuration with slightly different conditions.

The computational domain is partitioned into 1000×500 quadrilateral elements. The pre- and post-shock states are prescribed at the right and left boundaries, respectively. Symmetry is imposed at the top and bottom boundaries. The DG solution is obtained with $p = 1$, 2, and 3. The density distribution and particle locations at $t = 0$ are given in Fig. 23.

We first present quantitative comparisons with the centerline pressure profiles by Kiselev et al. [39] for $t = 0.75$ and 1.5 ms in Fig. 24. Small-scale numerical instabilities seem to be present in their pressure profiles, particularly at the upstream bow shock and the downstream normal shock. Nevertheless, good agreement between the two solvers is observed. At this point, there are no significant discrepancies among the three DG solutions. Differences become apparent at later times, which will be discussed near the end of this section.

Fig. 25 displays the $p = 3$ density fields of the carrier phase, superimposed with particle locations, at $t = 0.75$, 1.5, 2.25, 3, 3.75 ms, and 4.5 ms. The results for $t = 5$ ms are shown in Fig. 26, which also includes the $p = 1$ and $p = 2$ predictions. The particle cloud acts as a blunt body, causing a reflected bow shock to form at early times. Between $t = 0$ and $t = 0.75$ ms, the incident shock loses energy as it interacts with the particle cloud, splitting into two curved shocks that bend inward. Shear layers form along the top and bottom of the cloud. A Mach reflection then forms when the curved shocks reach the $y = 0$ line, connecting each curved shock with a normal shock. A series of compression and expansion waves is then

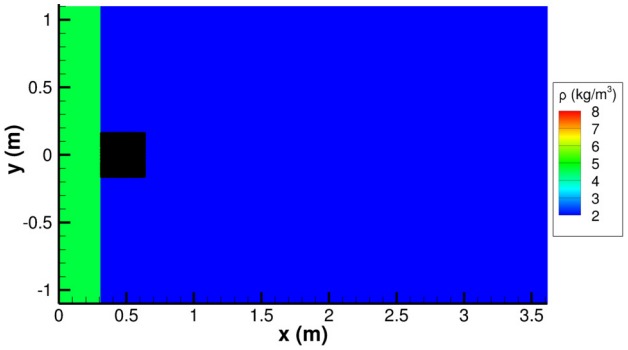
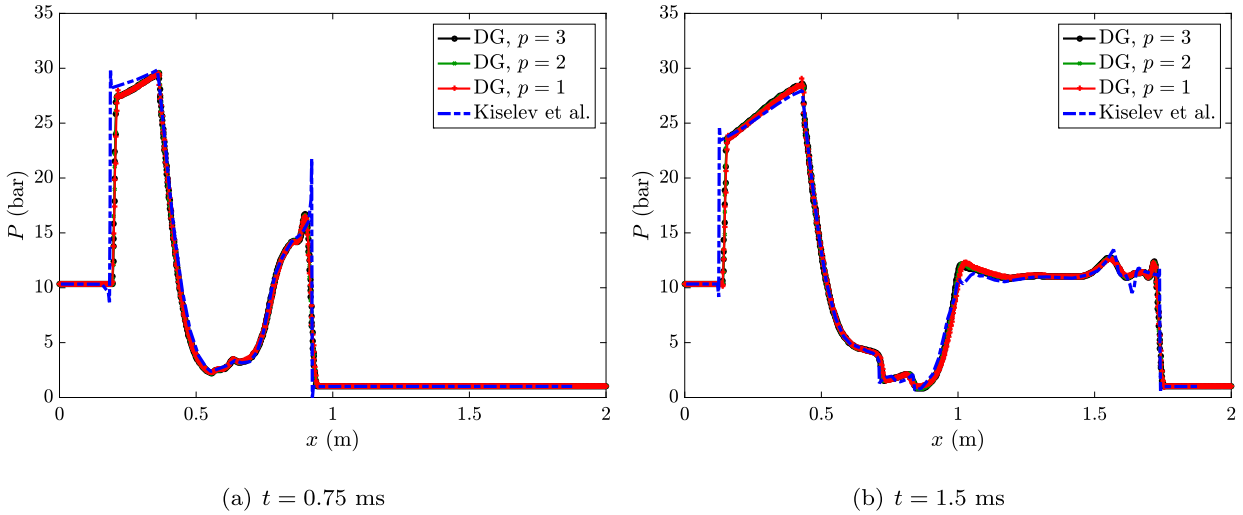


Fig. 23. Density distribution of the carrier gas, superimposed with particle locations, at $t = 0$ for shock interaction with a spanwise-inhomogeneous cloud of particles.



