Development Methodology for Immersive Home Laboratories in Virtual Reality

Visualizing Arbitrary Data in Virtual Reality

https://doi.org/10.3991/ijoe.v18i14.35099

Konrad Boettcher(✉), Alexander Behr, Claudius Terkowsky
TU Dortmund University, Dortmund, Germany
Konrad.Boettcher@tu-dortmund.de

Abstract—Although fluid mechanics should be an easy area of physics to learn due to the ubiquity of flows, many students already struggle to understand the basics. Unlike hands-on labs, this is where virtual labs can provide additional interactive visualizations to better facilitate and promote understanding of the fundamental phenomena. However, how can such laboratories be developed for engineering education relatively easily? This contribution presents a simple technical methodology to convert laboratory experiments into a virtual home laboratory. The basis is the representation of arbitrary data in virtual reality. The authors show on the example of a fluid mechanics laboratory experiment a simple numerical method to calculate the flow by using the so-called fictitious domain approach, a procedure to import CFD-calculated fluid-mechanical data into the Unreal engine 4 originally made for development of computer games. As a game engine, it offers high performance so that the virtual environment runs smoothly on ordinary computers without need of special hardware, so also VR-goggles are optionally. In the home laboratory, the flow can be influenced by setting the pressure gradient and we show how to display vector fields, particle motion, residence time, streamlines, information panels, and scalar fields in 3d representation and in arbitrarily set cut planes. Since many of the features cannot be displayed in a real lab, this contribution reflects some possibilities of the design principles of industry 4.0 in a virtual lab.

Keywords—virtual laboratory, virtual reality laboratory, fictitious domain method, Unreal engine 4, home laboratory, data visualization

1 Introduction

Students in the beginning semesters always have difficulties in understanding already the basics of fluid mechanics. Even if the lecturers make great efforts, a fundamental problem seems to be how to understand something that eludes visualization and is only to be understood indirectly via applied mathematical abstractions. The visualization of the invisible by means of interactive virtual reality can offer new possibilities for promoting understanding here [1], [2], [3]. But how can such labs be developed for

engineering education with relative ease? Moreover, while remote lab development can become an extremely expensive and laborious affair – depending on the discipline and equipment [4] – game engines offer novel low-cost lab design options.

To this end, the authors present a simple approach to develop an interactive home laboratory experiment. A conventional on-site hands-on experiment used in the engineering education of fluid mechanics is digitalized by engaging the simple fictitious domain method to computational fluid dynamics (CFD) simulations. The methodology in this work provides a way to develop an immersive virtual reality (VR) laboratory as used and evaluated in engineering education [5], [6]. In this context, the developed virtual environment can be considered as a digital twin of a laboratory experiment. Virtual laboratory experiments easily enable the utilization of the organizational design principles of Industry 4.0 of technical support, connectivity, transparency of information and trains students in the handling of them [7]. These can be used to reduce extraneous cognitive load [8], which acts like a pedagogical reduction [2]. According to [9], virtual laboratories offer four additional advantages:

- The abstract becomes experimental
- It is possible to do something actively instead of just passively observing it
- Elaborate things can be realized easily, and
- The boundaries of reality can be broken

Especially fluid mechanics education often uses complex CFD software, which students can only learn with great effort. Therefore, lecturers mainly present results using static presentations such as pictures or videos. This can be a problem, especially in the case of 3d and/or time-dependent displays. The virtual laboratory discussed here enables users not only to walk freely (i.e., change their point of view on data) but also to interact with the content presented. Thus, the immersion into the results is increased enabling a better understanding [7], [10].

The presented methodology is suitable for immersive visualization of any measured or calculated data and is not limited to fluid mechanics and (higher) education. An industrial usage would reduce the time required for the transmission of results at the business unit interfaces as no lengthy training is necessary in contrast to usual post-processing software. In addition, with such pedagogically reduced laboratories, interesting results can be made available to a wider audience, since less technical and scientific knowledge is required.

Several authors developed virtual environments for visualizing CFD results, and their features are summarized with the features of this work in Table 1. The authors of this contribution aim at a broad applicability especially for students so that low-end computers should be sufficient to run the home laboratory. Therefore, data creation and visualization must be decoupled in order not to perform CFD calculations in real time. While this is a limitation by enabling for less interactivity with the flow by the users, it also poses the advantage to use the same method for more complex flows like multiphase flows or compressive shocks which are computational very expensive and even stationary calculations on a supercomputer can easily last a few days. Real-time CFD would be even more expensive and therefore not expected on low-end computers even in the future. Nevertheless, to obtain real feedback from the manipulations of the

flow, the numerically simple particle motions are derived in real time from the a-priori computed velocity vector fields. However, the virtual environment (VE) created in this work only uses flow data calculated prior to compiling the VE. As a result, the performance does not change with higher complexity of the CFD calculation. How can this be developed in concrete terms?

To this end, in section 2 we firstly present the real experiment to be digitalized. In section 3 a simple way to perform CFD simulations so that the results are directly usable in the VR engine Unreal is shown. In section 4 we discuss how to realize the targeted features of Table 1 in Unreal. In section 5 we test the performance of the virtual laboratory on different hardware systems to ensure the capability of bring your own device, so that all users can use the virtual laboratory. Finally, a conclusion and outlook is given in section 6, which summarizes and critically classifies all the main points of the development methodology.

Table 1. Features of different virtual reality applications for fluid mechanics

	Berger and Cristie, 2015 [11]	**Yan, 2017 [12]**	**Yudin et al. 2019 [13]**	**This Work**
Field lines (streamlines)	X	X	–	X
Information panels	–	–	–	X
2d-scalar fields	X	–	–	X
3d-scalar fields	–	–	(X)	X
Particles	–	–	–	X
Residence time	–	–	–	X
Flow manipulation	X	–	X	X
Software	Unity 3D	Autodesk	Unreal	Unreal
Immersive data calculation	– (a priori)	– (a priori)	X (real time)	(X) Fields: a priori Particles: real time
Low-end hardware compatibility	–	–	–	X
Arbitrary Flows	X	X	–	X

2 Conventional laboratory experiment

In the mandatory course "Introduction into Chemical Engineering" undergraduate students perform a simple experiment. The experiment was created as part of a master's thesis and took six months to design, simulate, fabricate, and set up. A fluid flow is visualized, and the students have to analyze the behavior of the flow. In the experiment, the flow of water through a pipe with a sudden expansion (indicated with red dashed line, so-called Carnot-diffuser) is visualized in which one wall is transparent (see Figure 1).

