SynInflow:
a synthetic inflow boundary condition for OpenFOAM

Version 1.0 developed for OF 21.12

September 1, 2022

Contents

1 About SynInflow

Thank you for your interest in SynInflow. SynInflow provides an easy to use Boundary Condition, BC, able to impose synthetic turbulent velocity fields at the inlet of your CFD simulation. It is developed at the Laboratory of Computational Mechanics of University of Bologna, primarily targeting Computational Wind Engineering applications. For further info visit the website `https://site.unibo.it/cwe-lamc/en` and do not hesitate to write us in case you have any doubt.

1.1 Main features

SynInflow allows to generate synthetic turbulent fields, from homogeneous turbulence to atmospheric boundary layer flows, which can be used as inflow BC for scale resolving simulations. The adopted inflow generation method provides control over the turbulence intensity of the three velocity components and allows a good control over the nine integral length scales. The generated inflows can be easily modified by the user without editing the source code. SynInflow also incorporates velocity corrections able to strongly alleviate the insurgence of spurious pressure fluctuations, often encountered when using synthetic inflows. SynInflow also provides basic support to allows for time-variability of the applied velocity field.

1.2 Licensing and compatibility

SynInflow is released under the GNU GENERAL PUBLIC LICENSE Version 3. The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

The present implementation has been developed within OpenFOAM 21.12 and it is expected to work also for subsequent ESI-OpenCFD releases, while adaptation to the OpenFOAM Foundation version requires minor modifications of the source code.

2 Getting started

2.1 Set up your case

Setting up your synInflow BC usually requires four simple steps, reported in Fig. 1.

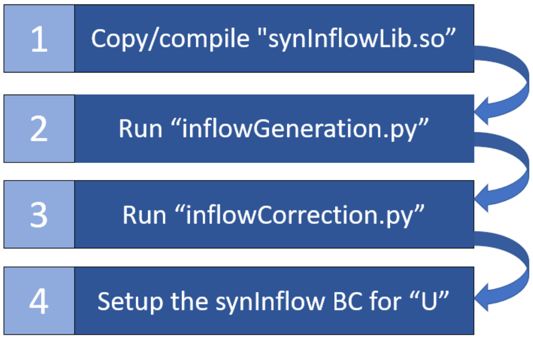
1
Copy/compile "synInflowLib.so”
2
Run “inflowGeneration.py”
3
Run “inflowCorrection.py”
4
Setup the synInflow BC for “U”

Figure 1: Steps needed to use the synInflow BC.

Firstly, copy the *synInflowLib.so* into your case main folder. If you are using it for the first time, you find it in the provided examples or in the provided *CodeOF* folder. It is anyway advisable to recompile the code for your system following the instructions provided in Section 2.2, it takes just a few seconds. Then, add the following line at the beginning of your *controlDict*:

```
libs("./synInflowLib.so").
```

Once *synInflowLib.so* is in your case main folder and you added it in the *controlDict*, run *inflowGeneration.py* producing the *inflowGenerationDict* as detailed in Section 4.1. Then, run *inflowCorrection.py* producing the *inflowCorrectionDict* as detailed in Section 4.2. Finally, set the BC for the velocity field, *U*, in the *”0”* folder as follows (where the name of the inflow patch has been here chosen to be *inlet*):

```
inlet
{
        type synInflow;
        origin (0 0 0);
        zDir (0 0 1);
        sf_dictName inflowGenerationDict;
        vc_dictName inflowCorrectionDict;
        value uniform (0 0 0);

}
```

The meaning of the keywords is detailed below:

type: [*required*] selects the synInflow BC. Do not change;
origin: [*required*] origin of the local reference system used to evaluate the inflow velocity field (it

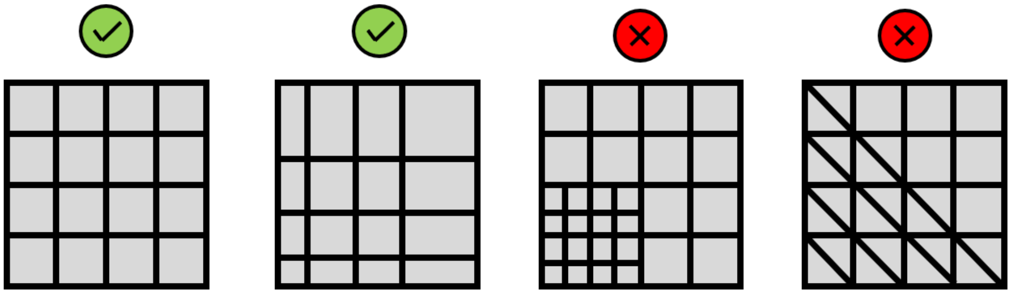

Figure 2: Examples of valid and invalid meshes at the inflow patch.

