Numerical assessment of the two-point statistical equations in liquid/gas flows

Fabien Thiesset, Alexandre Poux

Technical Report

Cite this article: F. Thiesset, A Poux. (2020). Numerical assessment of the two-point statistical equations in liquid/gas flows. [Technical Report] CNRS, Normandy Univ., INSA Rouen, CORIA. <hal-02614565>

Subject Areas:
Fluid Mechanics, Computational physics

Keywords:
Two-phase flows, Numerical Simulations, Two-point statistics

Author for correspondence:
Fabien Thiesset
e-mail: fabien.thiesset@coria.fr
Alexandre Poux
e-mail: alexandre.poux@coria.fr

Numerical assessment of the two-point statistical equations in liquid/gas flows

F. Thiesset, A. Poux

CNRS, Normandy Univ., UNIROUEN, INSA Rouen, CORIA, 76000 Rouen, France

May 2020

The transport equation for the two-point statistics of the liquid phase was first introduced by Thiesset et al. [1] and employed to characterize the transport of liquid in homogeneous decaying turbulence. The same framework was then applied to a liquid/gas shear flow by Thiesset et al. [2] and Thiesset et al. [3]. In these situations, the two-point statistics of the liquid phase is appraised from Direct Numerical Simulations data (DNSs) using the `archer` code and the `increments` library of `pyarcher` (the python platform for pre- and post-processing `archer` data). The present paper aims at documenting the appropriateness of such numerical tools on some validation test cases. Light is shed onto *(i)* the effect of the numerical resolution, *(ii)* the type of scalar (liquid-volume-fraction LVF *vs* colour-function CF) which is employed as representative of the liquid phase and *(iii)* the appropriateness of some numerical strategies for computing angular or spherical averages.

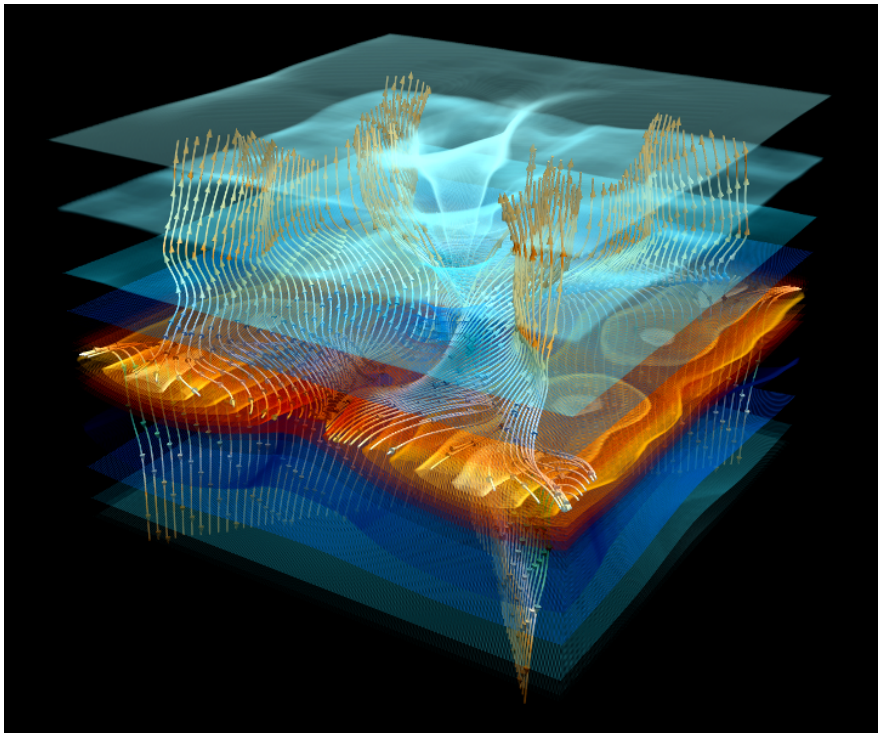

Illustration of the 3D subset of a 4D structure function in a liquid-gas shear flow [3]

Contents

List of Figures

1. Context and objectives

1.1. Two-point budget of the liquid phase

The analytical framework describing the scale/space/time transport of the liquid phase in two-phase flows is based on the transport equation for the second-order structure function:

$$\begin{aligned}\left\langle (\delta\phi)^2 \right\rangle_{\mathbb{E}} &= \left\langle \left[\phi(\boldsymbol{x}^+) - \phi(\boldsymbol{x}^-)\right]^2 \right\rangle_{\mathbb{E}} \\ &= \left\langle \left[\phi\left(\boldsymbol{X} + \frac{\boldsymbol{r}}{2}\right) - \phi\left(\boldsymbol{X} - \frac{\boldsymbol{r}}{2}\right)\right]^2 \right\rangle_{\mathbb{E}} \end{aligned} \tag{1.1}$$

where ϕ will be used to designate either the liquid-volume-fraction (LVF) or colour-function field (CF), the brackets denote average and the subscript $\mathbb{E}$ indicates an ensemble average. If applicable, ensemble averages can be replaced by spatial averages over homogeneity directions. The mid-point is defined by $\boldsymbol{X} = (\boldsymbol{x}^+ + \boldsymbol{x}^-)/2$ and the separation vector $\boldsymbol{r} = (\boldsymbol{x}^+ - \boldsymbol{x}^-)$ [6].

The transport equation for $\left\langle (\delta\phi)^2 \right\rangle_{\mathbb{E}}$ writes

$$\partial_t \left\langle (\delta\phi)^2 \right\rangle_{\mathbb{E}} = -\boldsymbol{\nabla}_{\boldsymbol{r}} \cdot \boldsymbol{\Phi}_{\boldsymbol{r}} - \boldsymbol{\nabla}_{\boldsymbol{X}} \cdot \boldsymbol{\Phi}_{\boldsymbol{X}} \tag{1.2}$$

where the two terms on the right-hand-side of Eq. (1.2) are two transfer terms in the combined space of scales $\boldsymbol{r}$ and geometrical positions $\boldsymbol{X}$, and read as the divergence of the flux $\boldsymbol{\Phi}_{\boldsymbol{r}}$ and $\boldsymbol{\Phi}_{\boldsymbol{X}}$ in scale and physical space, respectively:

$$\boldsymbol{\Phi}_{\boldsymbol{r}} = \left\langle (\delta\boldsymbol{u}) (\delta\phi)^2 \right\rangle_{\mathbb{E}} \tag{1.3a}$$

$$\boldsymbol{\Phi}_{\boldsymbol{X}} = \left\langle (\sigma\boldsymbol{u}) (\delta\phi)^2 \right\rangle_{\mathbb{E}} \tag{1.3b}$$

Further details on the derivations of Eq. (1.2) and the physical interpretation of the different terms of this equation are given by Thiesset et al. [1]. Note that Eq. (1.2) pertain to the total liquid field, but Refs. [1,3] also discussed the equations for the fluctuating part $\phi' = \phi - \langle\phi\rangle_{\mathbb{E}}$.