To make the flow visible, tracers are added to the flow. Since the particles obscure each other and only the top layer is visible, the entire flow cannot be observed. Therefore, the flow is realized by a flat channel, which however has differences to the flow

through a real Carnot diffuser. The velocity of the flow can be adjusted continuously. For the interpretation of flow physics important quantities like pressure and velocity fields cannot be observed and can hardly be measured even with great effort. Unfortunately, air frequently enters the flow through leaks and bubbles appear. These collect on the transparent pane and thus additionally cover large parts of the flow area, as can be seen in the right part of the Figure 1. Groups of eight students investigate the flow about one hour but cannot repeat it at home when reflecting on the results.

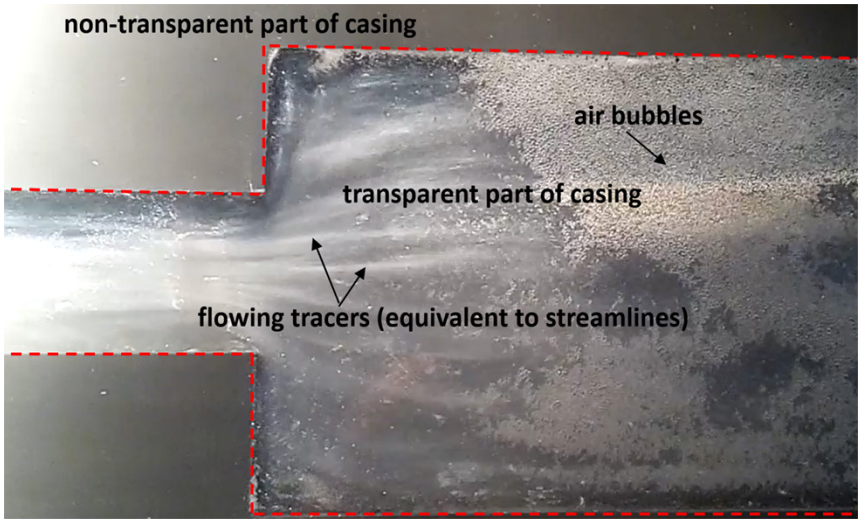

Fig. 1. Observable flow in real experiment, seeded with tracers and undesired air bubbles

3 Numerical methodology

This section shows a simple method of setting up CFD simulations so that the results can be easily imported into the VR engine being used. The general methodology is shown in Figure 2 and gives an overview of the procedures for the specific features. The data is generated using a CFD software and the data generated in post-processing is prepared for import into the VR engine mainly using MATLAB and BLENDER. In the CFD-based data generation, the authors have utilized the fictitious domain method [14] as it yields results in a structured, equidistant grid, which can directly be used in UNREAL without utilizing an interpolation procedure. In this method non-perfusable volume elements are created by giving the element the highest possible flow resistance.

This method provides surprisingly accurate results, as in the determination of pressure loss in fibrous filters [15, 16].

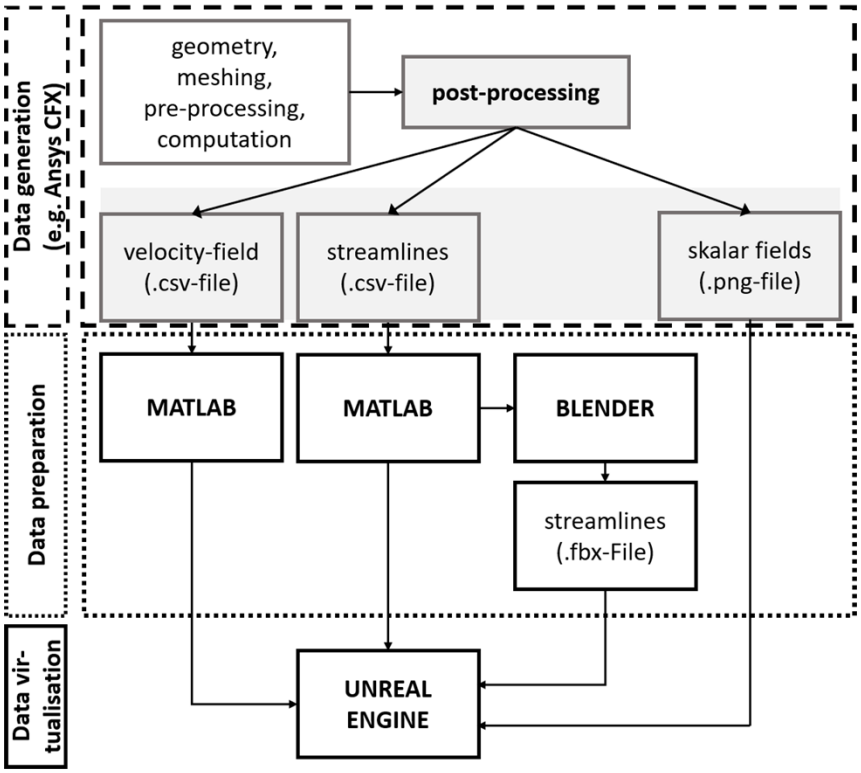

Fig. 2. General methodology for laboratory digitization

If the computation is not performed in a cuboid geometry on a structured mesh, the results must be converted in post-processing or in data adaptation, e.g., with Voronoi interpolation [17].

Figure 3 shows the dimensions of the entire cuboid computational domain and the geometry of the Carnot diffuser. The dimensions of the entire domain are 30×30×150 mm. The meshing is done with a spatial resolution of 16×16×76 nodes, which is sufficient for demonstration purposes. The utilization of the symmetry conditions for an easier import into the so-called Development Interface of Unreal engine is omitted. The suitable boundary conditions instead are the no-slip condition at the walls of the computational domain and at the inlet an inlet condition with the static pressure p_{in}. At the outlet an opening with an entrainment was chosen as boundary condition to model possible backflows in case of non-developed flow for higher pressure gradients.

