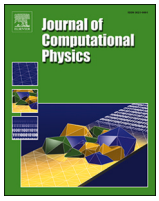
Journal of Computational Physics 542 (2025) 114375

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp



RBF-FD discretization of the Oseen equations

Michael Koch, Sabine Le Borne *

Hamburg University of Technology, Department of Mathematics, Am Schwarzenberg-Campus 3, 21073 Hamburg, Hamburg, Germany

ARTICLE INFO

Keywords:
RBF-FD
Oseen equations
Meshless methods
Pressure constraint

ABSTRACT

The radial basis function - finite difference (RBF-FD) method is a (meshless) technique for the discretization of differential operators on scattered node sets. In recent years, it has been successfully applied mostly to scalar partial differential equations (PDEs). The extension to the application to the steady state Oseen equations on (several) scattered node sets is not straightforward but requires novel components which are the subject of this paper. We consider the steady-state Oseen equations in three spatial dimensions, and as a radial basis function, we restrict ourselves to the polyharmonic spline (PHS) with polynomial augmentation. However, the following contributions of our paper may also be applied to other model problems and RBFs. In particular, we will consider the selection of two node sets for the two types of unknowns, velocity and pressure, and subsequent (flexible order) RBF-FD discretization of the various differential operators in the coupled system. We discuss variants for the discretization of the pressure constraint as well as the influence of the viscosity parameter on the convergence of the RBF-FD discretization. Finally, we provide numerical tests for the Oseen equations in three dimensions on complex domains using several node arrangements, convection directions and parameters inherent to the PHS RBF-FD method. The tests demonstrate that the proposed method is stable for discretization step widths between $h_u = 0.01$ and $h_u = 0.5$ and viscosities in the range of 10^{-3} to 1 not just on the unit cube but also on a more complicated three-dimensional bunny-shaped domain. In particular, for even degrees of polynomial augmentation of the Laplacian (and lower degrees for involved first order differential operators), we can reach convergence of the same (even) order.

1. Introduction

The numerical solution of partial differential equations has traditionally been done using mesh-based methods such as the finite element or finite difference method. In order to circumvent the potentially high computational cost for generating a high quality mesh, many meshless methods have been suggested in the past few decades, e.g. smoothed particle hydrodynamics [1], generalized finite difference method [2] and reproducing kernel particle method [3].

The Radial Basis Function - Finite Difference (RBF-FD) method is another meshless method. First introduced in [4], it has been successfully used for the numerical solution of many PDEs such as the shallow water equations [5], the reaction-diffusion equations [6], unsteady Navier-Stokes equations [7], transport equations [7] and acoustic wave equations [8].

Despite the fact that the RBF-FD method has been widely applied, there exists only relatively little literature regarding the discretization of steady-state fluid flow problems such as the Stokes equations [9,10], the Oseen equations [11] or the steady Navier-Stokes equations [12]. Furthermore, the existing literature mostly treats these equations in only two spatial dimensions or on rather simple domains such as the unit square or circle. Additionally, the discretizations are mostly illustrated for structured node sets. Two

* Corresponding author.
E-mail address: leborne@tuhh.de (S. Le Borne).

https://doi.org/10.1016/j.jcp.2025.114375
Received 27 May 2025; Received in revised form 29 August 2025; Accepted 11 September 2025
Available online 15 September 2025
0021-9991/© 2025 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

exceptions are [9] and [12]: [9] provides three test cases using Halton and random nodes, but the majority of their results is still shown for structured node sets. [12] provides tests on unstructured node sets, which were obtained using a node repel algorithm, however, the numerical tests are performed on the unit square and the stream function formulation of the steady-state incompressible Navier-Stokes equation is used. Further more general studies on RBF-FD for time-dependent Navier-Stokes equations are given in [13–15]. In [13], the Navier-Stokes equations in porous media are treated in three dimensional complex geometries with scattered nodes generated by the algorithm presented in [16]. Both of [14,15] show results in two spatial dimensions on square domains or rectangles with a hole. The solution procedure described in [15] uses different scattered node sets for the velocity and pressure. However, a mesh is required to achieve this[14]. treats the Boussinesq approximation using scattered nodes generated by the algorithm described in [17].

We will discuss a discretization in the primitive variables, i.e., velocity and pressure, as it is also done in [9–11]. Instead of Halton or random point sets, we will use the node placing algorithm proposed in [16]. It generates node sets with a prescribed separation distance which is important for the stability of the RBF-FD method [18]. We will present our results for two different domains: the unit cube and a bunny-shaped polyhedron (see Fig. 3).