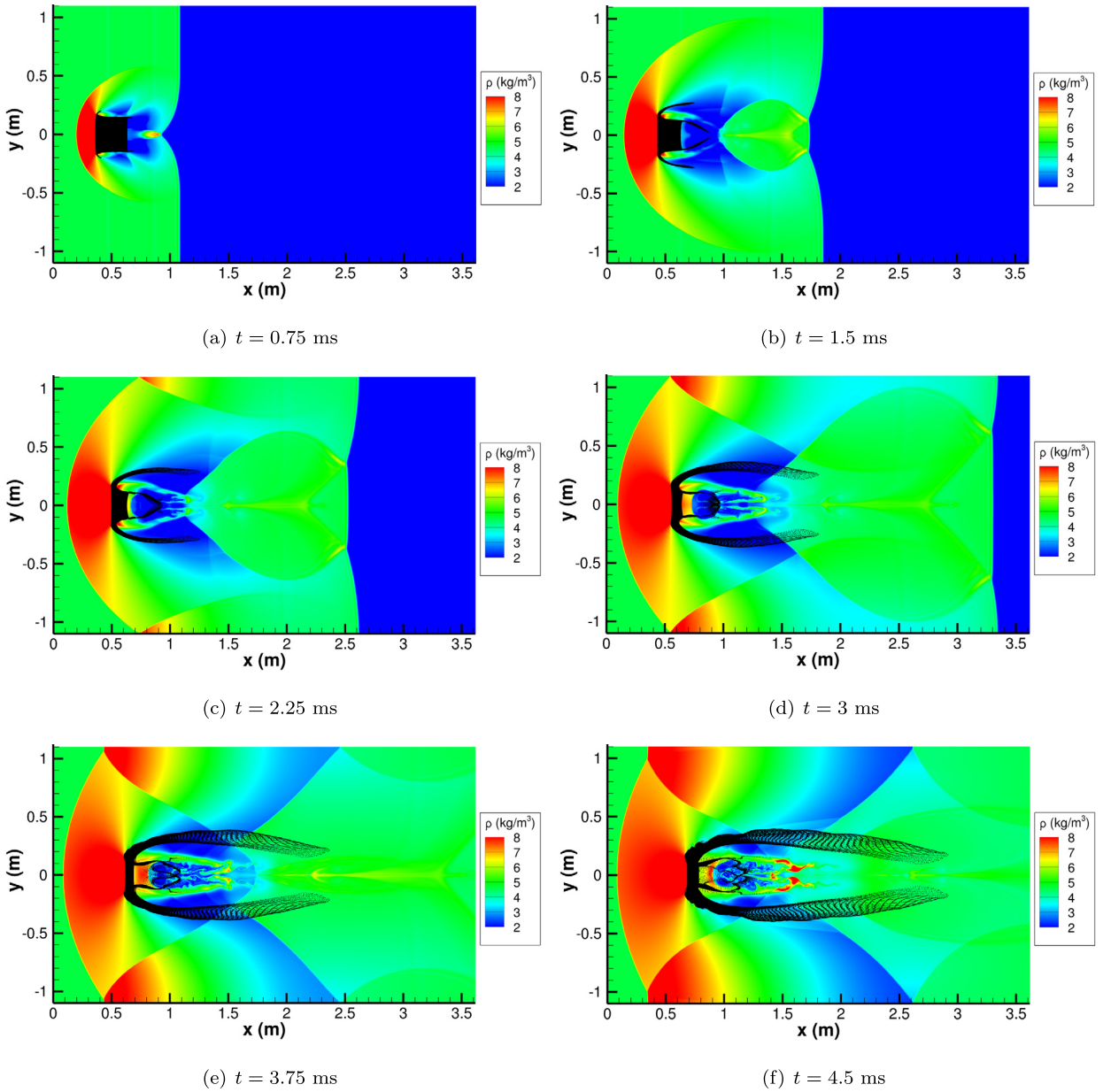
(a) $t = 0.75$ ms (b) $t = 1.5$ ms

Fig. 24. Centerline pressure profiles for $p = 1$, 2, and 3 at $t = 0.75$ and 1.5 ms for shock interaction with a spanwise-inhomogeneous cloud of particles. The computational results by Kiselev et al. [39] are included.

generated between the cloud and the normal shock. By $t = 2.25$ ms, the bow shock upstream of the cloud has reflected from the top and bottom boundaries. As the downstream normal shock elongates, the top and bottom curved shocks straighten out. Vorticity develops as a result of the interactions among the generated shocks and shear layers. Meanwhile, the particle cloud is compressed in both the x and y directions. The shear layers and vortices induce the formation of a pair of streaks along the top and bottom of the cloud that are advected downstream. This is evident in Figs. 25(b) and 25(c). At the same time, a pair of streaks in the rear of the cloud is caught in the wake and pulled toward the symmetry line. This latter pair remains trapped in the wake for the remainder of the simulation.