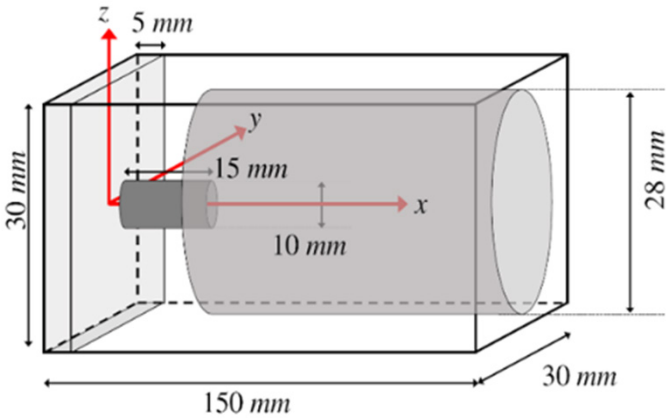

Fig. 3. Configuration and dimensions of the 3d computational domain

The commercial CFD software Ansys Cfx is used. A porous subdomain is created in the entire cuboid computational domain. In this subdomain the porosity is set to $\varepsilon = 1$. This solves the equations for mass transport and momentum transport for an incompressible, Newtonian fluid in laminar flow:

$$\nabla \cdot \vec{w} = 0 \tag{1}$$

$$\varrho\left(\frac{\partial \vec{w}}{\partial t} + \vec{w} \cdot \nabla\left(\vec{w}\right)\right) = \eta \Delta \vec{w} - \nabla p + \varrho \vec{s} \tag{2}$$

with the velocity vector $\vec{w}$, density ϱ, pressure p, time t, dynamic viscosity η and momentum source/sink term $\vec{s}$. Flow resistance in the porous medium is implemented via the momentum sink $\vec{s}$:

$$\vec{s} = \frac{\eta}{k(\vec{x})}\vec{w} \tag{3}$$

with the spatial-dependent permeability $k(\vec{x})$. The momentum sink must vanish $(k \to \infty)$ for a perfusable volume. The momentum sink must $\vec{s} \to \infty$ in the non-perfusable volumes of the cuboid which is realized by $k \to 0$ and the velocity vector gets $\vec{w} \to 0$. This procedure is transferable to any simulation software. Especially for Ansys Cfx, the momentum sink must not be infinity in the first cell at the inlet to ensure numerical stability. Therefore, the flow-restricted volume elements are shifted by $\Delta z = 5$ mm. This corresponds to the condition:

$$k(\vec{x}) = 10^{-30} + \begin{cases} 10^{30}, \text{if} & z \le 5\,\text{mm} \\ 10^{30}, \text{if} & 5\,\text{mm} \le z < 15\,\text{mm} \wedge x^2 + y^2 \le r_1 \\ 10^{30}, \text{if} & 15\,\text{mm} \le z < 150\,\text{mm} \wedge x^2 + y^2 \le r_2 \end{cases} \tag{4}$$

and corresponds to a Carnot diffusor with the cross-sectional jump from the radius r_1 = 5 mm to r_2 = 14 mm. The spatial dependence of the momentum sink is specified in Ansys Cfx using the following Expression:

```
Permeability = if(z >= 0.005[m],
    if(z <= 0.02[m],
      if((x-0.015[m])^2+(y-0.015[m])^2 > SmallDia^2,
          Null,10^30),SectJump),10^30)[m^2]
SectJump = if(z<=0.148[m],
      if((x-0.015[m])^2 + (y-0.015[m])^2 >ExtDia^2,
          Null,10^30),10^30)
Null = 10^-30
ExtDia = 0.014 [m]
SmallDia = 0.005[m]
```

An advantage of this method is the modification of perfusable geometry by simply adjusting $k(\vec{x})$ in equation (4). The actual geometry of the computational domain, meshing, post-processing, and the data transfer procedures for Unreal engine can be adopted without modification, whereby the interpolation step is completely omitted.

The results are exported in different ways, using the Ansys Cfx post-processing tool CFD-Post. The velocity field is exported together with the corresponding position vectors as a .CSV file, so that a clear assignment between position in the Cartesian coordinate system and direction of the vector is possible. The export is done with the post-processing program by File→Export→Export… and by selection of the position coordinates x, y, z and the components of the velocity vector components u, v, w.

Sectional views are created in the entire geometry at Δz = 1 mm intervals. As an example, the scalar fields of the absolute value of the velocity vector and the pressure are exported as .PNG files. Streamlines are plotted in post-processing and exported via Export→Streamline as splines consisting of the position coordinates of the points of a streamline. Those are used later to create Static Meshes to be displayed in the VE (c.f. section 4.4).

4 Realization in Unreal engine

The movement in the VE is controlled by mouse and keyboard of the PC. The keyboard controls the movement of the user in the VE and the mouse controls the direction of view. The free, spatial movement of the user is also made possible in vertical direction by a flight mode. Figure 4, left shows the opaque geometry of the Carnot diffusor in the VE. It is made of Static Meshes and provides a complete view of the perfused geometry and can also be entered and walked through. Switches in front of the geometry allow to choose between three different pressure gradients between inlet and outlet of the geometry. They also toggle the visibility of scalar fields as velocity field or pressure field.

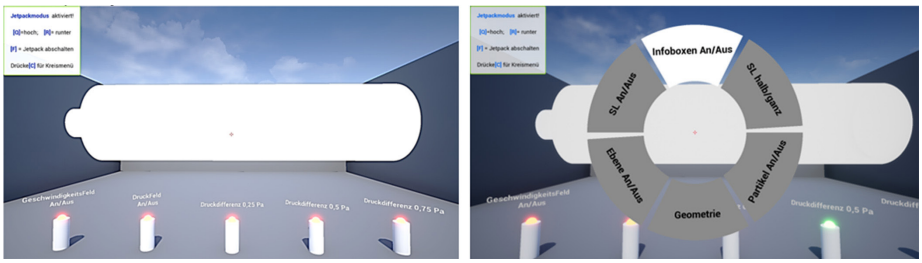

Fig. 4. (left) Opaque geometry with switches to adjust the pressure difference and display of the corresponding pressure and velocity fields. (right) Interactive radial menu of the VE

An interactive radial menu [18] can be activated by keyboard and operated with the mouse. It provides additional interaction possibilities (see Figure 4, right). By selecting different segments in the radial menu, the material of the perfused geometry can be switched between visible and invisible by assigning different materials to the Static Mesh. In addition, particle emitters set up via the Unreal engine internal editor Cascade as explained in more detail in section 4.1 are activated and deactivated by one of the segments. Furthermore, 3d streamlines can be switched between visible and invisible by teleporting the respective Mesh to its position within the perfused geometry and teleporting every other streamline-Mesh to an arbitrary position not visible to the user. Another option of the radial menu can be selected to toggle the display of streamlines for the entire or halved perfused geometry resulting in a better visibility of the internal flow. Finally, a plane can be placed either arbitrary or fixed in space (up to further user input), onto which the current scalar fields are then projected resulting in a cut plane within the digital twin. The following sections present the methods by which the options selectable in the radial menu can be produced.