can be used, for instance, if the ground plane does not pass through the origin);
`zDir`: [*required*] unit vector indicating the local reference system $z-$direction (in which the time-averaged velocity and turbulence characteristics can vary according to the *profiles* described in Section 4.1);
`sf_dictName`: [*required*] name of the dictionary containing data needed to build the synthetic flow. The dictionary must be put in the *system* folder and it is usually generated using *inflowGeneration.py*, as described in Section 4.1;
`vc_dictName`: [*optional*] dictionary containing data needed to calculate the velocity corrections. If not provided corrections are not calculated and the uncorrected synthetic field is applied. The dictionary must be put in the *system* folder and it is usually generated using *inflowCorrection.py*, as described in Section 4.2;
`value`: [*required*] initial value not actually used.

NB 1 : The inlet patch is can be arbitrarily oriented, but it is required to be planar.

NB 2 : The current implementation of synInflow requires the inlet patch to be meshed conformally exclusively with quadrilateral faces in order to calculate inflow corrections correctly. Examples of valid and invalid meshes at the inflow patch are provided in Fig. 2. It is always possible to use the BC without calculating corrections (and so expecting higher spurious pressure fluctuations) eliminating the optional keyword *vc_dictName*. In this way corrections are not calculated and no restriction on the adopted mesh is present.

NB 3 : The current implementation of synInflow does not allow to reconstruct a decomposed case.

NB 4 : The current implementation of synInflow does allow for restarts, but minor differences in the obtained results might appear in proximity of the restart time.

NB 5 : In the present version, applying corrections generally leads to a slight increase of turbulence intensity for velocity components normal to the time-averaged flow (see Section 5.2). If this is a problem, target a lower turbulence intensity from the beginning.

2.2 Compiling *synInflowLib.so*

In order to compile *synInflowLib.so* for your system, open a terminal and go to the provided *CodeOF* folder. Then execute the command:

```
wmake libso
```

Then copy the newly created *synInflowLib.so* into your case main folder.

3 Theoretical background

The present BC can be seen as the union of two components: an inflow generation method and an inflow correction procedure. The two components are briefly described below.

3.1 Inflow generation

SynInflow uses a spectral approach to generate random fluctuations. In the homogeneous case, the velocity field is calculated as:

$$\mathbf{u_s}(\mathbf{x},t) = \bar{\mathbf{u}} + \sum_i \left[\mathbf{p_i} cos(2\pi\, \mathbf{k_i}^T \mathbf{x_i}^* + \phi_i) + \mathbf{q_i} sin(2\pi\, \mathbf{k_i}^T \mathbf{x_i}^* + \phi_i)\right], \tag{1}$$

where $\mathbf{u_s} = [u_s, v_s, w_s]^T$ is the synthetic velocity field, $\bar{\mathbf{u}} = [\bar{u}, 0, 0]^T$ is the time-averaged velocity field, $\mathbf{x} = [x, y, z]^T$ is the position vector, $\mathbf{x}^* = [x - \bar{u}t, y, z]^{*T}$ is a modified position vector, in which the convection operated by the time-averaged velocity field, $\bar{u}$, is taken into account. Vectors $\mathbf{p_i}$ and $\mathbf{q_i}$ are amplitude vectors, $\mathbf{k_i}$ is a wavevector and ϕ_i is a random phase (actually redundant in this form).

The synthetic inflow generator here adopted (i.e. the technique used to calculate $\mathbf{k_i}$, $\mathbf{p_i}$, $\mathbf{q_i}$) is called Prescribed-wavelength Random Flow Generator^3, PRFG^3, and it is described in [1] for the homogeneous case and in [2] for the inhomogeneous case, targeting Atmospheric Boundary Layer, ABL, flows. PRFG^3 allows to impose the turbulence intensity for all three velocity components and provides good control over all nine integral length scales. Turbulence is currently generated in agreement with the well-known von Kármán spectrum.

In homogeneous conditions, turbulence is built in order to be divergence-free and to approximate momentum balance using linear convection based on the time-averaged velocity. This ensures that the resulting syntectic turbulence field complies with the linearization of Navier-Stokes equations in the proximity of the time-averaged conditions, in order to correctly transmit the synthetic field within the CFD domain. For inhomogeneous cases, the synthetic turbulent field generated using PRFG^3 can only approximate the aforementioned conditions in the case of slowly varying time-averaged velocity and turbulence properties [2].