After $t \approx 3$ ms, the carrier flow starts to noticeably lose symmetry. The wake becomes characterized by evident hydrodynamic instabilities. The outer boundary of the top and bottom particle streaks exhibits a slightly wavy structure, especially near the particle cloud. Further downstream, a banded pattern within these streaks is evident. The tails of the particle streaks are much more dilute than the upstream regions. Upon crossing a generated oblique shock, they are deflected toward the $y = 0$ line. The particle cloud ends up highly compressed, especially in the streamwise direction. The pairs of streaks in the wake behind the cloud begin to lose their symmetric, uniform structure as a result of interactions with the vortices.

As shown in Fig. 26, there are small differences among the instantaneous density distributions, particularly in the wake of the particle cloud, across the various polynomial orders. Nevertheless, the overall flow topology is the same. It can be observed that the hydrodynamic instabilities are better-resolved in the high-order solutions. In the $p = 1$ solution, the wavy behavior of the outer boundary of the top and bottom streaks is more pronounced. Small secondary particle streaks develop and branch off the primary streaks away from the centerline. In the $p = 2$ distribution, these secondary particle streaks are somewhat suppressed, and they are slightly further suppressed in the $p = 3$ solution. The banded pattern inside the primary streaks, as well as the shape of the tails, is similar between the $p = 2$ and $p = 3$ distributions, whereas noticeable differences can be observed in the $p = 1$ solution.



(a) $t = 0.75$ ms (b) $t = 1.5$ ms

(c) $t = 2.25$ ms (d) $t = 3$ ms

(e) $t = 3.75$ ms (f) $t = 4.5$ ms

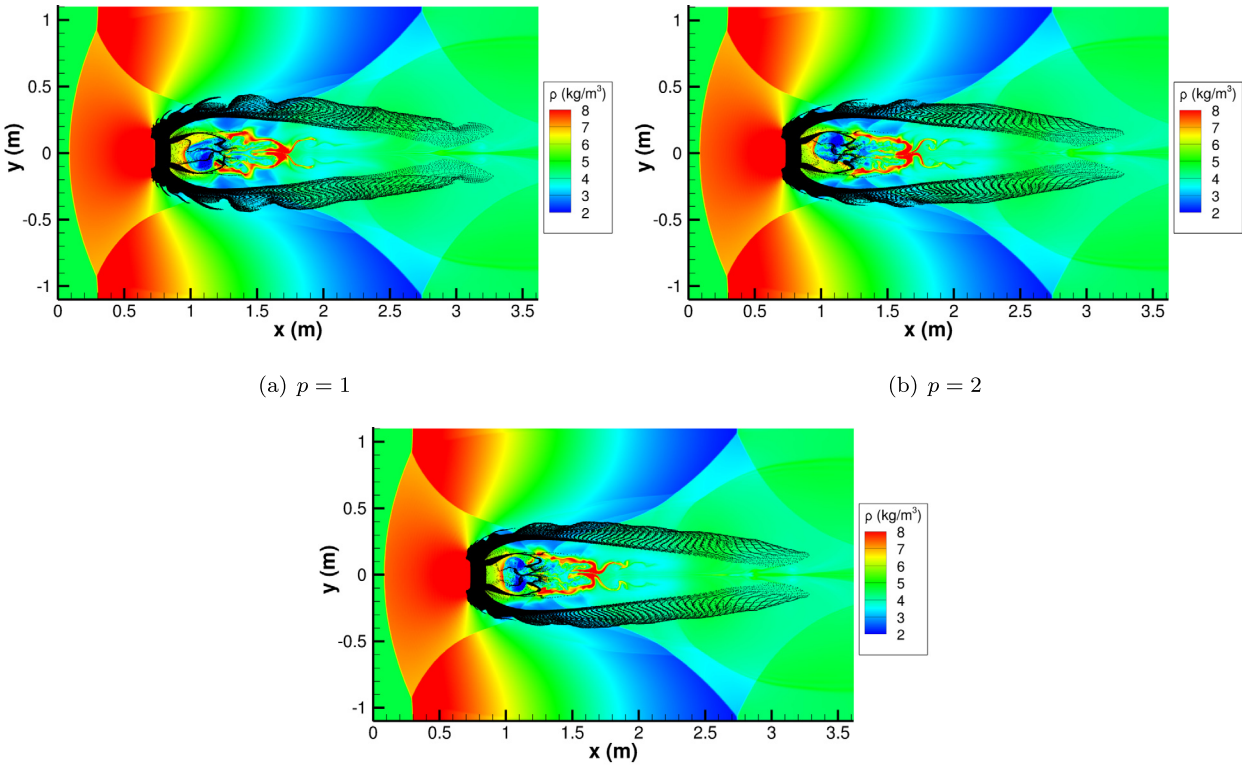
Fig. 25. $p = 3$ density distributions of the carrier gas, superimposed with particle locations, at $t = 0.75$, 1.5, 2.25, 3, 3.75, and 4.5 ms, for shock interaction with a spanwise-inhomogeneous cloud of particles. 0.02% of all particles is shown.