1.2. Recent changes

In Refs. [1–3], special care was given to the accuracy with which the two-point budget Eq. (1.2) was closed. This was considered as a stringent test of the appropriateness of both the numerical simulations and of the post-processing procedures. However, between Thiesset et al. [1,2] and Thiesset et al. [3] there were few changes in the usage of two-point statistical equations. These differences are listed below:

- in Refs. [1,2], the equations were tested for LVF field while in Ref. [3], we considered the CF which is constructed in `pyarcher` from the excursion set: level-set$(\boldsymbol{x}) > 0$. The latter can take only 0 or 1 values while the former can take any value between 0 and 1 in cells containing an interface. When Ref. [1] was peer-reviewed, one of the reviewer pointed out that:

  "*the authors base their analysis on a liquid volume fraction. The potential issue with this start is that the liquid volume fraction is by definition a quantity associated with a sampling volume. Why not start the analysis with a color function instead, which is a point quantity?*"

  We replied that:

  "*mathematically, the difference between the LVF and the phase indicator function is perceptible only in cells containing an interface since they both are equal to 1 (0) in the liquid (gas) phase. In addition, when the mesh cell goes to zero, the LVF tends to the colour function. Hence, it is expected that for sufficiently small mesh size, the budgets for the LVF and the colour function are the same.*"

  and we provided a figure showing that the second-order structure function of the LVF and CF field were very similar and conclude that the dependence to the mesh size was not significant. In Ref. [3], we employed the CF field instead because we made comparisons with some theoretical predictions [4,5,7]. These were developed by the community of porous media for which heterogeneous materials are characterized through their CF field and not their LVF field.

- Another difference between Refs. [1,2] and Ref. [3] is the numerical procedure for computing spherical or angular averages. This will be discussed in great details in §3.

1.3. Objectives of the present work

The objectives of the present technical report are the following:

- Firstly, in §2, a presentation of the `archer` solver and the `pyarcher.increments` library is provided.

- Secondly, §3 aims at discussing the appropriateness of the two different numerical strategies for computing angular or spherical averages.
- Thirdly, although the LVF field asymptotes the CF field for infinitely small grid spacing, it is worth documenting the dependence to the mesh size of two-point statistics for finite resolutions. This is tackled in §4.
- Fourthly, the order of convergence of the two-point budget with respect to the mesh size for both the LVF and CF fields is presented in §5.

2. The `archer` solver and the `pyarcher.increments` library

Before proceeding with the validation of the numerical tools used to compute the two-point budget Eq. (1.2), we start by providing a short description of the `archer` code, the `pyarcher.increments` library and its main functionalities.

2.1. The `archer` solver

We use the High-Performance-Computing code `archer` developed at the CORIA laboratory. It was one of the first code worldwide, undertaking the simulation of liquid-jet atomization under a realistic diesel injection configuration [8]. It solves on a staggered Cartesian mesh the one-fluid formulation of the incompressible Navier-Stokes equation, viz.

$$\partial_t\, \rho\boldsymbol{u} + \boldsymbol{\nabla}\cdot(\rho\boldsymbol{u}\otimes\boldsymbol{u}) = -\boldsymbol{\nabla}p + \boldsymbol{\nabla}\cdot(2\mu\mathbf{D}) + \boldsymbol{f} + \gamma\kappa\delta_s\boldsymbol{n} \tag{2.1}$$

where p is the pressure field, $\mathbf{D}$ the strain rate tensor, $\boldsymbol{f}$ a source term, μ the dynamic viscosity, ρ the density, γ the surface tension, $\boldsymbol{n}$ the unit normal vector to the liquid-gas interface, κ its mean curvature and δ_s is the Dirac function characterizing the locations of the liquid gas interface. For solving Eq. (2.1), the convective term is written in conservative form and solved using the improved Rudman's technique [9] presented in Ref. [10]. The latter allows mass and momentum to be transported in a consistent manner thereby enabling flows with large liquid/gas density ratios to be simulated accurately. The viscosity term is computed following the method presented by Ref. [11]. To ensure incompressibility of the velocity field, a Poisson equation is solved. The latter accounts for the surface tension force and is solved using a MultiGrid preconditioned Conjugate Gradient algorithm (MGCG) [12] coupled with a Ghost-Fluid method [13].

For transporting the interface, use is made of a coupled level-set and volume-of-fluid (CLSVOF) solver, in which the level-set function accurately describes the geometric features of the interface (its normal and curvature) and the volume-of-fluid function ensures mass conservation. The density is calculated from the volume-of-fluid ψ ($\equiv$ LVF) as $\rho = \rho_L\psi + \rho_G(1-\psi)$, where ρ_L, ρ_G is the density of the liquid and gas phase. The dynamic viscosity (μ_L or μ_G) depends on the sign of the level-set function. In cells containing both a liquid and gas phase, a specific treatment is performed to evaluate the dynamic viscosity, following the procedure of Ref. [11]. For more information about the `archer` solver, the reader can refer to Refs. [8,10,14].

2.2. Reading and pre-processing `archer` files

Results from `archer` simulations are written `hdf5` format with one time series per processor and multiple time-step per file. For visualization purposes using e.g. `paraview`, `archer` further writes `xdmf` files which also contain informations about the geometry (e.g. origin, grid spacing) and topology (connectivity between `MPI` blocks) of the simulation domain.

`pyarcher` reads such data using the `dask` library and store them in a `xarray.Dataset`. All the time-steps and all the processors are merged into a 4D array, with coordinates `"x", "y", "z", "t"`. These coordinates can be either specified by the user or read in the `xdmf` files. The minimal blocks (one processor and one time-step) are called chunks and are the unit of work for `dask`. We make extensive usage of the "lazy" approach of `dask`. More on this aspect is given in the subsection 2.3.

`archer` is a staggered CFD code following the MAC discretization. For some post-computations (e.g. the two-point statistical equations), the velocity field and scalar field (e.g. the volume-of-fluid, the level-set) should be known at the same point. This is not initially the case due to the staggered nature of the `archer` mesh. In this situation, one needs to interpolate the velocity field at the center of the (cubic) cells. In `pyarcher`, this is done with the `interp` function of `xarray` which proceeds by linear interpolation. This function has been extended to work with chunked data. It is embedded in the method `.to_cell_center()`.

The data written by `archer` contain ghost cells for boundary conditions and processors interface. The latter are removed during the previously mentioned merging operation, but not the former which can be removed with the method `.without_ghost()`. Boundary conditions (periodic, symmetric) can be prescribed with the method `.add_boundary_conditions()`.

2.3. The `all_increments` pipeline

The computation of two-point statistics can be done in one line of code through the use of the `all_increments` method of the `pyarcher.increments` library:

```
data = increments.all_increments(velocity, scalar, max_separation, inhomogeneity)
```

For the moment, the library can handle 3D budgets [in case in homogeneous fields as in Ref. 1] or 4D budgets [for configurations with 1 direction of inhomogeneity as in Refs. 2,3]. The function `all_increments` takes four input arguments:

- `velocity` $\equiv (u, v, w)$ of type `CenteredVectorField`, is a tuple containing the three velocity components as a function of (x, y, z).
- `scalar` $\equiv \phi$ of type `CenteredScalarField`, is the scalar field used to compute the two-point equations. `scalar` is either the CF ($\equiv$ level-set> 0) or LVF ($\equiv$ volume-of-fluid) field.
- `max_separation` of type `integer` is the maximum value (in terms of number of grid points) for the separation vector $\boldsymbol{r}$
- `inhomogeneity` = {`None`, `"x"`, `"y"`, `"z"`} is the inhomogeneity direction (if any). By default `inhomogeneity` = `None`

The function `all_increments` creates a pipeline constituted of 6 consecutive steps described below:

(i) `increments`, takes as input arguments `velocity`, `scalar`, `max_separation`, `inhomogeneity` and computes the spatially averaged (along homogeneity directions) of all type of two-point quantities (differences, sums, cross-moments, etc) that are needed in the budgets. For computational efficiency, this function uses a `Fortran90` kernel with a memory distributed `openMP` parallelization. It returns a `xarray.Dataset` containing every two-point statistics with some associated keywords, e.g. $\langle(\delta\phi)\rangle \equiv$ `"dphim"`, $\langle(\delta\phi)^2\rangle \equiv$ `"dphi2"`, $\langle\delta u(\delta\phi')^2\rangle \equiv$ `"du_dphip2"`, $\langle\sigma w(\delta\phi)^2\rangle \equiv$ `"sw_dphi2"`, etc. These datasets have coordinates `"rx"`, `"ry"`, `"rz"` $\in \{-$`max_separation`$;+$`max_separation`$\}$. The coordinate `"t"` is also added only if `velocity` and/or `scalar` contain `"t"` as coordinate. If `inhomogeneity` is not `None`, then `"X"` is added as coordinates of the `xarray.Dataset`.