3.2 Inflow correction

Once the synthetic turbulence field is generated, it can be applied at the inflow patch as a Dirichlet type BC. Unfortunately, this almost inevitably leads to spurious pressure fluctuations. The origin of such pressure fluctuations shall be attributed to mismatches between the synthetic turbulence field and

1. the Navier-Stokes equations;
2. the BCs confining with the inflow patch.

As regards (1), if the synthetic field is homogeneous, PRFG^3 ensures that the Navier-Stokes equations linearized in correspondence of the time-averaged velocity field are fulfilled (also assuming null-normal pressure gradient as usually specified at the inflow and disregarding viscous terms). In other cases, the generated turbulent field will only approximately satisfy such conditions, so that some spurious pressure fluctuations shall be generally expected. As regard (2), synthetic inflow conditions usually do not take into account BCs confining with the inflow patch.

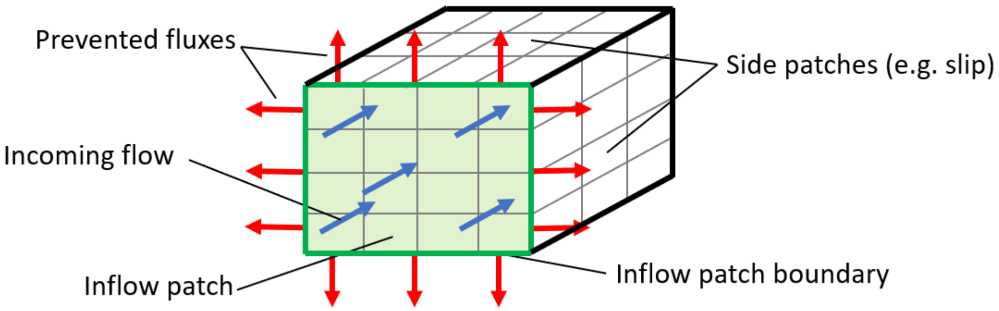

Figure 3: Sketch of an inflow patch confining with side patches which prevent fluxes.

As an example, consider a channel flow for which the channel walls prevent mass fluxes along their normal direction (see Fig. 3). If the synthetic flow applied at the inflow does not take such aspect into account, spurious pressure fluctuations arise at the edge between the inflow patch and the side walls to compensate for such missing mass fluxes.

The procedure here adopted to moderate spurious pressure fluctuations is named Variationally Based Inflow Correction, VBIC [3]. The VBIC procedure has been designed aiming at moderating spurious pressure fluctuations due to both (1) and (2) applying corrections to the synthetic velocity field, so that

$$\mathbf{u} = \mathbf{u_s} + \mathbf{u_c}, \tag{2}$$

where $\mathbf{u}$ is the corrected velocity field actually applied at the inflow patch and $\mathbf{u_c} = [0, v_c, w_c]^T$ are velocity corrections calculated in order to be of minimal norm. Notice that is it sufficient to apply corrections only to the velocity components laying on the plane of the inlet patch. Interested readers can refer to the original paper for details [3].

4 Usage

As already stated, using the synInflow BC requires to provide two dictionaries which contain all the data needed to generate the synthetic flow and calculate corrections, named *inflowGenerationDict* and *inflowCorrectionDict*, respectively. Such dictionaries are usually automatically generated relying on *inflowGeneration.py* and *inflowCorrection.py*. Both routines have been developed using Pythn 3.8. Once generated, such dictionaries must be copied in your case *system* folder.

As a general remark, the synthetic inflow velocity field is initially generated in such a way that the $x-$ direction corresponds to the time-averaged flow direction and the $z-$ direction corresponds to the one in which both the time-averaged velocity and the turbulence properties can vary. Then, using the BC in OpenFOAM, the $x-$ direction is identified as the inlet patch inward normal, the $z-$ direction is specified by the user (as reported in Section 2.1, using the **`zDir`** parameter) and the $y-$ direction is obtained by cross product. It follows that in the current implantation the time-averaged velocity is always normal to the inflow patch.

4.1 Inflow generation

SynInflow uses *profiles* in order to specify variations of the time-averaged velocity and turbulence properties along the $z-$ direction. *Profiles* are specified using $n \times 3$ matrices in which: column 1 is a coordinate (i.e. z), column 2 is a value, column 3 is a multiplier. The value actually adopted during calculations is the product of the value (column 2) and the multiplier (column 3). This allows to apply corrections to the *profiles* keeping trace of the original value. When needed, intermediate values are linearly interpolated while extrapolation is performed taking the first (or last) provided value. By extension, the same format is also adopted to describe time-variations.

4.1.1 *inflowGenerationDict*

The keywords appearing in *inflowGenerationDict* are discussed below.