Fig. 27 compares density fields (without particle locations) obtained with smooth kernels and delta functions, zoomed in on the location of the particle cloud and the wake. The use of delta functions leads to significant noise and numerical instabilities inside the particle cloud and in the wake. Conversely, the solution computed with smooth kernels is mostly free from such instabilities, illustrating their benefit.

4.5. Hypersonic flow laden with medium-sized particles over a sphere

In the final test case, we simulate a flow similar to that in Section 4.3, but with larger particles. Under the conditions considered, the back-coupling of the disperse phase to the carrier gas is much stronger, and the use of delta functions causes numerical instabilities that can lead to solver divergence.

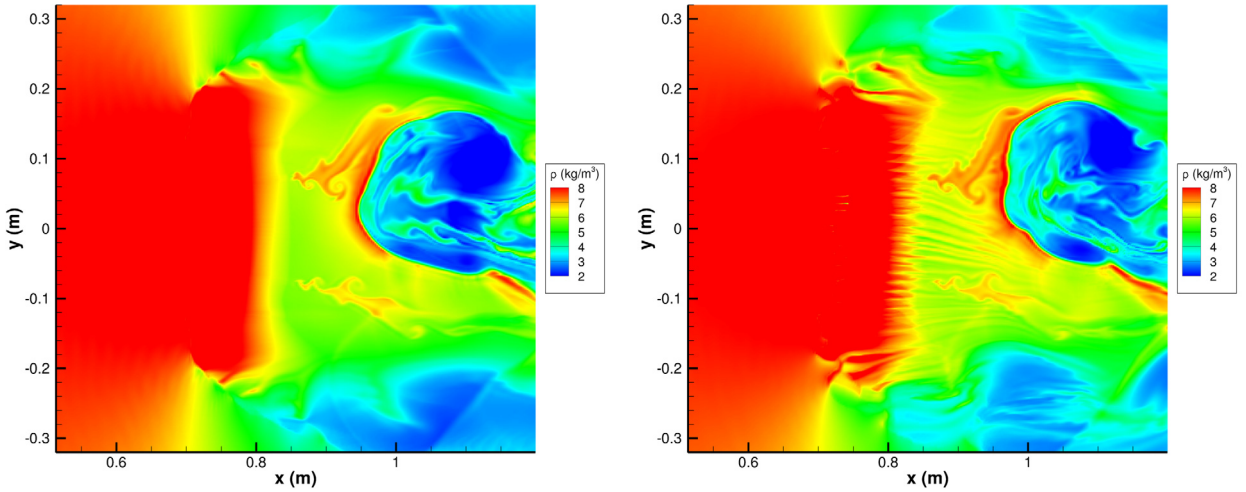
Table 5 lists the relevant parameters of the disperse phase. The majority of the setup is identical to that in Section 4.3. However, the dust material is Fe_2O_3. The polydisperse size distribution from the corresponding experiment is employed (see Ref. [31] for details). Note the higher material density and larger diameters of these particles than the SiO_2 particles



(a) $p = 1$ (b) $p = 2$

(c) $p = 3$

Fig. 26. Density distributions of the carrier gas, superimposed with particle locations, at $t = 5$ ms for shock interaction with a spanwise-inhomogeneous cloud of particles. Results for $p = 1$, 2, and 3 are given. 0.02% of all particles is shown.



(a) Smooth anisotropic kernel (b) Delta function

Fig. 27. Zoomed-in density distributions obtained with smooth kernels and delta functions for shock interaction with a spanwise-inhomogeneous cloud of particles.

previously considered. The variation of specific heat with temperature for the Fe_2O_3 particles is obtained from Refs. [35] and [40].

Fig. 28 gives 3D and zoomed-in 2D views of the temperature and streamwise velocity of both phases. Recall that the particles are indicated by circles scaled in size by particle diameter and colored by the corresponding quantities. The heavier, larger particles here equilibrate with the flow considerably slower than the SiO_2 particles. The thermal response times

Table 5
Disperse phase parameters for hypersonic flow laden with medium-sized particles over a sphere forebody. β is the mass loading ratio, and $\overline{(\cdot)}$ indicates the averaging procedure employed by Vasilevskii et al. [31].

ρ_d (kg/m^3)	$\overline{D}$ (μm)	β
5250	0.37	1%

of the particles are noticeably faster than their momentum response times. Nevertheless, the particles do not heat up significantly inside the shock layer. The thermal energy available for deposition to the carrier gas is therefore less than if the particles thermally equilibrate with the flow field, as with the SiO_2 particles. The particles retain much of their initial velocities as they approach the sphere wall, resulting in high impact energies. Furthermore, a substantially larger fraction of particles strike the surface than in the SiO_2 case. The accumulation of low-velocity particles in the vicinity of the sphere surface is due to the highly inelastic particle-wall collisions. The greater incident velocities and higher relative frequency of surface impacts results in significant collisional energy transfer. This is the primary contribution to heating augmentation, instead of interphase exchange in the shock and boundary layers, as in Section 4.3. Note that although the collisional energy transfer dominates the interphase transfer, the back-coupling is nevertheless strong, such that the flow is more susceptible to numerical instabilities and noise than with the smaller, lighter SiO_2 particles.