(ii) `flux_velocity`, takes as input argument the output `xarray.Dataset` of the `increments` function described above. The flux velocities are noted $\boldsymbol{\omega_{r,X}}$ and are defined as the ratio of the flux component in one direction of the combined $\{\boldsymbol{r}, \boldsymbol{X}\}$ space divided by $\langle(\delta\phi)^2\rangle$ or $\langle(\delta\phi')^2\rangle$ [see Ref. 1]. This function returns an `xarray.Dataset` containing the flux velocity with associated keywords, e.g. $\omega_{r_x} \equiv$ `"w_rx"`, $\omega'_Y \equiv$ `"wp_Y"`, etc.

(iii) `scalar_potential`, takes as input argument the output `xarray.Dataset` of the `flux_velocity` function described above. This function decomposes the flux velocity field into a solenoidal and non solenoidal velocity components by solving a Poisson equation [see Ref. 1, for more details]. Here again, an `xarray.Dataset` is returned with e.g. the keywords `"w_pot_rx"`, `"wp_sol_Y"`. The returned `xarray.Dataset` also contain the scalar potential field with the keywords `"potential"` or `"potential_p"` for either the total or fluctuating field. The Poisson equation is solved with zero gradient boundary conditions using the `minres` algorithm and a Jacobi preconditioning. If `inhomogeneity` is not `None`, then the decomposition is carried out for all points in the inhomogeneity direction and `"X"` is added as coordinates of the `xarray.Dataset`.

(iv) `budget_terms` takes as arguments the `xarray.Dataset` returned by `increments`, the `xarray.Dataset` returned by `flux_velocity`, the `xarray.Dataset` returned by `scalar_potential` and the argument `inhomogeneity`. It returns a `xarray.Dataset` containing all the possible terms and/or decompositions of the scale budget of the total and/or fluctuating field. These terms are given keywords e.g. `"transfer_r"`, `"transferp_X"`, `"production_r"`, `"production_X"`, `"advection_sol_r"`, `"advection_pot_r"`, etc. The time derivative term is not handled.

(v) `spherical_average`, computes and return the spherical average (described later) of the `xarray.Dataset` returned by `increments` and `budget_terms`. The returned dataset uses the same keywords as in input to which is added the extension `"_sav"`, e.g. `"dphi2_sav"`, `"transfer_r_sav"` and has coordinate `"rn"` $\in \{0:$`max_separation`$\}$. If `inhomogeneity` is not `None`, then the spherical average is computed for all points in the inhomogeneity direction and `"X"` is added as coordinates of the `xarray.Dataset`.

(vi) `angular_average`, computes and return the angular average (described later) of the `xarray.Dataset` returned by `increments` and `budget_terms`. Here the extension `"_aav"` is added, e.g. `"dphi2_aav"`, `"transfer_r_aav"` and the returned `xarray.Dataset` has coordinate `"rn"` $\in \{0:$`max_separation`$\}$. If `inhomogeneity` is not `None`, then the angular average is computed for all points in the inhomogeneity direction and `"X"` is added as coordinates of the `xarray.Dataset`.

`all_increments` returns a merged `xarray.Dataset` containing all aforementioned datasets with their respective coordinates.

2.4. The `dask`'s delayed method and the `load` function

Because it is generally not necessary to compute all aforementioned quantities, the `all_increments` pipeline described above is tackled in lazy mode. This means that the dependence between variables is created but these variables are neither computed nor loaded into memory. This has been done using `dask`'s delayed functions.

This is a very beneficial feature of the `dask` library that we use extensively throughout the library for managing the data. This is completely transparent to the user as it is still possible to ignore completely the laziness aspect of the library because e.g. a `plot` triggers the computation.

However, there is no caching mechanism. For illustrating this, let us take a simple example of two arbitrary variables `B` and `C` that depend on a variable `A`. Plotting `B` triggers the computation of `A` then `B`. Similarly, plotting `C` requires computing `A` then `C`. In this example. `A` is thus computed twice. In order to avoid this, the `load` function that can handle multiple variables have been added to `pyarcher`. This function specifically aims at precluding computing several times a given variable.

When applied to the `increments` library, if one wants to compute e.g. the spherical average and angular averages of $\langle(\delta\phi)^2\rangle$, one simply call the function

```
load(data.dphi2_sav, data.dphi2_aav)
```

In this example, $\langle(\delta\phi)^2\rangle$ will first be computed through `increments` and then it will proceed to the spherical and angular average through `spherical_average` and `angular_average`, respectively. Here, $\langle(\delta\phi)^2\rangle$ is computed only once.

2.5. The `write` & `read` functionalities

Results from the `pyarcher.increments` library can be saved in `hdf5` format by use of the function `pyarcher.increments.write(hdf5_file, data, **kwargs)`. It is worth stressing that only the quantities that have been loaded by the `load` function will be saved. The coordinates `{"rx", "ry", "rz"}` will also be saved together with `{"rn", "t", "X"}` if applicable. Any scalar value can also be written using user-defined `**kwargs`. This can be convenient to save e.g. the grid spacing, the surface density, or any other relevant quantity.

The `pyarcher.increments` library also contains a function `read(hdf5_file, option)`. This function allows one to read the data that have been written by the `pyarcher.increments.write` function. It proceeds in recasting the data in the same form as the original `xarray.Dataset`. It has two options. `option="read_only"` which is faster, recreates an `xarray.Dataset` containing only the variables saved in the `hdf5_file`. The `option="post_pro"` resets the full pipeline of `all_increments` and replaces the variables by that saved in the `hdf5_file`. The latter option is made for allowing to post-compute some quantities. For instance, if the three components of $\boldsymbol{\Phi_r}$ were saved in the `hdf5_file`, then one can easily post-compute the angularly averaged r-transfer term by the commands:

```
data = increments.read(hdf5_file, option="post_pro")
load(data.transfer_r_aav)
```

3. Computation of spherical and angular averages

3.1. Definition

The angular average is noted $\langle\bullet\rangle_\Omega$ and is defined by [3]:

$$\langle\bullet\rangle_\Omega = \frac{1}{4\pi}\iint_\Omega \bullet \,\sin\theta \mathrm{d}\theta\mathrm{d}\varphi \tag{3.1}$$

where the set of solid angles $\Omega = \{\varphi, \theta \mid 0 \leq \varphi \leq \pi, 0 \leq \theta \leq 2\pi\}$ with $\varphi = \arctan(r_y/r_x)$ and $\theta = \arccos(r_z/|\boldsymbol{r}|)$. The angular average is related to the spherical average within a sphere of radius $r = |\boldsymbol{r}|$, noted $\langle\bullet\rangle_\mathbb{S}$ and defined by [1,6]:

$$\langle\bullet\rangle_\mathbb{S} = \frac{3}{4\pi r^3}\iiint_\mathbb{S} \bullet\, r^2 \sin\theta \mathrm{d}\theta\mathrm{d}\varphi\mathrm{d}r = \frac{3}{r^3}\int_0^r \langle\bullet\rangle_\Omega\, r^2 \mathrm{d}r \tag{3.2}$$

Note that the angular average relies on a double integral while the spherical average necessitates integrating the three directions of the separation vector written in spherical coordinates.