`cutOff:` [*required*] during spectral synthesis only waves such that $|k| \; \Delta x < cutOff$ are considered, where Δx is a typical mesh edge size calculated as the square root of the inlet patch area divided by the total number of faces on the same patch. This eliminates from Eq. (1) high frequency contributions, which would be anyway quickly dissipated. A value comprised between 0.5 and 1.0 is usually appropriate.
`seeds:` [*required*] $z-$coordinates of the seeds used to generate turbulence in the inhomogeneous case [2]. Substantially, turbulent velocity fields are generated based on the properties evaluated at seeds. Then, such turbulent fields are blended together using weighting functions. The spacing between seeds shall be the highest possible which allows to reproduce the variations of the target time-averaged speed and turbulence properties with sufficient accuracy. See Section 4.3 for further details.
`tProfile:` [*required*] a time-varying multiplier used to modulate the generated field in time (both time-averaged and fluctuating parts). It is provided using the profile format previously described.
`uProfile:` [*required*] a profile describing the time-averaged velocity, $\bar{u}$, distribution along the local $z-$ direction.
`sProfile:` [*required*] a profile describing the standard deviation of the along-wind velocity component, σ_u, along the local $z-$ direction.
`lProfile:` [*required*] a profile describing the along-wind integral length scale, L_{ux}, distribution along the local $z-$ direction.
`kMainSeed:` [*required*] a list of wavevectors used to generate the random fluctuations. Such values shall be normalized in order to yield a unit along-wind integral length scale, $L_{ux} = 1.0$.
`pMainSeed:` [*required*] a list of amplitude vectors used to generate the turbulent fluctuations. Such values, together with *qMainSeed*, shall be normalized in order to yield a unit standard deviation of the along-wind turbulent velocity fluctuations, $\sigma_u = 1.0$.

`qMainSeed`: [*required*] a list of amplitude vectors used to generate the turbulent fluctuations. Such values, together with *pMainSeed*, shall be normalized in order to yield a unit standard deviation of the along-wind turbulent velocity fluctuations, $\sigma_u = 1.0$.

4.1.2 *inflowGeneration.py*

The routine allows to generate the *inflowGenerationDict*. It relies on two libraries *profilesLib* and *PRFG3Lib*. In order to generate the *inflowGenerationDict* proceed as follows:

1. **`seeds`** must be manually defined as an array of $z-$ coordinates;

2. **`uProfile`** is defined using, for example, the command

   ```
   uProfile = profilesLib.UProfile(zBottom,zTop,nZ).uniformProfile(1.0)
   ```

   where *zBottom,zTop* are the lower and higher z coordinate values, *nZ* is the number of subdivision of the z interval and *uniformProfile* indicates that a uniform-type profile of value 1.0 is targeted. In order to have a complete list of the available profiles open *profilesLib.py* in correspondence of the *UProfile* object definition. Profiles can be easily directly edited by the user.

3. **`sProfile`** is defined using, for example, the command

   ```
   sProfile = profilesLib.SProfile(zBottom,zTop,nZ).uniformProfile(1.0)
   ```

   with the same meaning of the symbols previously presented. In order to have a complete list of the available profiles open *profilesLib.py* in correspondence of the *SProfile* object definition. Profiles can be easily directly edited by the user.

4. **`lProfile`** is defined using, for example, the command

   ```
   lProfile = profilesLib.LProfile(zBottom,zTop,nZ).uniformProfile(1.0)
   ```

   In order to have a complete list of the available profiles open *profilesLib.py* in correspondence of the *LProfile* object definition. Profiles can be easily directly edited by the user.

5. **`tProfile`** is defined using, for example, the command

   ```
   tProfile = profilesLib.TProfile(zBottom,zTop,nZ).constantProfile()
   ```

In order to have a complete list of the available profiles open *profilesLib.py* in correspondence of the *TProfile* object definition. Profiles can be easily directly edited by the user.

6. In the previous steps, all the data needed to characterize the time-average and turbulence characteristics distributions have been provided. It is now necessary to extract the vectors $\mathbf{k_i}$, $\mathbf{p_i}$ and $\mathbf{q_i}$ appearing in Eq. (1). For each *seed*, a set of $\mathbf{k_i}$, $\mathbf{p_i}$ and $\mathbf{q_i}$ shall be defined, according to the local target properties.