The resulting heat flux profile is displayed in Fig. 29. The dusty-gas heat flux here is significantly larger than in the SiO_2 case. The heating augmentation is greatest near the stagnation point because the particle-wall collisions there are characterized by normal (as opposed to tangential) impacts. At high incident normal velocities, the impact approaches a perfectly inelastic collision [38]. Consequently, almost all of the incident kinetic energy is absorbed by the sphere in the form of heat. However, the coefficient of restitution in the tangential direction is less dependent on impact speed, and the energy lost from primarily tangential collisions is comparatively smaller. This, coupled with the fact that particles crossing the shock at greater θ are more likely to miss the surface completely, is why the heat flux augmentation is smaller away from the stagnation point. Although the DG solution slightly overpredicts the dusty-gas heat flux at the stagnation point, good overall agreement is observed.

The results here, as well as those in the previous test case (Section 4.4), demonstrate the ability of the proposed reverse-coupling methodology to robustly and accurately compute particle-laden flows with a strong influence of the disperse phase on the carrier gas.

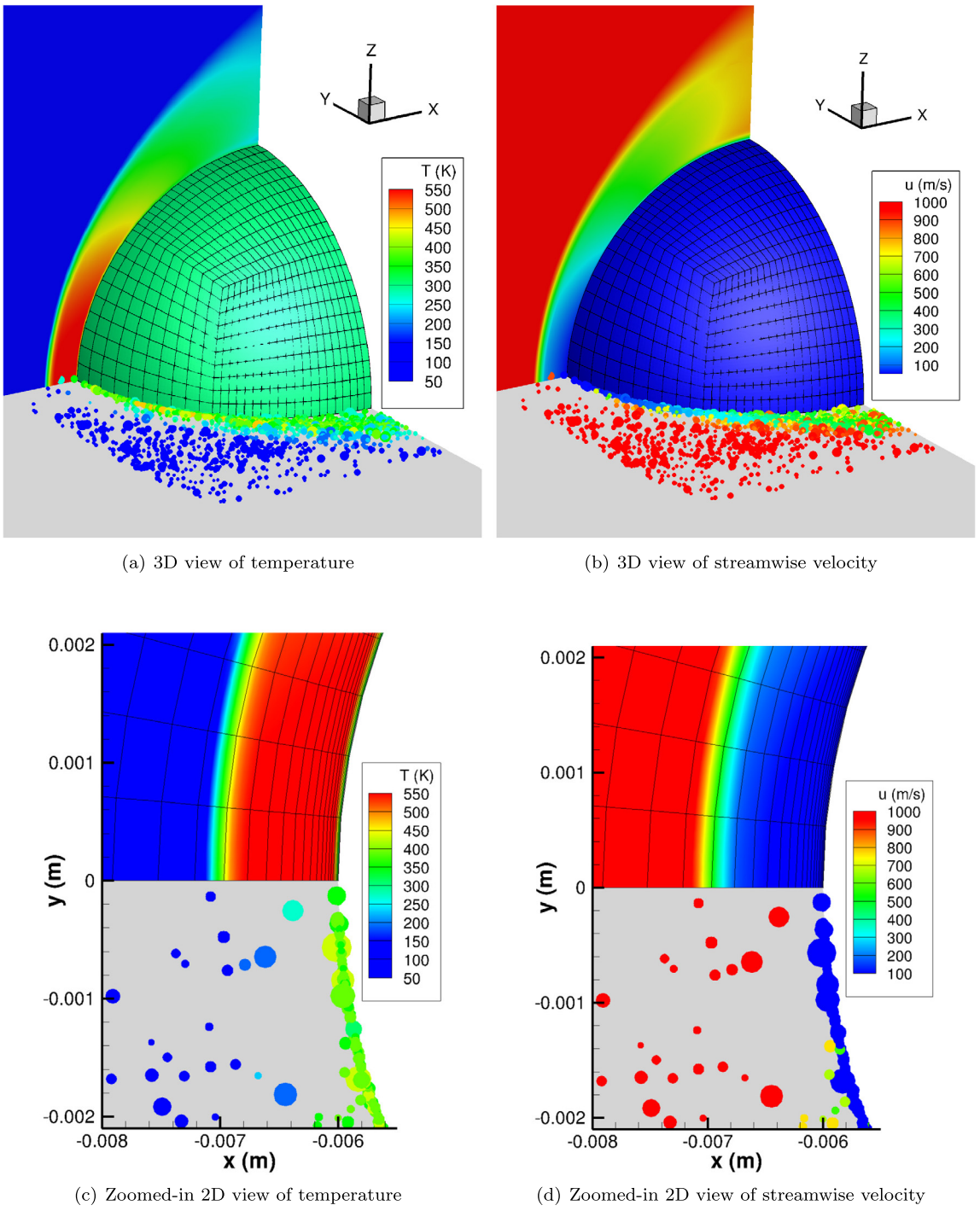
5. Concluding remarks

In this paper, we introduce a reverse-coupling methodology for projecting the action of particles onto the Eulerian mesh in Euler-Lagrange simulations of disperse multiphase flows. The proposed methodology is made up of two major ingredients with the objective of achieving a good balance among accuracy, stability, and computational efficiency. It can be applied to meshes of arbitrary topology, order, and element-to-element variations in size and orientation. Furthermore, the formulation is suited for both nodal and model basis functions.

The first ingredient entails finite-size, anisotropic projection kernels to distribute the effect of the disperse phase over the carrier fluid. We find that a polynomial-based kernel is preferable to a Gaussian-type kernel due to its amenability to numerical quadrature and zero cutoff error. The shape of the kernel is elliptical in 2D and ellipsoidal in 3D and is constructed from the mesh-implied metric commonly utilized in mesh adaptation [16,17]. The resulting kernel accurately accounts for the size and orientation of the host element and generally does not extend past the node-sharing elements. This is confirmed on a variety of 2D and 3D meshes, including unstructured, high-order, and high-aspect-ratio meshes. This type of kernel enables iterating over only this neighborhood of elements while maintaining low projection error, which is a major advantage over conventional isotropic kernels. The overall compactness of DG methods is thus maintained. At the same time, not only is it more robust than the delta shape functions we previously used [12], but it also more accurately approximates the finite-size influence of larger particles. Of course, the computational cost is increased, and implementation is more difficult. If desired, the size and orientation of the kernel can be easily modified.