In Refs. [1,2], we used the `RegularGridInterpolator` function of the python `scipy` library for linearly interpolating two-point statistics from the Cartesian (r_x, r_y, r_z) mesh to the spherical mesh (r, θ, ϕ). Integration over (r, θ, ϕ) is then done using the `trapz` function of `scipy`. In Ref. [3], we realized that at small scales where the angular dependence of two-point statistics is limited to few points, this method is not very accurate. We then replaced this

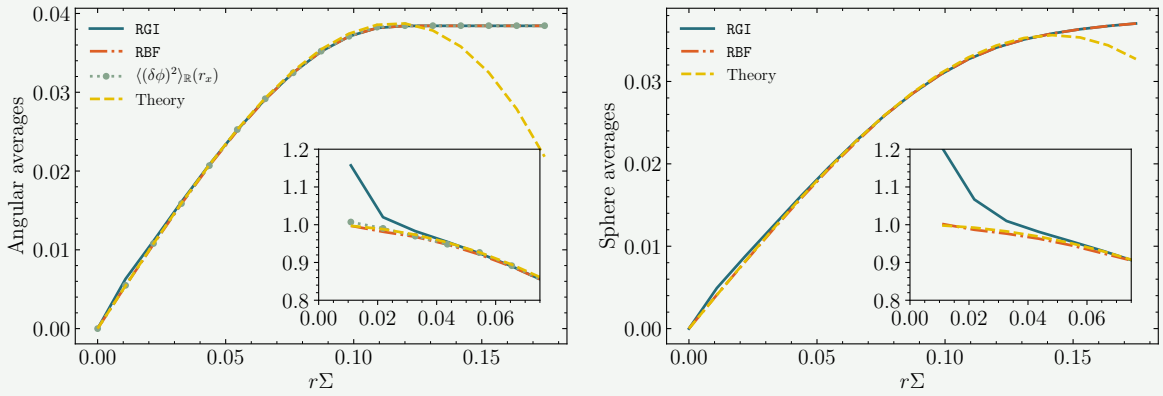

Figure 1. Comparison of different methods for computing angular (left) and spherical (right) averages, using `RegularGridInterpolator + trapz` (abbreviated by `RGI`) or the `Rbf + dblquad` or `tplquad` (abbreviated by `RBF`). Angular and spherical averages are compared to theoretical predictions of Refs. [4,5]. The angular average (left) is also compared to a cut of the 3D structure function along the $r_x, r_y = r_z = 0$ axis. The insets show the second-order structure function normalized by $\Sigma r/2$ (left) and $3\Sigma r/8$ (right)

procedure by using the Radial Basis Function `Rbf` interpolation of `scipy` and integration is performed using `dblquad` (angular average) or `tplquad` (spherical average).

3.2. Description of the validation test case

With the aim of comparing the performance of the two aforementioned procedures for computing spherical and angular averages, we use the following benchmark:

- A level-set function field representing a sphere of radius $R_s = 10 + 2/3$ is set. The numerical domain contains $N_x \times N_y \times N_z = 64^3$ points and the grid spacing is $\Delta x = \Delta y = \Delta z = 1$. The sphere is placed at the center of the numerical domain.
- The `increments` library of `pyarcher` is then used to calculate the 3D second-order structure function of the CF field, the latter being defined by the excursion set: level-set$(\boldsymbol{x}) > 0$. A spatial average is performed, i.e. two-point statistics are averaged over $\mathbb{R} \in \{x, y, z\}$. Given the geometry, two-point statistics are isotropic and thus $\langle(\delta\phi)^2\rangle_{\mathbb{R}}(\boldsymbol{r}) = \langle(\delta\phi)^2\rangle_{\mathbb{R}}(r = |\boldsymbol{r}|)$.
- `pyarcher.increments` further allows us computing the spherical and angular averages using either the `RegularGridInterpolator + trapz` (abbreviated by `RGI`) or `Rbf + dblquad` or `tplquad` (abbreviated by `RBF`). For the `Rbf` method, use was made of a linear basis function, and for the `RegularGridInterpolator` linear interpolation was used. Here, isotropy yields $\langle(\delta\phi)^2\rangle_{\mathbb{R}}(r = |\boldsymbol{r}|) = \langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}(r)$ which is convenient for assessing the accuracy of the angular average procedure.
- Angular and spherical averages are further compared to the theoretical prediction of Refs. [4,5], which writes

$$\langle(\delta\phi)^2\rangle_{\Omega} = \frac{\Sigma r}{2}\left[1 - \frac{r^2}{8}\left(\langle\mathcal{H}\rangle_0 - \frac{\langle\mathcal{G}\rangle_0}{3}\right)\right] \tag{3.3}$$

  where $\mathcal{H}$ and $\mathcal{G}$ are the mean and Gaussian curvatures, Σ is the interface surface area divided by the total (liquid+gas) volume and $\langle\bullet\rangle_0$ denotes the area weighted average. For a sphere $\mathcal{H}^2 = \mathcal{G} = R_s^{-2}$. By virtue of Eq. (3.2), one can readily write

$$\langle(\delta\phi)^2\rangle_{\mathbb{S}} = \frac{3\Sigma r}{8}\left[1 - \frac{r^2}{12}\left(\langle\mathcal{H}\rangle_0 - \frac{\langle\mathcal{G}\rangle_0}{3}\right)\right] \tag{3.4}$$

3.3. Results

Results are presented in Fig. 1. The difference between the two numerical procedures is perceptible only at small scales where the angular dependence of $\langle(\delta\phi)^2\rangle_{\mathbb{R}}$ is limited to few points in the $\boldsymbol{r}$-space. The `RGI` method deviates from theoretical expectations while the `RBF` behaves very nicely irrespectively of the probed separation r. In Fig. 1 (left), it is observed that the `RBF` angular average collapses perfectly onto a cut of $\langle(\delta\phi)^2\rangle_{\mathbb{R}}$ along the axis $(r_x, r_y = r_z = 0)$ which further validates the appropriateness of this procedure.