   In the present implementation, for the sake of simplicity, only one set of $\mathbf{k_i}$, $\mathbf{p_i}$ and $\mathbf{q_i}$ is defined, indicated as the *mainSeed.* The *mainSeed* is expected to be characterized by $\sigma_u = 1$ and $L_{ux} = 1$. All other seeds are obtained from the *mainSeed* by rescalings. In order to generate the $\mathbf{k_i}$, $\mathbf{p_i}$ and $\mathbf{q_i}$ the *PRFG3Lib.PRFG3Turb(turbParameters)* object is used, in which *turbParameters* is a dictionary with the following entries:

   `L:` a matrix of integral scales organized as $[[L_{ux}, L_{uy}, L_{uz}], [L_{vx}, L_{vy}, L_{vz}], [L_{wx}, L_{wy}, L_{wz}]]/L_{ux}$;
   `Covariance:` a diagonal matrix $diag([\sigma_u^2, \sigma_v^2, \sigma_w^2]/\sigma_u^2)$;
   `dEd:` an integer controlling how many velocity waves shall be used to discretized the marginal spectra. Usually already 5 leads to acceptable results. The total number of waves appearing in Eq. (1) will increase sharply when increasing *dEd*;
   `finalE:` the velocity marginal spectra are sampled up to reaching approximately $\mathit{finalE}^{1/3}$ of the total energy: the contribution of higher frequencies is disregarded. Usually choose 0.95.

NB 6 : In the first lines of the *inflowGenerationDict* a full description of the parameters and functions used within *inflowGenerationDict.py* is reported within the comments.

NB 7 : Notice again that quantities appearing in *L* shall be normalized so that $L_{ux} = 1$ and that quantities appearing in *Covariance* shall be normalized so that $\sigma_u^2 = 1$.

NB 8 : By using only one *mainSeed* to obtain all others by rescalings, the turbulence anisotropy is fixed. This means that the ratios specified in *L* and *Covariance* are the same for all seeds.

NB 9 : In the current implementation the Reynolds stress tensor is assumed to be diagonal.

NB 10 : The user can easily add additional profiles or modify existing ones. Other spectral turbulence generation methods might be used to calculate $\mathbf{k_i}$, $\mathbf{p_i}$ and $\mathbf{q_i}$.

4.2 Inflow correction

As already specified, the synInflow BC implements velocity corrections following the VBIC method in order to moderate pressure fluctuations [3]. In this implementation, at the inflow patch boundary, prevented mass fluxes are avoided setting the appropriate velocity component to zero. Such choice is consistent with symmetry/slip/wall BC confining with the inlet patch (see Fig. 3). Such choice, although not optimal, provides good results also when periodic BCs confine with the inlet patch (i.e. the velocity component normal to the periodic patch is made periodic with a null value).

4.2.1 *inflowCorrectionDict*

The *inflowCorrectionDict* contains all the data needed to build the velocity corrections. Such data must be generated using *inflowCorrection.py*. The only parameter which can be edited by the user is:

`corr`, which is a multiplier applied to the velocity corrections (i.e. setting it to 0.0 yields an uncorrected velocity field applied at the inflow patch, while a value 1.0 leads to a fully corrected field).

4.2.2 *inflowCorrection.py*

In order to use the routine copy the folder *polyMesh* of your case into the folder containing *inflowCorrection.py*. Change *patchName* to your inlet patch name and the *dictName* to the desired name for the dictionary to be given in output. Run the code, copy the resulting dictionary into your case *system* folder and proceed as previously indicated in Section 2.1.

NB 11 : The *polyMesh* must be in ASCII format, the binary format is not supported.

NB 12 : Velocity corrections are calculated in a local reference system, which might be different from the one used to generate the synthetic flow. Such local reference system is provided in output when running the routine, but it is chosen in a fully automatic way and the user neither has nor needs control over it.

4.3 Tips and tricks

Here are some useful tips and tricks:

1. it is always a good idea to recompile *synInflowLib.so* for your system and OpenFOAM version, even if the provided compiled library does not rise errors;

2. *seeds* shall be as few as possible keeping the distance between them as large as possible, in order to have slowly varying weighting functions. This, on the other hand, decreases the accuracy of the obtained profiles. It is necessary to compromise between such opposing necessities. The distance between *seeds* is an upper limit to the obtainable vertical integral length scales for all velocity components. The time-averaged field is interpolated directly from the profiles not using seeds, so that its accuracy does not depend on the number of *seeds* or location;

3. just before the first iteration some checks are performed comparing the data in the *inflowCorrectionDict* and the current mesh. Check if such controls are passed. If not, maybe your *inflowCorrectionDict* has not been generated for the present mesh;

4. at each iteration the value of the time multiplier is reported. Such value is the one obtained from the *tProfiles*;

5. *tProfiles* are very useful for initializing your simulation, gradually increasing velocity, so avoiding the need to select very small time steps during initialization;