In this paper we extend existing work on RBF-FD and introduce a meshless RBF-FD discretization of the Oseen equations in three-dimensional, complicated polyhedral domains using unstructured node sets. Our main novel contributions are the following:

- **Different scattered node sets in coupled PDEs:** The Oseen equations contain two types of unknowns for the velocity field and pressure, respectively, which may be approximated on different scattered node sets. We will study the relationship between these two node sets with respect to their separation distances (and hence total number of nodes) and their distance to the boundary of the domain. We introduce RBF-FD discretizations for the gradient and divergence operators of the Oseen equations which link these two types of unknowns in the resulting stiffness matrix. Choosing different node sets for the velocity and pressure has also been done in [15] where the unsteady Navier-Stokes equations were solved. However, there the discretization required a mesh which is not the case in our approach. In [9], grid-structured staggered node sets as well as (subsets of) pseudo- or quasi-randomly generated node sets have been used.
- **Different discretization orders for different differential operators:** We allow different degrees of polynomial augmentation for the different types of differential operators within the Oseen equations and (numerically) study the effect on the convergence behavior of the discretization. This has previously only been done for the two-dimensional Stokes equations on structured node sets [9].
- **Selection and discretization of the pressure constraint:** In the case of Dirichlet boundary conditions (enclosed flow), the pressure is only determined uniquely by the Oseen equations after an additional constraint is imposed. We will show (numerically) that the type of constraint and its discretization have a significant effect on the convergence behavior of the discretization.
- **Interplay between viscosity and discretization error:** We investigate the dependence of the discretization error on the viscosity parameter (i.e., on the convection dominance) and in Theorem 1 provide a theoretical justification for our numerical observations. To the best of our knowledge, this gives new insight into the behavior of the errors in RBF-FD discretizations of the Stokes and Oseen equations with respect to the viscosity.

In our RBF-FD discretization we will neither use upwind stencils [19,20] nor hyperviscosity [21,22] in order to stabilize the method at low viscosities. This has not been necessary for the viscosities $\nu \in [0.001, 1]$ and problem sizes that we use in our numerical tests and we will leave this aspect for future work.

The remainder of this paper is organized as follows. In Section 2 we review the RBF-FD method and briefly explain how it can be used to (numerically) solve PDEs. Then in Section 3 we derive an RBF-FD discretization of the Oseen equations. We begin with the selection of node sets within the domain for the velocity field and the pressure and derive differentiation stencils for the various differential operators within the Oseen equation (Section 3.1). Then we introduce different ways to select and discretize a pressure constraint (Section 3.2). In Section 3.3, we analyze the solution of the arising linear system with respect to the viscosity. Lastly we demonstrate the effectiveness of the described discretization with several numerical tests in Section 4.

2. The RBF-FD method

In this section, we will briefly review the approximation of linear differential operators with the RBF-FD method and its application in the numerical solution of (scalar-valued) partial differential equations.

2.1. RBF and RBF-FD approximation

We wish to obtain an approximation of a linear differential operator $\mathcal{L}$ acting on a sufficiently smooth scalar-valued function u on a finite set of nodes. For this purpose we define the following node sets.

Definition 1. For a spatial dimension $d \in \mathbb{N}$, an open domain $\Omega \subset \mathbb{R}^d$, $N_I, N \in \mathbb{N}$ with $N_I \leq N$, a set of pairwise distinct nodes $X = \{x_1, \dots, x_N\} \subset \Omega \cup \partial\Omega$, let

$$X_I := \{x_1, \dots, x_{N_I}\} \subset \Omega, \quad X_B := \{x_{N_I+1}, \dots, x_N\} \subset \partial\Omega$$

be subsets of interior and boundary nodes, respectively. Further, for a $K \in \mathbb{N}$, let

$$X_C := \{x_1^c, \dots, x_K^c\} \subset \Omega \cup \partial\Omega$$

be a set of pairwise distinct center nodes.

A canonical choice appears to be $X_C \subset X$ (e.g, $X_C = X_I$ in the case of Dirichlet boundary conditions) but we will see in Section 3 that there is a need for different choices in the case of coupled PDEs with different types of unknowns (and respective node sets).

In contrast to finite difference discretizations, these nodes may be scattered and not connected by a mesh. Hence, we cannot use a predefined generic differentiation stencil that is applied at every node as it is often the case in finite difference methods. Instead, for each center node in X_C, we choose a set of stencil nodes (typically in the geometric neighborhood) which will later be used to approximate a differential operator $\mathcal{L}$.