This has important consequences when estimating the surface density, the mean and Gaussian curvatures from the limit to small separations of $\langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}$ or $\langle(\delta\phi)^2\rangle_{\mathbb{R},\mathbb{S}}$. The insets show the second-order structure function normalized by $\Sigma r/2$ (left) and $8\Sigma r/3$ (right). Given Eq. (3.3) and (3.4), one should expect such normalized structure functions to

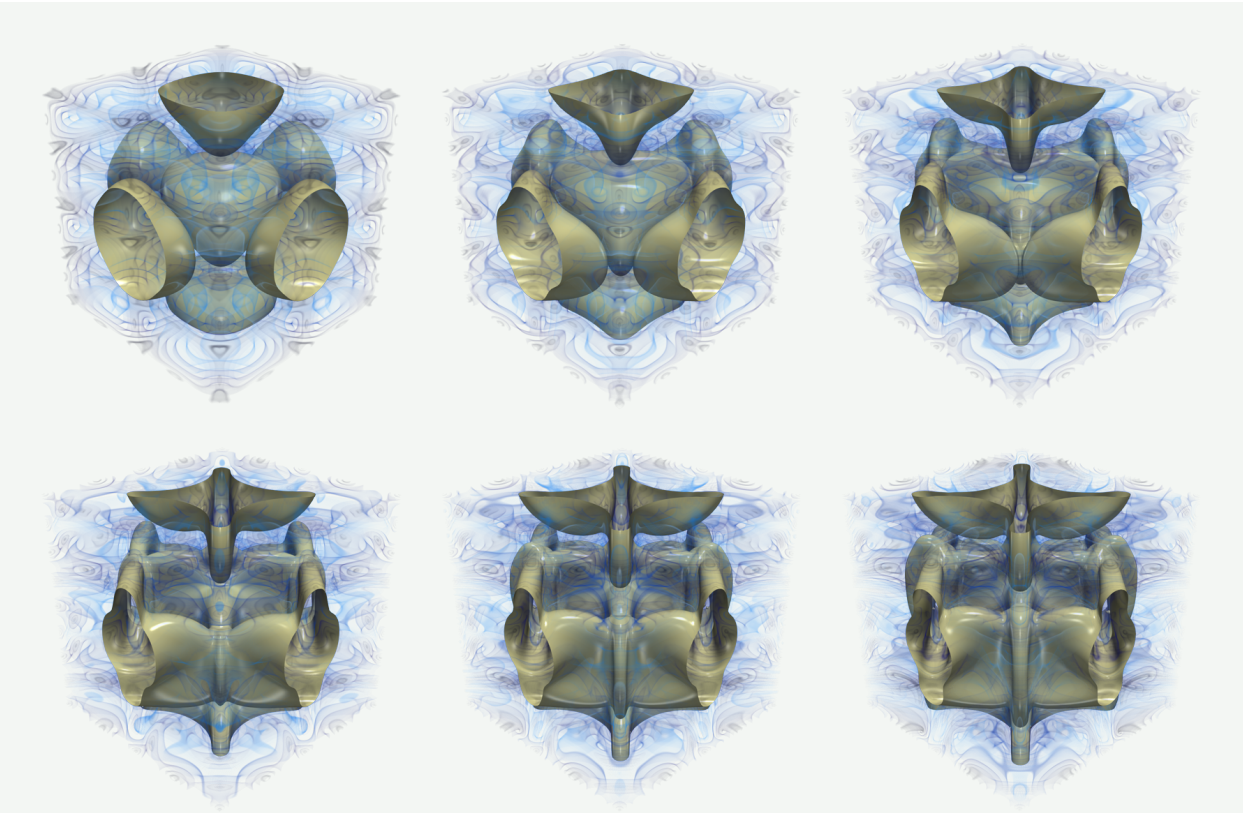

Figure 2. Simulation using the `archer` code of some liquid droplets evolving in a Taylor-Green flow. From top left to bottom right, $t = 5$ μs to $t = 30$ μs. Snapshots are separated by $t = 5$ μs.

approach a value of 1 when r goes to zero. It is observed that while the `RBF` method is very accurate, the error made on the surface density using `RGI` is about 15-20%. The two-procedures yield however very similar values at intermediate and large scales.

Although the `RBF` method appears much more accurate, it requires a larger computational effort. For some situations, when the $\boldsymbol{r}$-space was discretized using more than 64^3 points, the program failed due to memory limitations. For overcoming this issue, we employed a blended version of the two aforementioned approaches, where at small scales (i.e. $|\boldsymbol{r}| \leq 12\Delta x$) the `RBF` method is used while the `RGI` applies to larger scales. Note also that spherical averages are performed by integrating over r, φ, θ (hence `tplquad`) while angular averages only require integration over φ, θ (`dplquad`). Therefore, angular averages are more efficient in terms of computational time and should be preferred over spherical averages when large datasets are to be considered.

4. LVF vs CF fields

4.1. Description of the validation test case

For assessing the convergence of the two-point budget of the LVF and CF fields with respect to the mesh size, `archer` is used to simulate the configuration of 4 initially spherical droplets evolving in a Taylor-Green type of flow. The velocity field is initialized as follows

$$u(x,y,z) = +U\cos(4\pi\tilde{x})\sin(2\pi\tilde{y})\sin(2\pi\tilde{z}) \tag{4.1}$$

$$v(x,y,z) = -U\sin(4\pi\tilde{x})\cos(2\pi\tilde{y})\sin(2\pi\tilde{z}) \tag{4.2}$$

$$w(x,y,z) = -U\sin(4\pi\tilde{x})\sin(2\pi\tilde{y})\cos(2\pi\tilde{z}) \tag{4.3}$$

Here $(\tilde{x},\tilde{y},\tilde{z}) = (x/L_x, y/L_y, z/L_z)$, and we chose $L_x = L_y = L_z = 300$ μm and U was set to 4 m.s^{-1}. For the liquid phase, 4 droplets of radius 70 μm are placed within the domain with 3 located in the center of each face and one located in the center of the domain. Triply periodic boundary conditions are employed. The liquid (gas) density is $\rho_L = 753.0$ kg.m^{-3} ($\rho_G = 25.0$ kg.m^{-3}) and dynamic viscosity $\mu_L = 5.65\,10^{-4}$ Pa.s ($\mu_G = 1.879\,10^{-5}$ Pa.s). The surface tension of the liquid/gas interface is 0.0135. Different numerical resolutions were considered, i.e. $N_x \times N_y \times N_z = 48^3,\ 64^3,\ 96^3,\ 128^3$ & 192^3. The grid spacing $\Delta x = \Delta y = \Delta z$ decreases accordingly. In order to compute the time derivative in Eq. (1.2), we saved the simulation results every 0.5 μs.

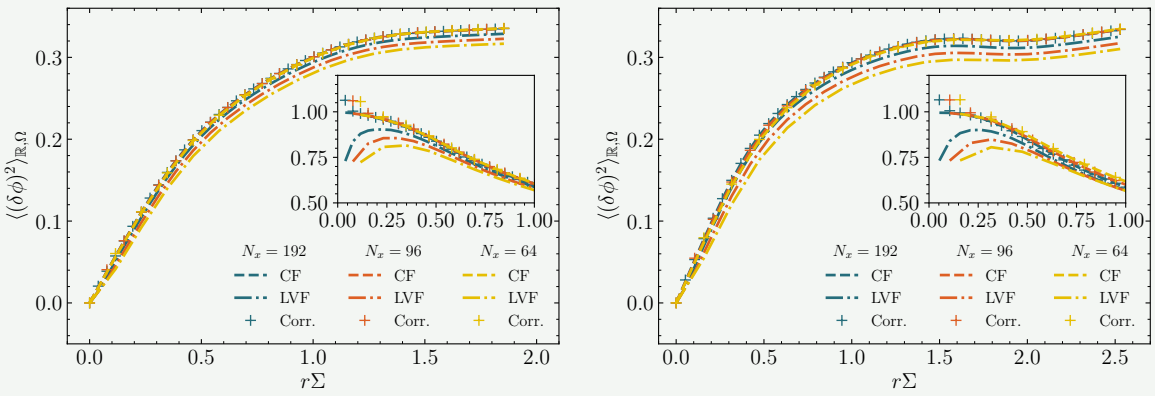

Figure 3. Angularly averaged second-order structure functions of the CF, LVF and corrected LVF fields at $t = 15\ \mu$s (left) and $t = 30\ \mu$s (right) for three spatial resolutions $64^3, 96^3, 192^3$. The inset shows the second-order structure function divided by $\Sigma r/2$.