The second ingredient is a strategy to conserve interphase transfer near boundaries discretized with curved elements. Specifically, we precompute a high-order polynomial approximation of the rescaling factor that ensures proper normalization of the kernel. Its general applicability to arbitrary curved elements is a notable benefit over classical strategies designed for planar elements. A secondary advantage is that this approach naturally addresses the scenario in which the kernel extends past the neighborhood of node-sharing elements. We find that the combination of these two ingredients can achieve very low projection errors on the above selection of meshes, even when the aforementioned scenario occurs.

We apply the proposed methodology to five test cases. In the first three cases, the back-coupling of the disperse phase to the carrier phase is moderate; therefore, results obtained with delta shape functions serve as a good benchmark for assessing the accuracy of the developed methodology as well as its successful implementation. In all of these flows, essentially no discrepancies between the results computed with both types of projection kernels are found, in particular on different mesh topologies and on high-order, high-aspect-ratio elements. The final test two cases entail shock interaction with a spanwise-inhomogeneous cloud of particles and hypersonic flow laden with dust particles over a blunt body.



(a) 3D view of temperature

(b) 3D view of streamwise velocity

(c) Zoomed-in 2D view of temperature

(d) Zoomed-in 2D view of streamwise velocity

Fig. 28. 3D and zoomed-in 2D views of both the carrier and disperse phases for hypersonic flow laden with medium-sized particles over a sphere forebody: (a) 3D view of temperature; (b) 3D view of streamwise velocity; (c) 2D view of temperature; (d) 2D view of streamwise velocity. The particles are indicated by circles scaled in size by particle diameter and colored by the corresponding quantities. Only 0.00625% of all particles is shown.

These flow configurations are characterized by complex physical processes, such as strong shocks, interactions among shocks and other flow discontinuities, hydrodynamic instabilities, and gas-particle momentum and energy exchange. Compared to the first three cases, the particles are larger relative to the mesh elements. Under the conditions considered, the use of delta functions results in noise and numerical instabilities due to the stronger back-coupling. The proposed methodology, on the other hand, yields robust and accurate results with high-order polynomials. Across all test cases, good agreement with experiments, theory, and/or other numerical simulations is found, further confirming the capability of the proposed