Definition 2. Let X, X_C be node sets with $|X| = N$ as in Definition 1 and let $n \leq N$. For every node $x_c \in X_C$, let $\mathbb{S}_c \subset X$ be a stencil of size n and with index set $\mathcal{I}_c \subset \{1, \dots, N\}$ associated to the (stencil) center x_c, i.e.,

$$\mathbb{S}_c = \{x_{s_1^c}, \dots, x_{s_n^c}\} = \{x_i \in X : i \in \mathcal{I}_c\} \quad \text{for} \quad \mathcal{I}_c := \{s_1^c, \dots, s_n^c\}.$$

The goal is to find weights $\{w_1^c, \dots, w_n^c\} \subset \mathbb{R}$ such that the weighted sum of function values at the stencil nodes in $\mathbb{S}_c$ approximates the action of the linear differential operator $\mathcal{L}$ on the function u evaluated at the stencil center node x_c, i.e.,

$$\mathcal{L}u(x_c) := \mathcal{L}u(x)|_{x=x_c} \approx \sum_{i=1}^{n} w_i^c u(x_{s_i^c}). \tag{1}$$

The weights are computed such that Eq. (1) holds exactly for certain functions u. To this end we introduce the space $\Pi_\ell(\mathbb{R}^d)$ of d-variate polynomials of degree at most $\ell \in \mathbb{N}_0$ and with a basis $\{p_1, \dots, p_M\}$ where $M := \dim(\Pi_\ell) = \binom{\ell+d}{d}$. We call ℓ the degree of polynomial augmentation. The functions for which we enforce exactness in Eq. (1) are called (polynomially augmented) radial basis functions. RBFs refer to functions that are radially symmetric with respect to a center point, i.e., the function values depend solely on the distance of the evaluation node to the center. In this paper we focus on radial basis functions based on polyharmonic splines (PHS) as they have shown good results in combination with polynomial augmentation in the context of RBF-FD in multiple recent papers [12,23,24]. Although we focus on the PHS, the following is also applicable to other radial basis functions. However, in contrast to PHS, many other radial basis functions possess a shape parameter which must be carefully tuned to achieve satisfactory results.

Definition 3. For a spatial dimension $d \in \mathbb{N}$, an open domain $\Omega \subset \mathbb{R}^d$, a node $x_i \in \Omega \cup \partial\Omega$ and $k \in \mathbb{N}$, we define the PHS(k) radial basis function (RBF) centered at node x_i as

$$\Phi_{x_i} : \Omega \cup \partial\Omega \to \mathbb{R}, \quad \Phi_{x_i}(x) := \|x - x_i\|_2^{2k-1}.$$

Given a fixed stencil center $x_c \in X_C$ with stencil $\mathbb{S}_c$, we next illustrate the computation of weights such that Eq. (1) is exact for all functions of the form

$$u(x) = \sum_{i=1}^{n} \alpha_i \Phi_{x_{s_i^c}}(x) + \sum_{k=1}^{M} \beta_k p_k(x) \tag{2}$$

that satisfy the constraints

$$\sum_{i=1}^{n} \alpha_i p_k(x_{s_i^c}) = 0 \qquad \text{for all } k = 1, \dots, M \tag{3}$$

with coefficients $\alpha_i, \beta_k \in \mathbb{R}$. For a center node $x_c \in \Omega \cup \partial\Omega$, we define vectors

$$\mathcal{L}\boldsymbol{\Phi}_c := \left[\mathcal{L}\Phi_{x_{s_1^c}}(x_c), \cdots, \mathcal{L}\Phi_{x_{s_n^c}}(x_c)\right]^T \in \mathbb{R}^n,$$

$$\mathcal{L}\boldsymbol{p}_c := \left[\mathcal{L}p_1(x_c), \cdots, \mathcal{L}p_M(x_c)\right]^T \in \mathbb{R}^M$$

and the $(n+M) \times (n+M)$ matrix

$$\begin{bmatrix} A_c & P_c \\ P_c^T & \mathbf{0} \end{bmatrix} := \begin{bmatrix} \Phi_{x_{s_1^c}}(x_{s_1^c}) & \cdots & \Phi_{x_{s_n^c}}(x_{s_1^c}) & p_1(x_{s_1^c}) & \cdots & p_M(x_{s_1^c}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \Phi_{x_{s_1^c}}(x_{s_n^c}) & \cdots & \Phi_{x_{s_n^c}}(x_{s_n^c}) & p_1(x_{s_n^c}) & \cdots & p_M(x_{s_n^c}) \\ p_1(x_{s_1^c}) & \cdots & p_1(x_{s_n^c}) & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_M(x_{s_1^c}) & \cdots & p_M(x_{s_n^c}) & 0 & \cdots & 0 \end{bmatrix}.$$