4.1 Vector fields

Vector fields are used to implement particle movements in the VE where the particles ideally follow the flow. This allows a simple, intuitive visualization of the flow pattern without obstruction as in the real experiment. Before importing the data into the Development Interface, the data should be prepared to improve the performance in the VE, for which simple Matlab scripts are sufficient. The Development Interface supports the import of 2d and 3d velocity vector fields in a structured, cuboid grid by importing FGA (Fluid Grid Ascii) files. These files are comma-separated value tables and consist of a ‘header’ and a ‘body’. The maximum size of such a velocity field supported in Unreal is limited to about 2 million elements with a resolution of 128 × 128 × 128, but several adjacent vector fields can be used. The structure of such a .FGA file is shown in Table 2.

Table 2. Exemplary structure of a .FGA file for generating a vector field

	Content	*x*-Coordinate	*y*-Coordinate	*z*-Coordinate
'Header'	Number of elements	16	16	76
	Start coordinate	0	0	0
	End coordinate	160	160	760
'Body'	Vector to element 1	–4.37E-04	–4.16E-04	4.80E-01
	Vector to element 2	2.05E-03	–1.46E-02	6.90E-01
	Vector to element 3	4.83E-02	–2.49E-02	7.26E-02

The first three lines of the header for each spatial dimension show the total number of elements and the start and end coordinates of the spatial extent of the grid. The start and end coordinates can be freely selected and form the cuboid space of the vector field. In the exemplary setup shown here, the start coordinate of the vector field is set to be at the origin. The ending coordinates are set to the 10-fold value of the respective number of elements. As the values set here correspond to centimeters in the Unreal engine, the distance between the vectors is (160 cm – 0 cm)/16 = 10 cm and the overall vector field set by the exemplary FGA has the dimensions of 1.6×1.6×7.6 m. The spatial extension in the Development Interface and in the VE corresponds to the perception of an adult human being. Thus, a new scaling of the vector field within the Unreal engine is not necessary as it is big enough to be investigated. Afterwards the velocity vectors are listed element by element. The individual columns describe the *x*-, *y*- and *z*-components of the velocity vector for each element. Figure 5 shows the imported data values of the .FGA file are assigned to a position in the structured grid of the vector field via the element number in the Development Interface of the Unreal engine, whereby a fixed assignment scheme must be followed. In this way, the elements of the structured grid are first processed in *x*-direction, then in *y*-direction, and finally in *z*-direction.

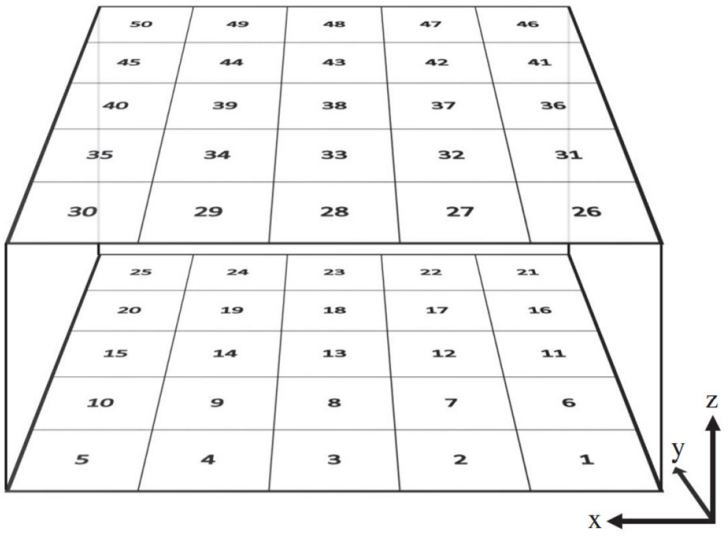

Fig. 5. Spatial arrangement of elements of FGA-vector fields in Unreal engine (5×5×2 vector field)

The order in Ansys CFX does not strictly follow this scheme (cf. Figure 6, left). In this, the coordinates of the position of the vectors are plotted against the node number. Blue corresponds to the *x* coordinate, orange to the *y* coordinate, and gray to the *z* coordinate. At the top of each component a representation of all components can be found in one graph, below the individual representation of the directions *x*, *y* and *z* are depicted. The file exported from Ansys Cfx post-processing does not contain a continuous string of node numbers, which can be recognized by the irregular jumps of the location vectors in the last third of the node numbers.

To obtain the arrangement of the nodes usable for Unreal engine, the required formatting is created using a Matlab script that automates header creation, vector field formatting, and conversion to the .FGA file format.

After importing the simulation data (vector fields) generated with Ansys CFX as .CSV files, the velocity vectors are sorted according to the position in the directions. These data consist of a matrix and in each row the location vectors and velocity vectors. This results in a n × 6 matrix for the velocity vector field generated with Ansys CFX.

The Matlab command Sortrows is then used to sort the matrix row by row in ascending order, first according to *x*- then *y*- and finally according to the *z*-direction of the position vector of the velocity vector. Thus, the velocity vectors are also sorted spatially. The position vectors are omitted for the implementation in Unreal engine wherefore the correct sorting is necessary. The Matlab code used is:

```
% Import of position and velocity vectors
% obtained from Ansys as Matrix with 6 columns
    A = csvread(NameDat,6,0);
    B = sortrows(A,1); % Sort for x
    C = sortrows(B,2); % Sort for y
    D = sortrows(C,3); % Sort for z
% Save velocity components u,v,w matrix G
    G = [D(:,4),D(:,5),D(:,6)];
% Create header matrix
      Header = [n_ElementsX, n_ElementsY, n_ElementsZ;
                0, 0, 0;
                10*n_ElementsX, 10*n_ElementsY,
                10*n_ElementsZ]
% Join header matrix with body matrix
    Export = [Header;G(:,1:3)];
% Save matrix as .csv file
    csvwrite(csvFileName,Export,0,1);
```