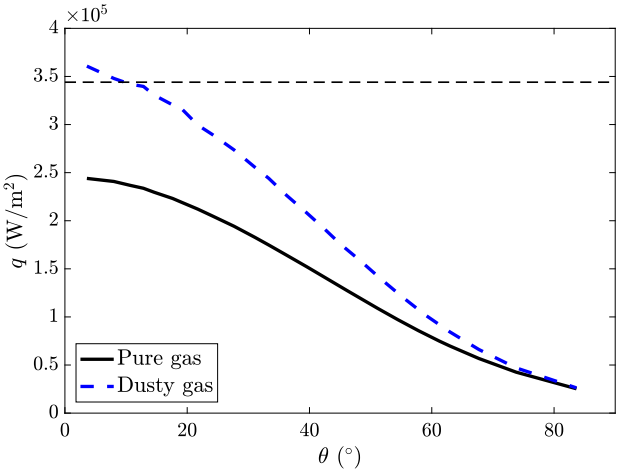


Fig. 29. Pure-gas and dusty-gas surface heat flux profiles for hypersonic flow laden with medium-sized particles over a sphere forebody. The dusty-gas heat fluxes are computed with the proposed smooth anisotropic kernels. The horizontal dashed black line represents the stagnation-point dust-gas heat flux from the corresponding experiment [31]. θ is the polar angle with respect to the $-x$-axis.

reverse-coupling methodology to efficiently compute high-speed particle-laden flows on arbitrary, curved, high-aspect-ratio elements.

CRediT authorship contribution statement

Eric J. Ching: Conceptualization, Methodology, Writing – original draft. **Matthias Ihme:** Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by a Stanford Graduate Fellowship and an Early Career Faculty grant (NNX15AU58G) from the NASA Space Technology Research Grants Program. Resources supporting this work were provided by the NASA High-End Computing (HEC) Program through the NASA Advanced Supercomputing (NAS) Division at Ames Research Center.

References

[1] J. Capecelatro, O. Desjardins, An Euler-Lagrange strategy for simulating particle-laden flows, J. Comput. Phys. 238 (2013) 1–31.
[2] G.B. Jacobs, W.-S. Don, A high-order WENO-Z finite difference based particle-source-in-cell method for computation of particle-laden flows with shocks, J. Comput. Phys. 228 (2008) 1365–1379.
[3] G.B. Jacobs, J.S. Hesthaven, High-order nodal discontinuous Galerkin particle-in-cell method on unstructured grids, J. Comput. Phys. 214 (2006) 96–121.
[4] David Zwick, S. Balachandar, A scalable Euler–Lagrange approach for multiphase flow simulation on spectral elements, Int. J. High Perform. Comput. Appl. 34 (3) (2020) 316–339.
[5] C.T. Crowe, Review—numerical models for dilute gas-particles flows, J. Fluids Eng. 104 (1982) 297–303.
[6] K.D. Squires, J.K. Eaton, Particle response and turbulence modification in isotropic turbulence, Phys. Fluids A, Fluid Dyn. 2 (7) (1990) 1191–1203.
[7] S. Elghobashi, G.C. Truesdell, On the two-way interaction between homogeneous turbulence and dispersed solid particles. I: Turbulence modification, Phys. Fluids A, Fluid Dyn. 5 (7) (1993) 1790–1801.
[8] D.Z. Huang, P.-O. Persson, M.J. Zahr, High-order linearly stable, partitioned solvers for general multiphysics problems based on implicit–explicit Runge–Kutta schemes, Comput. Methods Appl. Mech. Eng. 346 (2019) 674–706.
[9] K. Sengupta, B. Shotorban, G.B. Jacobs, F. Mashayek, Spectral-based simulations of particle-laden turbulent flows, Int. J. Multiph. Flow 35 (2009) 811–826.
[10] G. Akiki, W.C. Moore, S. Balachandar, Pairwise-interaction extended point-particle model for particle-laden flows, J. Comput. Phys. 351 (2017) 329–357.
[11] Z.J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H.T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, M. Visbal, High-order CFD methods: Current status and perspective, Int. J. Numer. Methods Fluids 72 (2013) 811–845.
[12] E.J. Ching, S.R. Brill, M. Barnhardt, M. Ihme, A two-way-coupled Euler-Lagrange method for simulating multiphase flows with discontinuous Galerkin schemes on arbitrary curved elements, J. Comput. Phys. 405 (2020).
[13] D. Zwick, S. Balachandar, Dynamics of rapidly depressurized multiphase shock tubes, J. Fluid Mech. 880 (2019) 441–477.
[14] A. Stock, J. Neudorfer, M. Riedlinger, G. Pirrung, G. Gassner, R. Schneider, S. Roller, C.-D. Munz, Three-dimensional numerical simulation of a 30-GHz gyrotron resonator with an explicit high-order discontinuous-Galerkin-based parallel particle-in-cell method, IEEE Trans. Plasma Sci. 40 (7) (2012) 1860–1870.