Then Eq. (1) with weights

$$w^c := \left[w_1^c, \cdots, w_n^c\right]^T$$

is exact for functions of the form Eq. (2) that satisfy Eq. (3) if and only if there holds (see e.g. [24,25])

$$\begin{bmatrix} A_c & P_c \\ P_c^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} w^c \\ \widetilde{w}^c \end{bmatrix} = \begin{bmatrix} \mathcal{L}\boldsymbol{\Phi}_c \\ \mathcal{L}\boldsymbol{p}_c \end{bmatrix}. \tag{4}$$

To ensure that system Eq. (4) is uniquely solvable we require that $\ell + 1 \geq k$ (where ℓ denotes the degree of polynomial augmentation and k is the parameter in the radial basis function PHS(k)) and the stencil $\mathbb{S}_c$ to be $\Pi_\ell(\mathbb{R}^d)$-unisolvent [18,24].

Definition 4. A (finite) set $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^N$ of distinct nodes is called $\Pi_\ell(\mathbb{R}^d)$-unisolvent if there is no nonzero polynomial that vanishes on all nodes in X, i.e., for all $q \in \Pi_\ell(\mathbb{R}^d)$ there holds

$$q(x_j) = 0 \quad \forall j \in \{1, \dots, N\} \implies q = 0.$$

2.2. RBF-FD for partial differential equations

The approximation to a linear differential operator $\mathcal{L}$ can now be used for the discretization of a partial differential equation of the form

$$\mathcal{L}u(x) = f(x), \quad x \in \Omega, \tag{5}$$

$$u(x) = g(x), \quad x \in \partial\Omega, \tag{6}$$

where $f : \Omega \to \mathbb{R}$ is a sufficiently smooth function and $g : \partial\Omega \to \mathbb{R}$ specifies some Dirichlet boundary conditions. Replacing $\mathcal{L}$ in Eq. (5) by its approximations Eq. (1) at centers $x_j \in X_C = X_I$ leads to

$$\sum_{i=1}^{n} w_i^j u(x_{s_i^j}) \approx f(x_j), \quad j = 1, \dots, N_I. \tag{7}$$

Splitting this sum into two parts depending on whether $x_{s_i^j}$ is an interior or a (Dirichlet) boundary node, we obtain

$$\sum_{\substack{i \in \{1,\dots,n\} \\ \text{s.t. } x_{s_i^j} \in \Omega}} w_i^j u(x_{s_i^j}) \approx \underbrace{f(x_j) - \sum_{\substack{i \in \{1,\dots,n\} \\ \text{s.t. } x_{s_i^j} \in \partial\Omega}} w_i^j g(x_{s_i^j})}_{=: \widetilde{f}_j}$$

which leads to the definition of the right-hand side vector

$$\widetilde{f} := (\widetilde{f}_1, \dots, \widetilde{f}_{N_I})^T \in \mathbb{R}^{N_I}$$

and the global system matrix $B \in \mathbb{R}^{N_I \times N_I}$ which contains the weights of each stencil row-wise as follows,

$$B_{j,r} := \begin{cases} w_i^j & : \exists r \in \{1, \dots, n\} \text{ such that } r = s_i^j \in \mathcal{I}_j, \\ & \text{i.e., the interior node } x_r = x_{s_i^j} \text{ is in the stencil } \mathbb{S}_j, \\ 0 & : \text{else}, \end{cases} \tag{8}$$

for all $j, \ell \in \{1, \dots, N_I\}$. Proving that the matrix B is nonsingular has so far only been achieved in very restricted settings, see e.g. [26, Appendix A]. In practice, and in particular in our numerical results in Section 4, B turns out to be always nonsingular.

The solution of the system of linear equations

$$Bu = \widetilde{f} \tag{9}$$

yields the solution $u \in \mathbb{R}^{N_I}$ which is the RBF-FD approximation $u_j \approx u(x_j)$ to the solution of the partial differential Eq. (5), (6) at the interior nodes x_j for $j \in \{1, \dots, N_I\}$.