Figure 6, left shows that the storage of the generated data is not geometrically coherent, even with using the commercial software Ansys CFX. Figure 6, right shows the components after sorting. It is irrelevant whether the values for the *y* axis (Figure 6, left, orange) or the *x* axis (Figure 6, right, blue) are passed through first. It is also

irrelevant whether the traversal direction is negative (Figure 6, left, gray) or positive (Figure 6, right, gray). Implementing non-coherent data in Unreal leads to unpredictable behavior of the particle movement due to the abrupt change of the components of the velocity vector in the last quarter of Figure 6, left. The storage structure of the position coordinates with the assigned values of the flow variables must be done in the described manner, which is shown in Figure 6, right after the re-sorting. The arrangement of the position vectors is now stringently sorted by the scheme presented in Figure 5, with firstly the *x*-, secondly the *y*- and lastly the *z*-component. The number of position vectors in each of the three spatial directions is manually specified in the Matlab code. Using the distance of 10 cm between the position vectors in the code, the start and end coordinates are determined automatically, and the header is generated. The locations belonging to the velocity vector are assigned in the Unreal engine via the dimensions and resolution of the vector field.

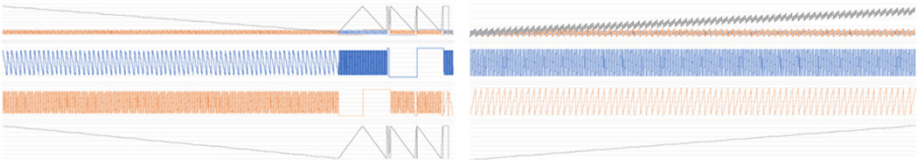

Fig. 6. Structure of the exported .CSV files (left) and the sorted .FGA files (right)

The previously sorted 'body' of the velocity vector field is merged with the 'header' by the Matlab code. Then, the so generated matrix of header and body is saved as $(n + 3) \times 3$ matrix with the number of location vectors n by the Matlab code as a .CSV file and renamed to an .FGA file. The file can be imported and saved in the content browser of the Development Interface of Unreal engine. Thus the .FGA file is recognized as a vector field into the Development Interface.

Vector fields imported in this way are not visible in the compiled VE. Particle emitters seed particles into the VE at defined positions and visualize the vector field by particle motion. The movement of the particles in the vector field is then calculated by the GPU and displayed in the VE. They follow the interpolated vectors of the imported vector field at the respective position in the VE. The particle class editor Cascade in Unreal allows the use of so-called Global Vector Fields (c.f. Figure 7) and Local Vector Fields.

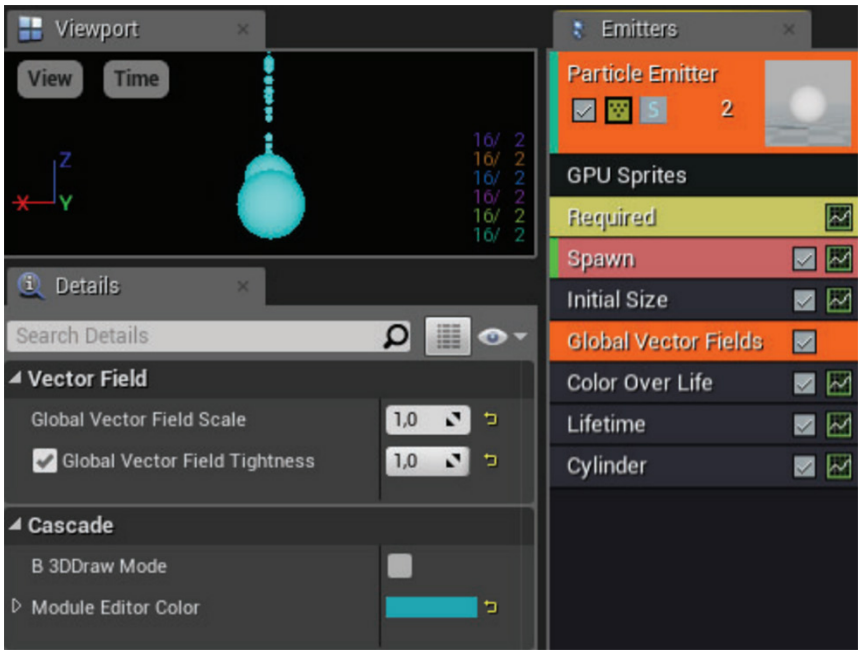

Fig. 7. Settings for particle emitters in Unreal engines particle editor Cascade

Particle emitters with local vector fields do not allow to change the vector fields while being in the VE. Thus, a particle always follows exactly one constant vector field. With global vector fields a particle can be influenced by different vector fields. The particle movement result from the vector field in which it is currently located.

Global vector fields enable the realization of larger vector fields by dividing them into several 128 × 128 × 128 sized vector fields. Thus, the movement of existing particles changes with a change of the vector field. This enables the influence on the particle movements in the VE.

The emitter region is defined as a circular disk-shaped area with the same diameter as the tube before the cross-sectional jump via the so-called Cylinder module and then placed at the inlet of the flow. To visualize the separation eddies, an annular emitter region is placed at one third of the total length of the diffuser. This annular region has the same outer diameter of the outlet pipe and an inner diameter of about 2/3 of the outlet pipe.

The parameter Global Vector Field Tightness adjusts the slip of the particles. This defines the follow-up capability of the particles in relation to the fluid. A value of 1 corresponds (fluid-mechanically) to a particle Stokes number St = 0 and thus to an ideal follow-up behavior. Components and flow fields can be scaled in the Development Interface by extending the dimensions of the geometries and vector fields. The parameter Global Vector Field Intensity corresponds to an amplification factor for the values of the velocity field. This can be used to compensate for perceived changes in velocity caused by scaling. The two parameters mentioned here were set to 1. As a further parameter of the particle emitter the Spawnrate equals five and thus sets the number of

particles spawned per frame. The Global Lifetime module sets the lifetime of the particles to 300 seconds to limit the number of particles visible at a time. This ensures that enough particles are displayed at the same time to visualize the flow in an intuitive way while reducing the workload on the GPU. Using the module Color over life a color gradient can be set for the particles, so that the particle color is determined by the time of the particle in the perfused geometry. The dependence of the particle color on its lifetime is set by the Particle Emitter module Color Over Life. A Distribution Vector Constant Curve is used, which shows a linear course from colored (input value 0) to black (input value 1).