6. not specifying the *inflowCorrectionDict* in the BC setup for the *U* field in the *"0"* folder or setting **`corr`** equal to 0 in the *inflowCorrectionDict* might have similar results, but is it substantially different (due to the different implementations). If your *inflowCorrectionDict* is not generated for your current mesh, suppress velocity corrections exclusively eliminating *inflowCorrectionDict* form the *U* field boundary values specifications in your case *"0"* folder.
7. if the purpose of your analysis is not to measure pressures, it might be preferable to set **`corr`** to a value of about 0.5 to obtain the best match between the targeterd and the profiles actually found in the simulation (see Section 5.2). Pressure fluctuations usually decay considerably at a distance approximately equal to two integral length scales downstream the inflow patch;
8. when calculating velocity corrections, it is fundamental that no flow reversal happens at the inflow boundary. This is obtained automatically setting minimal values to the along wind velocity component;
9. the inflow patch shall be large enough with respect to the relevant integral length scales;
10. profiles (time-averaged velocity and turbulence properties) might evolve downstream the inflow patch, which is a typical problem in ABL flows. This is only partially related to the inflow generation procedure and selected target values. Indeed, if the correct roughness is not applied at the ground patch, the same behaviour is observed in nature and in wind tunnel tests. Also the dissipation operated by the grid might play an important role;
11. the names of *inflowGenerationDict* and *inflowCorrectionDict* can be changed arbitrarily, which is useful in case multiple inlets with different characteristics are defined. Just point to the correct ones when setting the BC for the velocity field.

5 Examples

To run the examples, first run *blockMesh* and than the solver, e.g. *pisoFoam*. If you make any change to the mesh or for some reasons preliminary checks are not passed update *inflowCorrectionDict* following the instructions above.

5.1 *SimpleInflow*

The first example provided is called *SimpleInflow*, derived from the examples shown in [3]. Assuming time-average velocity equal to 1.0, we reduce the summation appearing in Eq. (1) to a single term such that

$$\mathbf{k} = [1, 1, 1]^T; \qquad \mathbf{p} = [0.1, 0.05, 0.05]^T; \qquad \mathbf{q} = [0, 0, 0]^T. \tag{3}$$

Such inflow, denoted as *Original* in the following, is not divergence-free as $\mathbf{k}$ and $\mathbf{p}$ are not orthogonal, so that spurious pressure fluctuations and a corresponding modification of the velocity field are to be expected if no corrections are applied. Then, we modify the inflow so that

$$\mathbf{k} = [1, -1, -1]^T; \qquad \mathbf{p} = [0.1, 0.05, 0.05]^T; \qquad \mathbf{q} = [0, 0, 0]^T. \tag{4}$$

Such inflow, denoted as *Div-free* in the following, is divergence-free (and fulfills linear convection due to the presence of $\mathbf{x}^*$ in Eq. 1). As a result, it is expected to lead to major spurious pressure fluctuations only due to BCs mismatches. For both inflows, pressure fluctuations can be mitigated relying on VBIC.

Figure 4 reports the pressure coefficient $C_p = p/q$ with q kinetic pressure based on the time-averaged velocity. Pressure fluctuations measured for the corrected case are one order of magnitude smaller than the uncorrected cases, as shown also by the minimum and maximum values reported on top of each case.

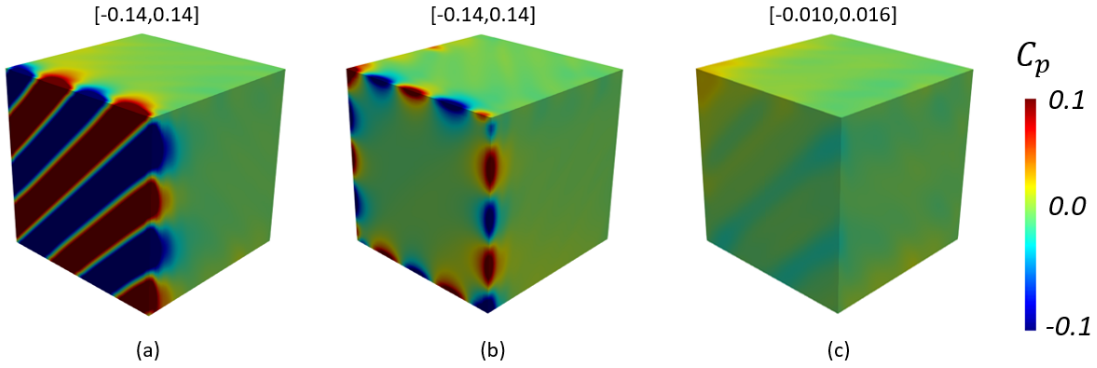

Figure 4: The C_p distribution at $t = 3$ for *SimpleInflow*: (a) *Original*, (b) *Div-free*, (c) *Original* corrected. Minimum and maximum values reported on top of each case.