Typical snapshots of the simulation results using 128^3 points are presented in Fig. 2. These are separated by $t = 5\ \mu$s. Fig. 2 shows that the four droplets progressively deform and become more and more elongated. The surface area of the liquid-gas interface increases with time. At time $t \geq 15\ \mu$s, the four droplet also merge. As a consequence, this configuration is archetypal of the mechanisms which are generally at play in turbulent two-phase flows. It thus appears as a relevant and reproducible test case for assessing the convergence of the two-point budgets w.r.t the mesh size.

4.2. Second-order structure function

We start by investigating the effect of mesh resolution on the second-order statistics of the CF and LVF fields. Results are presented in Fig. 3 where $\langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}$ (here ϕ is used interchangeably for either the LVF or CF field) is plotted against r for different numerical resolutions ($N_x = N_y = N_z = 64,\ 96,\ 192$). The separation r is normalized by the surface density Σ which is estimated using the highest numerical resolution (192^3), using the routines of Refs. [15,16]. These results are obtained at $t = 30\ \mu$s

We observe that $\langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}$ of the CF field is independent of the mesh resolution. On the other hand, $\langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}$ reveals a substantial dependence to the mesh size. Note also that the difference between the CF and LFV field is more pronounced at $t = 30\mu$s than at $t = 15\mu$s suggesting that the second-order structure function of the LVF field is influenced by both the grid spacing Δx and the surface density Σ. Note that, as expected, the difference between the LVF and CF fields tends to zero when the mesh resolution increases, and thus:

$$\lim_{\Delta x \to 0} \langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}\Big|_{\text{LVF}} = \langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}\Big|_{\text{CF}} \tag{4.4}$$

The second-order structure function of the LVF field can be expressed in terms of that of the CF field. For this purpose, we drew inspiration from the result of Ref. [17] for filtered telegraphic signals, and ended up with

$$\langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}\Big|_{\text{LVF}} = \frac{\langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}\Big|_{\text{CF}} - a\Sigma F(r)}{1 + (a\Sigma)^2} \tag{4.5}$$

where $a = \Delta x/3$ and $F(r) = \left[1 + \frac{a}{r}\exp(1 - r/a)\right]$. Eq. (4.5) provides the expected asymptotic regime Eq. (4.4) when $\Delta x \to 0$. Hence, we can now correct the second-order structure function of the LVF field from the bias associated with finite resolutions by inverting Eq. (4.5), yielding

$$\langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}\Big|_{\text{Corr.}} = \frac{\langle(\delta\phi)^2\rangle_{\mathbb{R},\Omega}\Big|_{\text{LVF}} + a\Sigma F(r)}{1 + (a\Sigma)^2} \tag{4.6}$$

Fig. 3 shows the appropriateness of the correction Eq. (4.6). Results are portrayed as symbols and are labelled by "Corr". The corrected LVF structure functions collapse perfectly well with those of the CF field whatever the spatial resolution Δx and surface density Σ. Eq. (4.6) is likely to be demonstrated on some mathematical grounds using geometrical arguments, but this is much beyond the scope of the present technical report.

4.3. Budget of the LVF vs CF field

We now proceed with the estimation of the angularly averaged budgets (using the blended `RBF/RGI` method) of the CF and LVF fields using different spatial resolution. We use triply periodic boundary conditions, hence the X-transfer term vanishes, i.e. $-\langle\boldsymbol{\nabla}_{\boldsymbol{X}} \cdot \boldsymbol{\Phi}_{\boldsymbol{X}}\rangle_{\mathbb{R}} = 0$ [see Ref. 1].

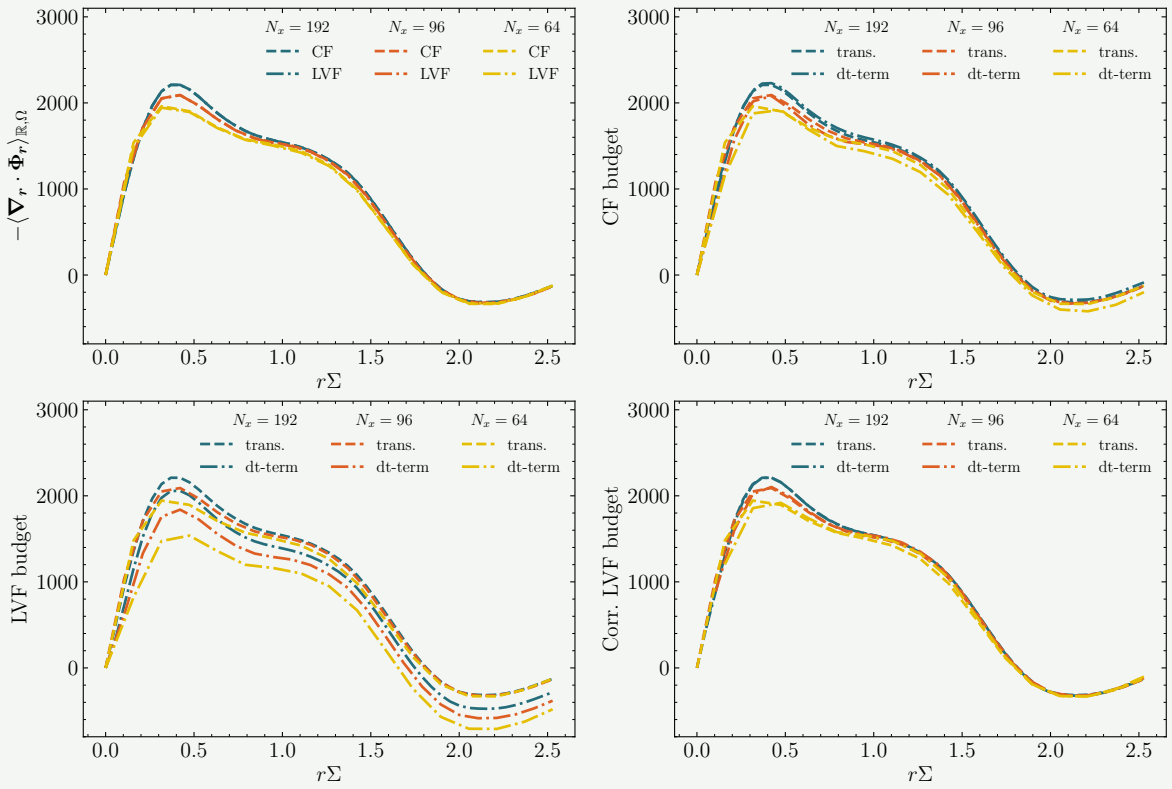

Figure 4. Effect of the mesh size on the transfer term of either CF or LVF fields (top left), the budget of the CF field (top right), the budget of LVF field (bottom left) and corrected LVF field (bottom right). Results are for $t = 30$ μs

We start by analysing the difference between the transfer term $-\langle \boldsymbol{\nabla_r} \cdot \boldsymbol{\Phi_r} \rangle_{\mathbb{R},\Omega}$ of the LVF and CF fields for different spatial resolutions. Results are presented in Fig. 4 for three spatial resolutions $64^3, 96^3, 192^3$. It appears that the transfer term of the CF and LVF fields are undistinguishable. However there remains an influence of the spatial resolution which is mostly perceptible around the peak of energy transfer $r\Sigma \approx 0.5$. Similar trends were observed at different time during the simulation. Therefore, although there was a clear difference between second-order structure functions of the LVF and CF fields for finite resolutions, the r-transfer term does not reveal such discrepancies. Further investigations are required for confirming this as regards to the X-transfer and production terms.