The so generated massless particles are displayed as GPU-rendered sprites. Since a sprite is a 2d image that is always displayed orthogonal to the viewing direction, visualizations of particles with tails, e.g., to represent a trajectory line, cannot be realized in a straightforward way with this method.

4.2 Scalar fields

Scalar variables such as pressure provide further possibilities to gain an understanding of fluid physics. The graphical representation of the scalar quantities by means of a color scale is common as it enables a fast, qualitative impression of the scalar field. However, this is only done in a sectional plane of the flow. This section describes how such a graphical representation of scalar fields and an interaction with them in the VE can be realized: In post-processing, ordinary contour plots are created, where values of scalar quantities are transferred into a color scale.

To generate the images, Ansys Cfd-Post was used to display the pressure contours on a *xz*-plane in the direction of the main flow (directions of axes, cf. Figure 3) and the command Save Picture… to save them as .PNG image files. By moving the plane along the *y* axis, 31 images of the scalar field are saved per simulation, imported into the Development Interface and placed there as so-called Decals. For this purpose, Deferred Decals are used and each Decal is assigned to one of the 31 images. These Decals have a cuboid volumetric extent and project the image assigned to them onto the surface of objects with a MESH that are located within the volume of the Decal. If there is no object with an assigned Mesh in the Decal volume, the Decal is invisible.

The corresponding lengths of the diffusor in the spatial directions of the *xz*-plane is assigned to the Decal, while the width of a Decal in *y*-direction corresponds to a 31st part of the diffusor width. The Decals are placed that their images represent the corresponding *xz*-planes of the scalar fields at the corresponding location of the *y*-direction from the simulation results. If there are objects inside a Decals, they get the image file as the material property and thus show the contour plot of the scalar field. The same procedure is utilized to the magnitudes of the velocity vector fields. Any other scalar field can also be imported and displayed. This means that the engine does not require calculation from a database, which improves performance for low-end computers. Since only ready-made images are projected, no actual data of the scalar fields are included in the program.

The command Create Dynamic Instance creates a dynamic material instance, which allows to modify the material properties in real-time without performing a new shader

rendering. This dynamic material then is accessed in the coding of Unreal. The coding of the VE is facilitated in the so-called Level Blueprint, which provides an easy-to-use graphical programming of the VE. Pre-defined material parameters can be changed using the commands SET Texture Parameter Value and Set Material. Each Decal is assigned such a material so that the appropriate scalar field can be displayed volumetrically. Since the Decals cannot be seen directly in the VE, a plane can be spawned and moved by the user in the VE. For this purpose, the vector of the user's line of sight and the function Linetracebychannel are used. This function checks which visible objects are hit in the line-of-sight vector and outputs the object as a Hitresult. Afterwards a test is performed if the hit object is the floor of the VE. If this is the case, a plane object, which is made of a Cube mesh with the length of the diffusor geometry, is teleported to this position on the floor to provide a plane within the Decals. In addition, the plane is rotated, so that it is always displayed perpendicular to the line-of-sight vector. For easier handling of this plane, the possibility to display the plane only along the main flow direction of the extension or perpendicular to the main flow direction is also implemented. This is done by comparing the magnitudes of the x- and y-components of the line-of-sight vector. If $x \geq y$, the plane is rotated by 90° along the z-axis, otherwise, no rotation will be performed.

Thus, a plane can be placed arbitrarily in space orthogonal to or along the direction of the user's line of sight by selection in the radial menu. This plane is teleported to the desired position (i.e. the direction in which the user points) and can be set to not moving by the user. The Deferred Decals placed in the flow region project the images corresponding to the scalar fields onto the placed plane. This allows to display the pressure or velocity contours depending on the position on the placed plane and to set the position of the plane. Figure 8, left shows an arbitrary set position of the plane within the opaque perfused geometry and Figure 8, right the same plane but with velocity visualized at the highest pressure-difference between inlet and outlet (here: red – high pressure and blue – low pressure).

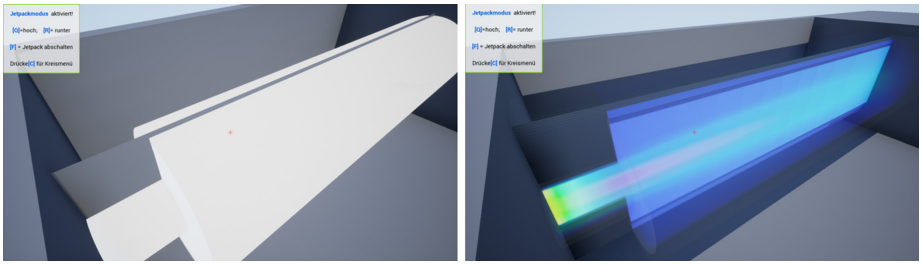

Fig. 8. (left): Arbitrary position of cut plane with opaque geometry and (right) absolute value of velocity vector on that cut plane

A likewise novel representation of such scalar fields is realized by volumetric images. Like the Decals, 31 cuboid objects with an assigned Cube Mesh are placed in the Development Interface. The material of the cubes is then assigned to the corresponding scalar field according to their position. By using a translucent material, a hologram-like, 3d scalar field is created and the entire scalar field can be viewed in the

VE at once. Figure 9 shows the volumetric representations of the scalar fields for the pressure field (left) and the field of absolute value of the velocity vector (right).

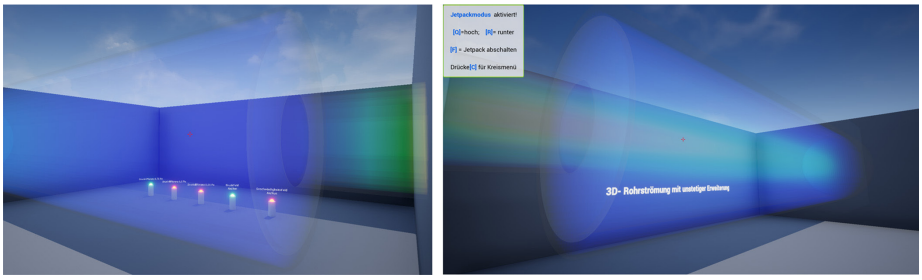

Fig. 9. Volumetric representation of scalar fields (left): pressure and (right): velocity. Blue color indicates low values, while red color represents high values

4.3 Displaying additional information

To display additional information (e.g., explanations of local flow properties or boundary conditions, c.f. Figure 10) Widget blueprints are used. These 2D, inscribable planes are faded into the user's field of view, the so-called 'Heads up Display' (HUD), and can thus provide additional information. They are therefore visible and readable regardless of the position of the user. By using a global HUD blueprint class, the individual widgets are switched on and off or the content of these widgets is changed by the user through interaction with the VE. The widgets used here mainly use images and text blocks, which can also be varied in real time via blueprints.