[15] M. Pfeiffer, F. Hindenlang, T. Binder, S.M. Copplestone, C-D. Munz, S. Fasoulas, A particle-in-cell solver based on a high-order hybridizable discontinuous Galerkin spectral element method on unstructured curved meshes, Comput. Methods Appl. Mech. Eng. 349 (2019) 149–166.
[16] K.J. Fidkowski, A simplex cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations, PhD thesis, Massachusetts Institute of Technology, 2007.
[17] T.A. Oliver, A high-order, adaptive, discontinuous Galerkin finite element method for the Reynolds-Averaged Navier-Stokes equations, PhD thesis, Massachusetts Institute of Technology, 2008.
[18] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (2) (1981) 357–372.
[19] F. Bassi, S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations, in: B. Cockburn, C.-W. Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, Springer, Berlin, 2000, pp. 197–208.
[20] E.J. Ching, Y. Lv, P. Gnoffo, M. Barnhardt, M. Ihme, Shock capturing for discontinuous Galerkin methods with application to predicting heat transfer in hypersonic flows, J. Comput. Phys. 376 (2019) 54–75.
[21] C.T. Crowe, J.D. Schwarzkopf, M. Sommerfeld, Y. Tsuji, Multiphase Flows with Droplets and Particles, CRC Press, 2011.
[22] A. Allievi, R. Bermejo, A generalized particle search-locate algorithm for arbitrary grids, J. Comput. Phys. 132 (1997) 157–166.
[23] D. Dunavant, High degree efficient symmetrical Gaussian quadrature rules for the triangle, Int. J. Numer. Methods Eng. 21 (1985) 1129–1148.
[24] J. Marcon, A. Garai, M. Denison, S.M. Murman, An adjoint elasticity solver for high-order mesh deformation, in: AIAA Scitech 2021 Forum, 2021.
[25] X. Pennec, P. Fillard, N. Ayache, A Riemannian framework for tensor computing, Int. J. Comput. Vis. 66 (1) (2006) 41–66.
[26] K. Fidkowski, A local sampling approach to anisotropic metric-based mesh optimization, in: 54th AIAA Aerospace Sciences Meeting, 2016, AIAA Paper 2016-0835.
[27] G.S. Shallcross, R.O. Fox, J. Capecelatro, A volume-filtered description of compressible particle-laden flows, Int. J. Multiph. Flow 122 (2020).
[28] K. Zhai, T. Banerjee, D. Zwick, J. Hackl, S. Ranka, Dynamic load balancing for compressible multiphase turbulence, in: Proceedings of the 2018 International Conference on Supercomputing, 2018, pp. 318–327.
[29] B.Y. Wang, I.I. Glass, Compressible laminar boundary-layer flows of a dusty gas over a semi-infinite flat plate, J. Fluid Mech. 186 (1988) 223–241.
[30] V.M. Boiko, V.P. Kiselev, S.P. Kiselev, A.N. Papyrin, S.V. Poplavsky, V.M. Fomin, Shock wave interaction with a cloud of particles, Shock Waves 7 (5) (1997) 275–285.
[31] E.B. Vasilevskii, A.N. Osiptsov, A.V. Chirikhin, L.V. Yakovleva, Heat exchange on the front surface of a blunt body in a high-speed flow containing low-inertia particles, J. Eng. Phys. Thermophys. 74 (6) (2001) 1399–1411.
[32] E. Vasilevskii, A. Osiptsov, Experimental and numerical study of heat transfer on a blunt body in dusty hypersonic flow, in: 33rd AIAA Thermophysics Conference, Norfolk, Virginia, 1999, AIAA Paper 1999-3563.
[33] E. Ching, M. Ihme, Sensitivity study of high-speed dusty flows over blunt bodies simulated using a discontinuous Galerkin method, in: AIAA Scitech 2019 Forum, San Diego, California, 2019, AIAA Paper 2019-0895.
[34] Y.S. Touloukian, E.H. Buyco, Thermophysical Properties of Matter - The TPRC Data Series. Volume 5. Specific Heat-Nonmetallic Solids, Technical report, Thermophysical and Electronic Properties Information Analysis Center, Lafayette, IN, 1970.
[35] M.W. Chase Jr., NIST-JANAF Thermochemical Tables, fourth edition, Journal of Physical and Chemical Reference Data Monograph, vol. 9, 1998.
[36] C.B. Henderson, Drag coefficients of spheres in continuum and rarefied flows, AIAA J. 14 (6) (1976) 707–708.
[37] T.W. Fox, C.W. Rackett, J.A. Nicholls, Shock wave ignition of magnesium powers, in: 11th International Shock Tubes and Waves Symposium, Seattle, Washington, 1978, pp. 262–268.
[38] A.L. Stasenko, Velocity recovery factors of a particle repelled from a solid surface, J. Eng. Phys. Thermophys. 80 (5) (2007) 885–891.
[39] V.P. Kiselev, S.P. Kiselev, E.V. Vorozhtsov, Interaction of a shock wave with a particle cloud of finite size, Shock Waves 16 (1) (2006) 53.
[40] C.L.M. Snow, The Heat Capacity and Thermodynamic Properties of the Iron Oxides and Their Relation to the Mineral Core of the Iron Storage Protein Ferritin, PhD thesis, Brigham Young University, 2010.