3. Discretization of the Oseen equations

We next consider the Oseen equations with Dirichlet boundary conditions, given by the following system of partial differential equations

$$\begin{aligned} -\nu\Delta\boldsymbol{u} + (\boldsymbol{b} \cdot \nabla)\boldsymbol{u} + \nabla p &= \boldsymbol{f} && \text{in } \Omega, \\ \operatorname{div}(\boldsymbol{u}) &= 0 && \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{g} && \text{on } \partial\Omega, \end{aligned} \tag{10}$$

where $\nu > 0$ denotes the viscosity parameter, $\Omega \subset \mathbb{R}^d$ is the domain, $\boldsymbol{f} : \Omega \to \mathbb{R}^d$ an external force (e.g. gravity), $\boldsymbol{g} : \partial\Omega \to \mathbb{R}^d$ prescribes the Dirichlet boundary conditions, $\boldsymbol{b} : \Omega \to \mathbb{R}^d$ is a given (divergence free) velocity field, whereas $\boldsymbol{u} : \Omega \to \mathbb{R}^d$ and $p : \Omega \to \mathbb{R}$ are the unknown velocity vector field (in d spatial dimensions) and scalar pressure, resp. The functions $\boldsymbol{u}, \boldsymbol{b}$ and p are assumed to be sufficiently smooth and $\boldsymbol{f}$ and $\boldsymbol{g}$ are assumed to be bounded.

3.1. Domain discretization

We allow for the discretization of the velocity $\boldsymbol{u}$ and the pressure p on different node sets $X_u \subset \Omega \cup \partial\Omega$ and $X_p \subset \Omega$, respectively. Nodes are pairwise distinct within each respective set, but the same node may appear in both sets. We will later consider the case where the pressure nodes coincide with the interior velocity nodes as one of our example settings. Different node sets were also chosen in [15] for the unsteady Navier-Stokes equations and in [9] for the Stokes equations. Both, however, required a mesh or a grid for the construction of the respective node sets which we will not need.

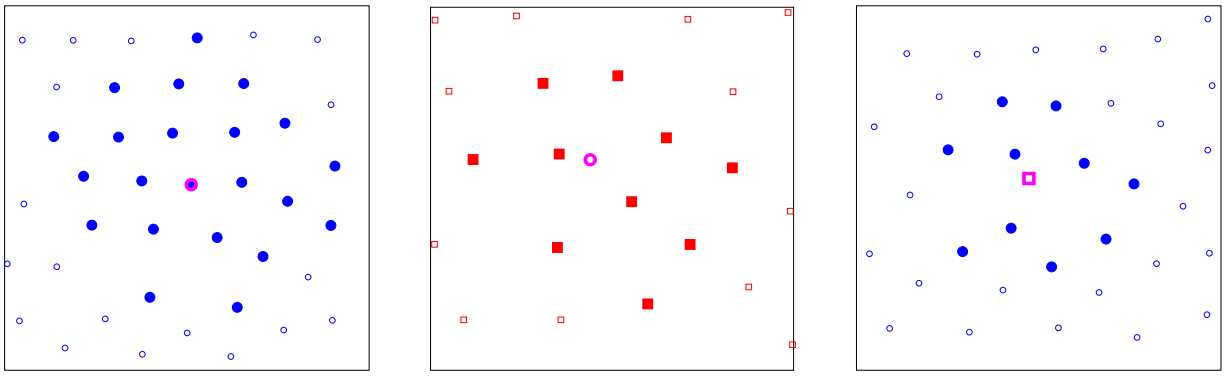


Fig. 1. Stencils illustrated in two spatial dimensions for the discretization of the convection-diffusion 1(a), the gradient 1(b) and the divergence 1(c) operators. Pressure nodes are red squares, velocity nodes are blue circles, nodes in a stencil are filled, stencil centers are magenta. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

We divide X_u into an interior node set $X_{u,I}$ and a boundary node set $X_{u,B}$ (see Definition 1), i.e.,

$$X_{u,I} = \{x^u_1, \dots, x^u_{N_{u,I}}\}, \quad X_{u,B} = \{x^u_{N_{u,I}+1}, \dots, x^u_{N_u}\}, \quad X_u = X_{u,I} \cup X_{u,B},$$
$$X_p = \{x^p_1, \dots, x^p_{N_p}\}$$

with $N_u := |X_u|$, $N_{u,I} := |X_{u,I}|$, $N_{u,B} := |X_{u,B}|$ and $N_p := |X_p|$.

For each of the d scalar component functions of $\boldsymbol{u}$ in Eq. (10), we discretize the diffusion and the convection operators separately as described in Section 2. In particular, we set $\mathcal{L} = -\nu\Delta$ and $\mathcal{L} = \boldsymbol{b} \cdot \nabla$, resp., and for both use the same node sets $X_I = X_{u,I}$, $X_B = X_{u,B}$ and $X_C = X_{u,I}$. The discretization of the two operators will (usually) differ in the degree of polynomial augmentation. We thereby obtain two matrices of size $N_{u,I} \times N_{u,I}$ whose sum yields the discrete convection-diffusion operator L in Eq. (11) (and some contribution to the right-hand-side vector through elimination of Dirichlet nodes).