They are inserted via the command Add User → Interface → Widget Blueprint and have their own blueprints. In addition, the Designer window of the Widget editor can be used to change the appearance of the blueprints, e.g., to include text fields or graphics on the displayed screen in the VE. A global HUD class creates instances of the required widgets at the beginning of the game and assigns them to the user actor. With the command Get All Widgets Of Class a widget is selected in the level blueprint and the previously defined commands of the widget class can be implemented.

4.4 Streamlines (arbitrary field lines)

The simplest way to display field lines (streamlines) is to display streamlines like scalar fields via an image export from the post-processing. 3d streamlines (c.f. Figure 10) however enable a better insight into more complex flow patterns. The color variations of the streamlines are generated by the Decals of the scalar fields for the magnitude of the velocity placed in space.

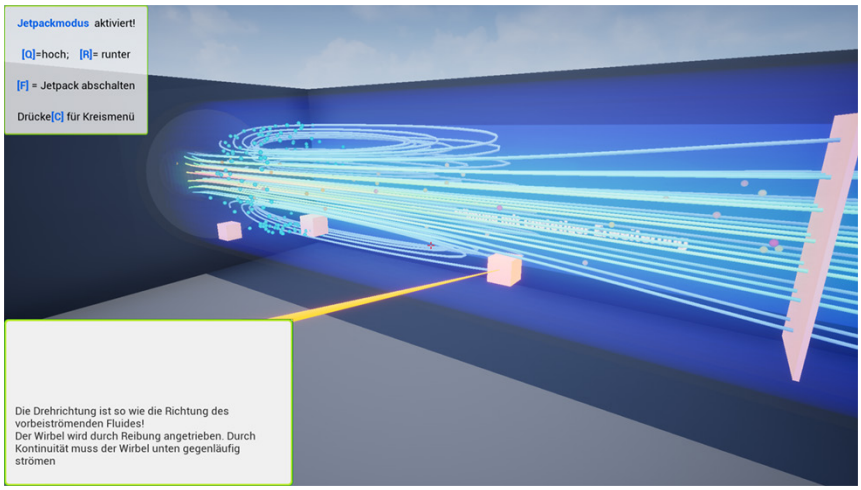

Fig. 10. Display of additional information and streamlines

The streamlines of the central jet are exported from ANSYS CFX as single csv file. For the illustration of the flow inside the eddies several single streamlines are created and exported separately as csv files. The streamlines are then built to a mesh file and exported as FBX file using the graphic program BLENDER (v. 2.79). This is achieved using the script to import a csv file and create meshes tool allowing for import of csv files creating polygon curves. Importing all streamlines exported for one pressure difference, this yields one complex STATIC MESH of streamlines. This then is exported as mesh file, which in turn then gets imported into UNREAL engine [19].

5 Performance

The intended use-your-own-device was implemented in the home laboratory, as shown in Table 3 based on the performance even for low-end laptops. The low-end laptop has already been used for virtual laboratories in the exercise. The low number of frames per second (FPS) is not an issue due to the lack of fast movements of the virtual character.

Table 3. Comparison of performances of the VE on different hardware

		MiFCOM (high end)	**Dell Latitude 3510 (mediocre)**	**Acer Spin SP111 – 32N (low end)**
Hardware	Main processor	Intel Core i7-11800H @ 2.3 GHz 8 cores (16 logical)	Intel Core i5-1021U @ 1.6 GHz 4 cores (8 logical)	Intel Pentium N4200 @ 1.1 GHz 4 cores (4 logical)
	RAM	64 GB	16 GB	4 GB
	GPU	GeForce GTX 2070	Integrated Intel UHD	Intel HD Graphics 505 (shared memory)
Consumption	CPU	5 %	25 %	10 %
	RAM	300 MB	900 MB	870 MB
	GPU	35 %	93 %	97.5 %N
Performance		30 FPS (cut-off)	25 FPS	12 FPS

6 Conclusion and outlook

In this work, a simple approach to develop an interactive home laboratory experiment is presented in detail. A conventional on-site experiment is digitalized by engaging the simple fictitious domain method to CFD simulations. The codes for data preparation for the import into the Unreal engine 4 are discussed in detail. The developed immersive laboratory experiment for visualizing the fluid flow through a Carnot diffusor in virtual reality yields several advantages to the conventional on-site experiment.

The mere visibility of the particles on the transparent wall and the required approximation of circular tube cross sections by flat channels was completely solved in the virtual laboratory experiment, since the particles are visible in the entire flow area. The accessibility to the pattern of the flow was improved by the display of streamlines. In addition, the scalar fields of velocity magnitude and pressure were displayed in two and three dimensions. These additions are not realistically possible to implement in an on-site experiment. The accessibility was improved, since in the on-site experiment eight students obstruct each other's view and a longer time is required for changing the position of the students. In contrast, in the virtual lab experiment not only is the experiment directly visible to each student and observable from all positions, the geometry flowed through can also be entered and the experiment can be repeated at home at any time, with integrated information panels reproducing the explanations in the visualization experiment. A disadvantage is that only three different pressure gradients were implemented in the virtual laboratory experiment, whereas in the real experiment a continuous adjustment is possible and even turbulent flows can be generated.

In future enhancements, further pressure gradients shall be implemented, and the turbulent flow shall be realized by adding random time-depending fluctuations in the vector field, so that the turbulent behaviour can be observed. These laboratory experiments are to be used and evaluated in the teaching of fluid mechanics.

7 Acknowledgements

The Faculty of Bio- and Chemical Engineering at TU Dortmund University supported the basic project with funds to improve the quality of teaching. The current advancement is part of the project "CrossLab – flexibly combinable cross-reality labs in university teaching: future-proof competence development for a learning and working 4.0" (project number FBM2020-VA-182-3-01130), funded by Stiftung Innovation in der Hochschullehre.

8 References

[1] C. Terkowsky, D. May, S. Frye, T. Haertel, T. R. Ortelt, S. Heix, and K. Lensing (Eds.), "Labore in der Hochschullehre:", wbv Media, 2020.