5.2 *ABLInflow*

This example provides on overview of how it is possible to generate an Atmospheric Boundary Layer, ABL, inflow condition with SynInflow. Firstly, we generate the *inflowGenerationDict* using *inflowGeneration.py*, targeting an Eurocode Category III profile in full-scale. We select a base velocity equal to 20 m/s. All the parameters used in the inflow generation procedure are reported at the beginning of the *inflowGenerationDict*. In order to analyse the obtained results we run the case for 1000 s and disregard the first 100 s to avoid initialization. In analysing the results, it shall be considered that the example is illustrative and the adopted mesh is relatively coarse, so that variations of the results might occur when adopting finer meshes and longer simulation times.

We start by analyzing the results obtained at *Profile 0*, corresponding to a central position just downstream the inflow patch. Remember that PRFG^3 can only be approximately divergence-free in the inhomogeneous case for cases characterized by slowly varying properties.

We firstly consider the case in which VBIC is disabled, i.e. `corr = 0` in the *inflowCorrectionDict*. An overview of the obtained results is reported in Fig. 5, showing a good agreement between the targeted and the obtained values. Notice that the shape of the I_w profile is modified close to the ground due to its impermeability, which damps out vertical velocity components: such aspect is not taken into account by the targeted profiles.

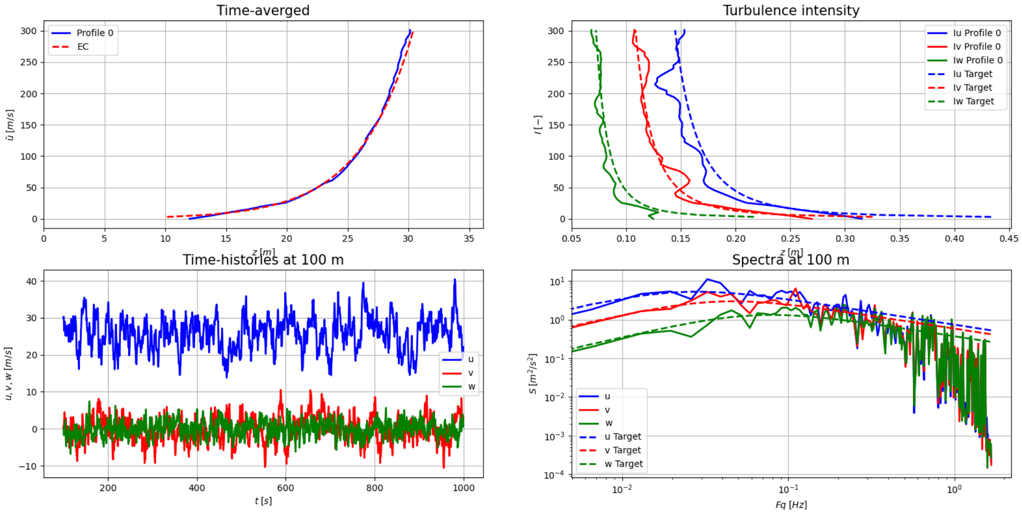

Figure 5: Overview of the results obtained for the *ABLInflow* in terms of velocity field without VBIC.

Then, we consider the same case enabling VBIC i.e. `corr = 1` in the *inflowCorrectionDict*. We see that, overall, good results are still achieved, although an increase of turbulence intensity for the v velocity component is observed, so that the I_v turbulence intensity is now similar to the I_u one. Notice that the I_u profile is actually identical to the uncorrected case, as the curract implementation of VBIC applies corrections only to transversal velocity components.