Let us now turn our attention to the budget Eq. (1.2) pertaining to either the LVF or CF field. The corrected LVF field which is obtained by Eq. (4.5) is also considered. Results at a time $t = 30$ μs are presented in Fig. 4 for three spatial resolutions $64^3, 96^3, 192^3$.

The budget for the CF field is not satisfactorily closed for a 64^3 resolution. However, we observe that the error between the r-transfer term and the time-derivative term diminishes when spatial resolution is increased. At the highest resolution the unsteady term compensates almost perfectly the r-transfer term and the budget is nicely closed.

As far as the LVF field is concerned, the discrepancy between the two terms in Eq. (1.2) are more pronounced. At the highest resolution 192^3, there remain substantial errors in the closure of the two-point equation. Our previous findings on the effect of spatial resolution on $\langle (\delta\phi)^2 \rangle_{\mathbb{R},\Omega}$ allows us elaborating a bit more on the origin of this discrepancy. Indeed, it was shown that second- and third-order statistics are not equally affected by finite resolution effects. More precisely, although we observed that

$$\langle \boldsymbol{\nabla_r} \cdot \boldsymbol{\Phi_r} \rangle_{\mathbb{R},\Omega}\big|_{\mathrm{LVF}} = \langle \boldsymbol{\nabla_r} \cdot \boldsymbol{\Phi_r} \rangle_{\mathbb{R},\Omega}\big|_{\mathrm{CF}}\,, \tag{4.7}$$

deriving Eq. (4.6) w.r.t time leads to

$$\partial_t \langle (\delta\phi)^2 \rangle_{\mathbb{R},\Omega}\Big|_{\mathrm{Corr.}} = \frac{\partial_t \left\langle (\delta\phi)^2 \right\rangle_{\mathbb{R},\Omega}\Big|_{\mathrm{LVF}}}{1 + (a\Sigma)^2} + \frac{F(r)\left[1 - (a\Sigma)^2\right] - 2a\Sigma \left\langle (\delta\phi)^2 \right\rangle_{\mathbb{R},\Omega}\Big|_{\mathrm{LVF}}}{\left[1 + (a\Sigma)^2\right]^2}\, \partial_t a\Sigma \tag{4.8}$$

Therefore, the errors on the budget closure due to finite resolutions are made at the level of the time-derivative term. These errors increases with both $a\Sigma$ and $\partial_t a\Sigma$. In Ref. [1], the errors on the budget were not perceptible because $a\Sigma$ was small enough and further because $\partial_t a\Sigma$ was negligible compared to $\partial_t \langle (\delta\phi)^2 \rangle_{\mathbb{R},\Omega}$.

The budget of the corrected LVF field with the time derivative term computed from Eq. (4.8) is shown in Fig. 4. The latter appears to be very nicely closed, almost irrespectively of the numerical resolution. Interestingly, Eq. (1.2) is even better satisfied than it was for the CF field.

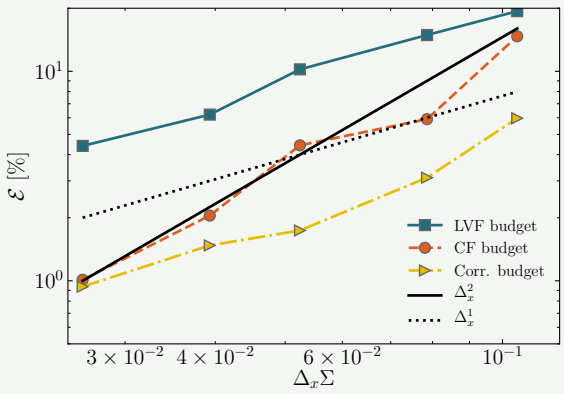

Figure 5. Convergence of Eq. (1.2) w.r.t the mesh size for both the CF, LVF and corrected LVF fields. The black dotted and full lines correspond to a first- and second-order convergence, respectively

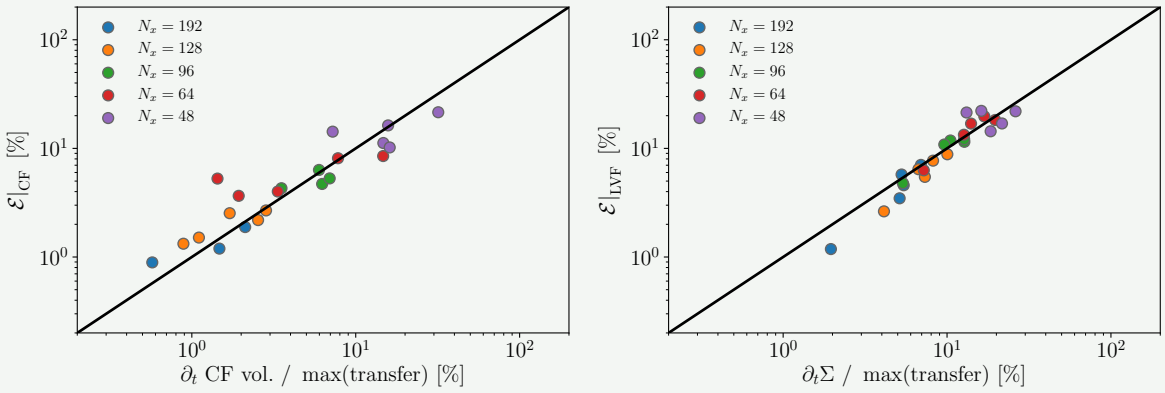

Figure 6. Scatter plot of the error on the budget of the CF field (left) as a function of the CF volume loss. On the right, the error for the LVF field is plotted as a function of the time-derivative of $a\Sigma$. Each quantity is divided by the maximum of the r-transfer term.

5. Order of convergence and sources of error

The error on Eq. (1.2) can be quantified by the L_2-norm of the difference between the time-derivative and the r-transfer terms, viz.

$$\mathcal{E}(r,t)\ [\%] = 100 \times \frac{\left|\partial_t \langle (\delta\phi)^2 \rangle_{\mathbb{R},\Omega} + \langle \boldsymbol{\nabla_r} \cdot \boldsymbol{\Phi_r} \rangle_{\mathbb{R},\Omega}\right|}{\max(-\langle \boldsymbol{\nabla_r} \cdot \boldsymbol{\Phi_r} \rangle_{\mathbb{R},\Omega})} \tag{5.1}$$

The evolution of this quantity, after being averaged over all separations r and over several time-steps t, is plotted as a function of the grid spacing in Fig. 5. A log-log plot representation is chosen so that to appraise the order of convergence of Eq. (1.2). It is readily seen that the error that is made on the budget of the LVF field is significantly larger than the one pertaining to the CF field. Further, while the budget of the LVF field appears to converge with an approximate first-order scaling, that of the CF field has a convergence close to second-order. The corrected LVF budget using Eq. (4.8) reveals a substantially smaller error especially for coarse meshes. In addition, the convergence of the corrected LVF field remains close to first order. For a resolution of 192^3 the error for the CF and corrected LVF fields is within 1% while for the LVF field, the error attains 5% of the maximum of the r transfer term.