[2] K. E. R. Boettcher, A. S. Behr, and D. Boettcher, "Virtuelle Realität des Unsichtbaren, Verständnisfördernde Visualisierung und Interaktivierung strömungsmechanischer Phänomene" in: *Labore in der Hochschullehre: Didaktik, Digitalisierung, Organisation*, C. Terkowsky, D. May, S. Frye, T. Haertel, T. R. Ortelt, and S. Heix, Eds. wbv Media [Online], 2020.

[3] Kammerlohr and D. Uckelmann, "A maturity model for the effective digital transformation of laboratories", *Journal of Manufacturing Technology Management*, vol. ahead-of-print, no. ahead-of-print, 2022. https://doi.org/10.1108/JMTM-01-2022-0050

[4] C. Terkowsky, C. Pleul, I. Jahnke, and A. E. Tekkaya, "Tele-operated laboratories for online production engineering education—platform for E-learning and telemetric experimentation (PeTEX)", *Int. J. Onl. Eng.*, vol. 7, no. S1, pp. 37–43, 2011. https://doi.org/10.3991/ijoe.v7iS1.1725

[5] K. E. R. Boettcher and A. S. Behr, "Usage of a virtual environment to improve the teaching of fluid mechanics." *International Journal of Online and Biomedical Engineering*, vol. 16, no. 14, pp. 54–68, 2020. https://doi.org/10.3991/ijoe.v16i14.16997

[6] K. E. R. Boettcher and A. S. Behr, "Using virtual reality for teaching the derivation of conservation laws in fluid mechanics." *International Journal of Engineering Pedagogy*, vol. 11, no. 4, pp. 42–57, 2021. https://doi.org/10.3991/ijep.v11i4.20155

[7] M. Hermann, T. Pentek, and B. Otto, "Design principles for Industry 4.0 scenarios" *49th Hawaii International Conference on System Sciences (HICSS)*, pp. 1530–1605, 2016. https://doi.org/10.1109/HICSS.2016.488

[8] J. Sweller, "Cognitive load during problem solving: Effects on learning", *Cognitive Science*, vol. 12, no. 2, pp. 257–285, 1988. https://doi.org/10.1207/s15516709cog1202_4

[9] M. Slater and M. V. Sanchez-Vives, "Enhancing our lives with immersive virtual reality", *Frontiers in Robotics and AI*, vol. 3. 2016. https://doi.org/10.3389/frobt.2016.00074

[10] J. LaViola, J. Prabhat, A. S. Forsberg, D. H. Laidlaw, and A. van Dam, "Virtual reality-based interactive scientific visualization environments." In: Trends in Interactive Visualization. Springer London; p. 225–250, 2008. https://doi.org/10.1007/978-1-84800-269-2_10

[11] M. Berger and V. Cristie, "CFD post-processing in unity3d.", *Procedia Computer Science*, vol. 51, pp. 2913–2922, 2015. https://doi.org/10.1016/j.procs.2015.05.476

[12] J. Yan, "A case study for using a building information modeling with virtual reality", Masterthesis; School of Architecture, University of Southern California, 2017.

[13] P. V. Yudin, M. I. Fedina, E. V. Strizh, O. V. Yudina, and V. Khomotiuk, "Computational hydrodynamics in air flows modelin: Using the unreal engine based on the numerical solution of the Navier-Stokes equations". In: 2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon). IEEE, 2019. https://doi.org/10.1109/FarEastCon.2019.8934325

[14] K. Khadra, P. Angot, S. Parneix, and J. P. Caltagirone, "Fictitious domain approach for numerical modelling of Navier-Stokes equations." *International Journal for Numerical Methods in Fluids*, vol. 34, no. 8, pp. 651–684, 2000. https://doi.org/10.1002/1097-0363(20001230)34:8<651::AID-FLD61>3.0.CO;2-D

[15] J. Chaudhuri, K. Boettcher, and P. Ehrhard, "Pressure drop in fibrous filter – representative domain size and effect of fibre orientation." *Chemical Engineering Science*, vol. 246, p. 116865, 2021. https://doi.org/10.1016/j.ces.2021.116865

[16] J. Chaudhuri, K. Boettcher, and P. Ehrhard, "Numerical investigation of coalescence filtration: Multiphase flow through fibrous structures." *Separation and Purification Technology*, vol. 257, p. 117853, 2021. https://doi.org/10.1016/j.seppur.2020.117853

[17] G. Voronoi, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. premier mémoire. sur quelques propriétés des formes quadratiques positives parfaites.", *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 133, pp. 97–102, 1908. https://doi.org/10.1515/crll.1908.133.97

[18] VinceBGameDev, "Radial menu basics – Unreal engine 4 how to", https://www.youtube.com/watch?v=LXvIQmpYU8, 2018.

[19] P. G. Hans, "A script to import a csv file and create meshes (for blender 2.5x or later)", https://blenderartists.org/t/a-script-to-import-a-csv-file-and-create-meshes-for-blender-2-5x-or-later/501410, 2011.

9 Authors

Konrad E. R. Boettcher is with the laboratory of equipment design, Faculty Bio- and Biochemical Engineering at TU Dortmund University, Emil-Figge Str. 68, 44227 Dortmund. He holds a PhD in Chemical Engineering and is responsible for laboratory education, and teaching fluid mechanics. He has developed several analog and digital enhancement projects for teaching fluid mechanics. (email: Konrad.Boettcher@tu-dortmund.de).

Alexander S. Behr is with the laboratory of equipment design, Faculty Bio- and Biochemical Engineering at TU Dortmund University, Emil-Figge Str. 68, 44227 Dortmund. He began to work on this project as a student HIWI at the working group of Fluid Mechanics and continued to work with VR laboratories at his new working group. (email: Alexander.Behr@tu-dortmund.de).

Claudius Terkowsky is with the Center for Higher Education (zhb) at TU Dortmund University in Dortmund, Germany. He holds a PhD in education science and is postdoc researcher in the field of Higher Engineering Education. He is a member of the International Association of Online Engineering (IAOE), of the German Association for educational and academic staff development in Higher Education (dghd), and of the HFD Community Working Group Digital Laboratories in Germany (email: claudius.terkowsky@tu-dortmund.de)

Article submitted 2022-09-03. Resubmitted 2022-10-12. Final acceptance 2022-10-15. Final version published as submitted by the authors.