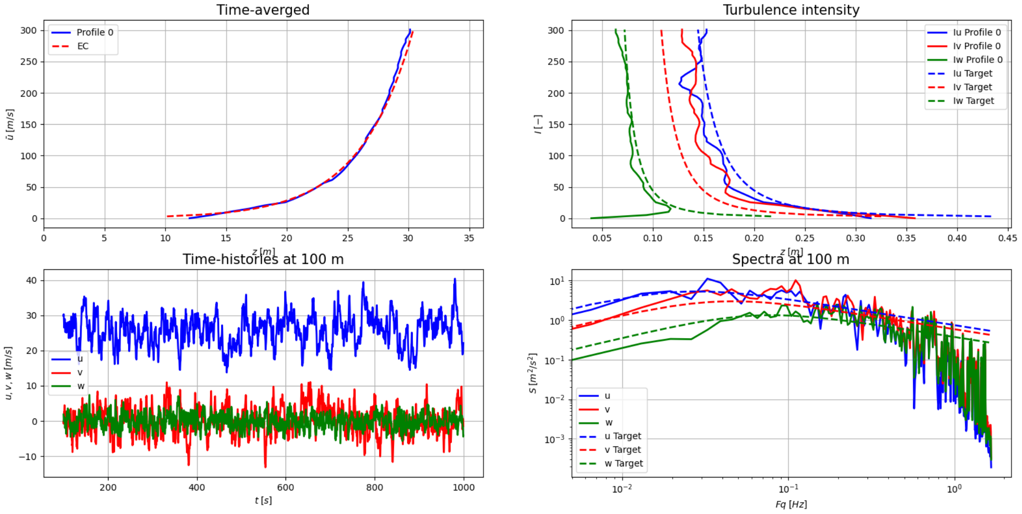

Figure 6: Overview of the results obtained for the *ABLInflow* in terms of velocity field with VBIC.

We now consider the evolution of the velocity field in the along-wind direction. Figure 7 provides an overview of the turbulence intensity and pressure fluctuations evolution in the along wind directions, measured at an height of 100 m. The following aspects are noticed:

- as expected pressure fluctuations start form a much lower values when adopting VBIC corrections;
- pressure fluctuations observed when using VBIC quickly adjust to a constant value typical of non spurious pressure fluctuations;
- when no corrections are adopted, spurious pressure fluctuations decrease rapidly and arrive to the expected value at a distance approximately equal to $2L$, where L is comparable to the integral length scales [2];
- when using VBIC, turbulence intensity has a monotonic trend, in this case we observe a decay due to mesh dissipation and an increase of I_w due to energy transfer from the other components;
- in this case VBIC corrections are acting mainly on the v velocity component, increasing the I_v value since the inflow patch;
- conversely, when VBIC corrections are not used, the along wind evolution is not monotonic, with I_u sharply increasing just downstream the inflow patch.

Overall, it shall be thus considered that whenever the inflow does not kate into account (linearized) Navier-Stokes equations and BCs, pressure fluctuations and an adjustment of the velocity field must be always expected. VBIC allows to strongly moderate the insurgence of pressure fluctuations keeping the along-wind turbulence intensity unchanged. If corrections are not applied, downstream the inflow patch, changes will anyway be applied to the velocity field, generating in addition strong spurious pressure fluctuations.

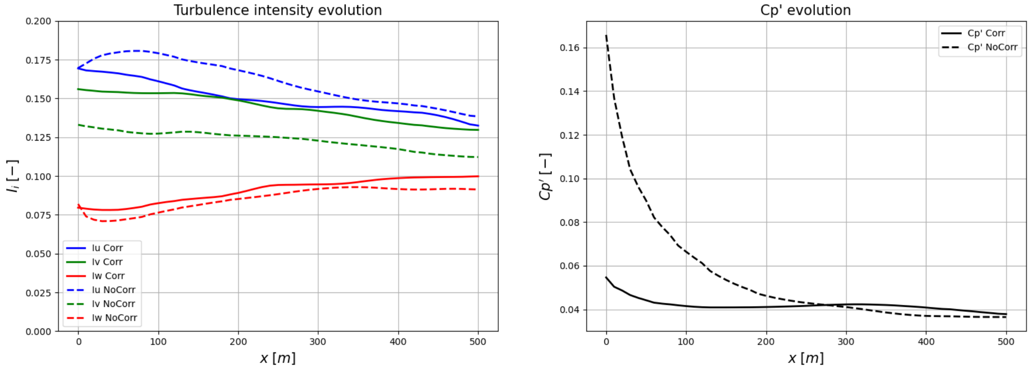

Figure 7: Evolution of the synthetic flow in the along wind direction.

6 Version history

v1.0 - September 1, 2022 - First release.

References

[1] L Patruno and M Ricci. A systematic approach to the generation of synthetic turbulence using spectral methods. *Computer Methods in Applied Mechanics and Engineering*, 340:881–904, 2018.

[2] M Bervida, L Patruno, S Stanič, and S de Miranda. Synthetic generation of the atmospheric boundary layer for wind loading assessment using spectral methods. *Journal of Wind Engineering and Industrial Aerodynamics*, 196:104040, 2020.

[3] L Patruno and S de Miranda. Unsteady inflow conditions: A variationally based solution to the insurgence of pressure fluctuations. *Computer Methods in Applied Mechanics and Engineering*, 363:112894, 2020.