A discretization of the pressure gradient ($\mathcal{L}p = \nabla p$) should also result in $N_{u,I}$ equations. However, in general we have $N_p \neq N_{u,I}$. Therefore, for each component of the gradient ∂_{x_i}, $i \in \{1, \dots, d\}$, we choose stencil centers in $X_{u,I}$ but stencil nodes in X_p, i.e., $X_C = X_{u,I}$, $X_I = X_p$, $X_B = \emptyset$, see also Fig. 1(b) for a visualization. The discretization yields the blocks G_i in Eq. (11) which combined represent the discrete gradient operator. In the special case that the pressure and (interior) velocity node sets are the same, $X_p = X_{u,I}$, we have $N_{u,I} = N_p$ and the blocks G_i will be square (since we do not have any boundary conditions for p).

For the discretization of the divergence $\mathcal{L} = \operatorname{div}$, we switch the roles of X_p and X_u, now choosing stencil centers in X_p but forming the corresponding stencils with nodes in X_u, i.e., $X_C = X_p$, $X_I = X_{u,I}$, $X_B = X_{u,B}$ (see Fig. 1(c)). The respective (first derivative) discretizations result in matrix blocks D_i of size $N_p \times N_{u,I}$ (and some contribution to the right-hand-side vector through elimination of Dirichlet nodes) which combined represent the discrete divergence operator.

We denote the matrices for convection-diffusion, gradient and divergence operators by L, G_i and D_i, $i \in \{1, \dots, d\}$, respectively, and obtain a (block) system of equations (shown for $d = 3$)

$$\begin{bmatrix} L & 0 & 0 & G_1 \\ 0 & L & 0 & G_2 \\ 0 & 0 & L & G_3 \\ D_1 & D_2 & D_3 & 0 \end{bmatrix} \begin{bmatrix} u_{h,1} \\ u_{h,2} \\ u_{h,3} \\ p_h \end{bmatrix} = \begin{bmatrix} \widetilde{f}_1 \\ \widetilde{f}_2 \\ \widetilde{f}_3 \\ \widetilde{f}_4 \end{bmatrix} \tag{11}$$

with matrix size $(d \cdot N_{u,I} + N_p) \times (d \cdot N_{u,I} + N_p)$. The vectors $\widetilde{f}_i$ on the right-hand-side are obtained as described in Section 2.2.

The solution of Eq. (11) approximates the analytical solution of the Oseen Eq. (10), i.e.,

$$\begin{bmatrix} (u_{h,1})_i \\ (u_{h,2})_i \\ (u_{h,3})_i \end{bmatrix} \approx \boldsymbol{u}(x^u_i) \quad \text{for all } 1 \le i \le N_{u,I} \text{ and}$$
$$(p_h)_j \approx p(x^p_j) \quad \text{for all } 1 \le j \le N_p,$$

where the subscript h is used to distinguish the discrete solution of Eq. (11) from the continuous solution of the Oseen equations Eq. (10).

Remark 1. Since the pressure p appears in the Oseen Eq. (10) only as a gradient and without any boundary conditions, it is only determined up to a constant. This property is carried over to the discrete case: The system Eq. (11) is singular since the vector of all ones is in the kernel of the matrices G_i. This follows since constants are always included in the polynomial augmentation of the PHS RBF-FD method (constants and linear functions are required to ensure convergence of the PHS RBF-FD method for a first order linear

differential operator [27]). In addition, the system Eq. (11) is typically inconsistent, i.e., the right-hand-side vector $\widetilde{f}$ is not in the range of the matrix. This is further discussed in Remark 2. In Section 3.2 we discuss different ways to introduce pressure constraints which augment Eq. (11) into a uniquely solvable system Eq. (14).

We characterize different node sets by (half of) the minimal pairwise distance between two points, the so-called separation distance.

Definition 5. Given a node set X in a domain Ω, the separation distance h_X is defined by

$$h_X := \frac{1}{2} \min_{x_i, x_j \in X, i \neq j} \|x_j - x_i\|_2.$$

In analogy to finite difference methods, we will set the step width equal to two times the separation distance, i.e., $h_u := 2h_{X_u}$ and $h_p := 2h_{X_p}$. Typically, the step widths for the velocity and pressure node sets will be different, i.e., $h_u \neq h_p$.