It is worth questioning the source of error in the budget for both the CF and LVF fields. As far as the CF field is concerned the origin of the discrepancy between the unsteady and transfer term are most likely due to the binarization used to define the CF field. Indeed, the volume of the CF field defined as the volume of the excursion set level-set$(\boldsymbol{x}) > 0$ may not be constant in time. Given that Eq. (1.2) is the equation for a conserved non diffusive scalar, the non constancy of the CF mass might cause significant errors. To confirm this, we have plotted in Fig. 6 the error on Eq. (1.2) as a function of the time-derivative of the CF field volume (divided by the maximum of the r-transfer term). Results align quite well along a line which indicate a correlation between the closure of the CF field and the conservation of the CF volume.

On the other hand, the LVF field does not suffer from this drawback since conservation of the LVF mass is much more accurately satisfied thanks to the Rudman type solver. However, as discussed before, the budget of LVF field is significantly altered by the value of $a\Sigma$ and its variation in time. A scatter plot of the error on the budget of the LVF

field as a function of $\partial_t a\Sigma$ (divided by the maximum of the r-transfer term) is presented in Fig. 6(right). Here again, results align quite well along a line which confirm our previous statements.

6. Conclusion

This paper provides a description of the `pyarcher.increments` library. It further aims at documenting some validation test cases allowing us to appraise *(i)* the accuracy of the numerical procedures for computing angular and spherical average, *(ii)* the effect of finite spatial resolution on the two-point statistics of the LVF and CF fields and *(iii)* the closure of Eq. (1.2) with respect to the grid spacing and the sources of errors. Different outcomes emerge from the present analysis:

SUMMARY

(1) The new procedure for computing spherical and angular averages on the basis of the `Rbf + tplquad` or `dblquad` works much better than the one initially used by Ref. [1] where use was made of the `RegularGridInterpolator` and the `trapz` functions. The difference between the two methods is mostly perceptible at small scales and hence, to limit computational expenses, a blended version is proposed that uses the `RBF` method at small scales and the `RGI` method at larger scales. Further, because the angular average requires only double integration over θ and φ, it should be preferred over the spherical average when large datasets are concerned.

(2) We emphasize that the second-order structure function of the CF field does not depend on the spatial resolution. On the contrary, the LVF field is strongly affected by the mesh size. We proposed a scale-dependent correction for the LVF field second-order structure function which explicitly account for the finite value of $\Sigma\Delta x$. On the other hand, we notice that there was no difference of the r-transfer term between the CF and LVF fields. This term was further shown to depend only weakly on the mesh size.

(3) The closure of Eq. (1.2) for the LVF and CF fields is appraised for different grid spacing. It is shown that the budget of the CF field converge more quickly than that of the LVF field. Furthermore, the error was substantially smaller for the CF field than the LVF field. The plausible sources of discrepancy for the budgets are discussed. In particular, it appears that the main source of error for the CF budget is the non-constancy of the CF volume while for the LVF field, the error arises from the influence of both $\Sigma\Delta x$ and $\partial_t\Sigma\Delta x$. In Ref. [1], the budget was well closed because the grid spacing was small enough for these effects to be negligible. Finally a correction for the LVF budget is proposed which has the double benefit of correcting the LVF statistics from the influence of $\Sigma\Delta x$ and $\partial_t\Sigma\Delta x$ while taking advantage the mass conservation of the LVF field. This corrected budget was shown to yield the smallest errors and is especially recommended when relatively coarse meshes are used.

References

1. Thiesset F, Duret B, Ménard T, Dumouchel C, Reveillon J, Demoulin F. 2020 Liquid transport in scale space. *J. Fluid Mech.* **886**, A4.
2. Thiesset F, Dumouchel C, Ménard T. 2019 A New Theoretical Framework for Characterizing the Transport of Liquid in Turbulent Two-Phase Flows. In *ILASS-Europe, Paris*.
3. Thiesset F, Ménard T, Dumouchel C. 2020 Space-scale-time dynamics of liquid/gas shear flow. *submitted*.
4. Kirste R, Porod G. 1962 Röntgenkleinwinkelstreuung an kolloiden Systemen Asymptotisches Verhalten der Streukurven. *Kolloid-Zeitschrift und Zeitschrift für Polymere* **184**, 1–7.
5. Frisch H, Stillinger F. 1963 Contribution to the statistical geometric basis of radiation scattering. *J. Chem. Phys.* **38**, 2200–2207.
6. Hill R. 2002 Exact second-order structure-function relationships. *J. Fluid Mech.* **468**, 317–326.
7. Berryman JG. 1987 Relationship between specific surface area and spatial correlation functions for anisotropic porous media. *J Math Phys* **28**, 244–245.
8. Ménard T, Tanguy S, Berlemont A. 2007 Coupling level set/VOF/ghost fluid methods: Validation and application to 3D simulation of the primary break-up of a liquid jet. *Int. J. Multiphase Flow* **33**, 510–524.
9. Rudman M. 1998 A volume-tracking method for incompressible multifluid flows with large density variations. *Int J Numer Methods Fluids* **28**, 357–378.
10. Vaudor G, Ménard T, Aniszewski W, Doring M, Berlemont A. 2017 A consistent mass and momentum flux computation method for two phase flows. Application to atomization process. *Computers Fluids* **152**, 204–216.
11. Sussman M, Smith K, Hussaini M, Ohta M, Zhi-Wei R. 2007 A sharp interface method for incompressible two-phase flows. *J Comput Phys* **221**, 469–505.
12. Zhang J. 1996 Acceleration of five-point red-black Gauss-Seidel in multigrid for Poisson equation. *Appl. Math. Comput.* **80**, 73–93.

13. Fedkiw RP, Aslam T, Merriman B, Osher S. 1999 A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J Comput Phys* **152**, 457–492.
14. Duret B, Luret G, Reveillon J, Ménard T, Berlemont A, Demoulin F. 2012 DNS analysis of turbulent mixing in two-phase flows. *Int. J. Multiphase Flow* **40**, 93–105.
15. Di Battista R, Bermejo-Moreno I, Ménard T, de Chaisemartin S, Massot M. 2019 Post-processing of two-phase DNS simulations exploiting geometrical features and topological invariants to extract flow statistics: application to canonical objects and the collision of two droplets. In *10^{th} International Conference on Multiphase Flow, Rio de Janeiro, Brazil*.
16. Essadki M, Drui F, Larat A, Ménard T, Massot M. 2019 Statistical modeling of the gas–liquid interface using geometrical variables: Toward a unified description of the disperse and separated phase flows. *Int. J. Multiphase Flow* **120**, 103084.
17. Fitzhugh R. 1983 Statistical properties of the asymmetric random telegraph signal, with applications to single-channel analysis. *Math. Biosci.* **64**, 75–89.