We construct scattered node sets using the node placing algorithm proposed in [16]. It requires as input the spatial dimension, a (desired) step width and a node set on the boundary of the domain, including outward pointing normal vectors for the boundary nodes. For the three-dimensional, polyhedral domains that we consider in this paper we proceed as follows: We begin with an equidistant node spacing along the edges (of the faces) of the polyhedron and then use the node placing algorithm [16] to first fill these faces and then the interior of the entire three-dimensional domain with nodes. This procedure is implemented in the C++ library `Medusa` [28]. The library also allows for extension to higher dimensions and parameterized boundaries.

In particular, we generate the two node sets X_u and X_p for a given polyhedron as follows:

- generate X_u with step width h_u by the procedure described above,
- generate a set of auxiliary boundary nodes X_A (including normal vectors) the same way that $X_{u,B}$ was generated but with a possibly different step width (we used the step width $h_u/2$),
- for each auxiliary boundary node $x \in X_A$, generate a so-called ghost node $x^* = x + h_g \cdot n$ where n is the outward pointing normal at x and $0 \leq h_g$ is the ghost node distance (x^* inherits the normal vector of x),
- remove the auxiliary boundary nodes in X_A,
- use these ghost nodes as a temporary boundary to generate the nodes in X_p with step width $h_p > h_g$ using the node placing algorithm,
- remove the ghost nodes.

If the ghost node distance h_g is large, nodes will be generated closer to the domain boundary, i.e., closer to nodes in $X_{u,B}$. And conversely, if h_g is small, nodes will be generated farther from the boundary.

A special case arises by choosing $X_p = X_{u,I}$ or even subsets $X_p \subsetneq X_{u,I}$. We will not pursue the latter option but mention that [29] provides an algorithm to select high quality subsets (in a certain sense) and also compares to other options to select such subsets. Using the same node sets for the velocity and pressure may be advantageous from a practical point of view, e.g., if certain node sets are already available.

The sets X_p and X_u are related by their step width ratio which we describe through the parameter $s = {}^{h_p}/_{h_u}$. In fact, our node generation allows to give the desired step width ratio as input such that we can generate (families of) point sets X_u, X_p for a given parameter $s > 0$.

In Fig. 2, discretizations of the unit square for different choices of the parameters h_u, h_p and h_g and the special case $X_p = X_{u,I}$ are shown in two dimensions (for illustrative purposes, numerical experiments will be performed in three dimensions). In particular, $h_u = h_p$ and $h_g = 0$ is not necessarily the same as choosing $X_p = X_{u,I}$.

3.2. Uniqueness of the pressure

As mentioned in Remark 1, the pressure p is only determined uniquely up to a constant. In order to obtain a unique pressure p (and a uniquely solvable system Eq. (11)), one may fix the value of p at one node, e.g. $p(x_1^p) = 0$. This can be realized by removing the corresponding row and column from the divergence and gradient matrices in Eq. (11), respectively, and adjusting the right-hand side accordingly.

Another possibility, also used in [9], is to enforce (a discrete approximation of) the constraint

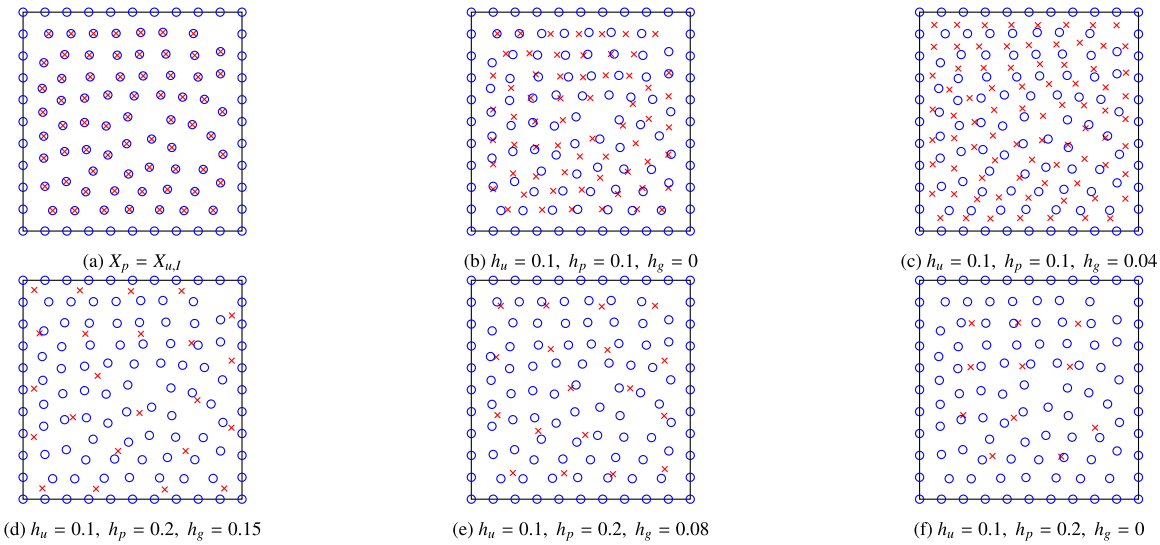
$$\int_\Omega p(x)\,\mathrm{d}x = 0.$$

We will introduce two approximations (i.e., quadrature formulas) of the type

$$y^T p_h \approx \int_\Omega p(x)\,\mathrm{d}x \tag{12}$$

where $y \in \mathbb{R}^{N_p}$ denotes the vector of quadrature weights. In [9], $y_i = 1$ has been used for all $i = 1, \ldots, N_p$ which worked well in numerical tests for the two-dimensional, structured node sets.

For unstructured node sets, we propose to compute the weights as introduced in [30] which is described next.



(a) $X_p = X_{u,I}$ (b) $h_u = 0.1,\ h_p = 0.1,\ h_g = 0$ (c) $h_u = 0.1,\ h_p = 0.1,\ h_g = 0.04$

(d) $h_u = 0.1,\ h_p = 0.2,\ h_g = 0.15$ (e) $h_u = 0.1,\ h_p = 0.2,\ h_g = 0.08$ (f) $h_u = 0.1,\ h_p = 0.2,\ h_g = 0$

Fig. 2. Domain discretizations of the unit square $\Omega = [0,1]^2$ for different step widths h_u, h_p and h_g.

For polynomial degree $k \in \mathbb{N}_0$ in the space of polynomials Π_k (see Section 2), let $L = \dim(\Pi_k)$ and $\{p_1, \dots, p_L\}$ be a basis of Π_k. With a node set $X_p = \{x_1, \dots, x_{N_p}\}$ where $L < N_p$, we define

$$P := \begin{bmatrix} p_1(x_1) & \cdots & p_1(x_{N_p}) \\ \vdots & \ddots & \vdots \\ p_L(x_1) & \cdots & p_L(x_{N_p}) \end{bmatrix} \in \mathbb{R}^{L \times N_p}$$

and

$$b \in \mathbb{R}^L \quad \text{with} \quad b_i = \int_\Omega p_i(x)\,\mathrm{d}x \quad \text{for all } 1 \le i \le L.$$

For polyhedral domains Ω (which will be the case in our numerical experiments), b can be computed exactly (up to machine precision) using the procedure described in [31]. We thus obtain the (underdetermined) system of linear equations

$$Py = b. \tag{13}$$

This system is consistent if X_p is $\Pi_k(\mathbb{R}^d)$-unisolvent and $L < N_p$ [30]. It is further shown that the solution of minimal ℓ_2-norm is unique. In [30], this unique minimal ℓ_2-norm solution is computed by `MATLAB's lsqminnorm` function which uses the complete orthogonal decomposition. We will also use the complete orthogonal decomposition [32, Section 5.4.7] in our implementation. The complete orthogonal decomposition has a complexity of $\mathcal{O}(L^2 N_p)$. Typically, there holds $L \ll N_p$. In our numerical experiments we restrict ourselves to polynomial degree $k \le 11$ (leading to $L \le 364$) such that the algorithm is linear in N_p.

For stability reasons it is desirable to obtain only nonnegative weights, i.e., $y \ge 0$ (componentwise). We will adopt the approach suggested in [30]: Starting with $k = 0$ results in the quadrature weights $y = \frac{|\Omega|}{N_p}(1, \dots, 1)^T > 0$. Then we repeatedly increase k by one and compute y until $y \not\ge 0$. Finally, we select the previously computed $y \ge 0$.

For domains more general than polyhedra, the approach described in [33] could be used to obtain the right-hand side b in Eq. (13). However, this is not further pursued here.

The discrete integral constraint Eq. (12) extends the linear system Eq. (11) in the following way:

$$\begin{bmatrix} L & 0 & 0 & G_1 & 0 \\ 0 & L & 0 & G_2 & 0 \\ 0 & 0 & L & G_3 & 0 \\ D_1 & D_2 & D_3 & 0 & y \\ 0 & 0 & 0 & y^T & 0 \end{bmatrix} \begin{bmatrix} u_{h,1} \\ u_{h,2} \\ u_{h,3} \\ p_h \\ \lambda \end{bmatrix} = \begin{bmatrix} \widetilde{f}_1 \\ \widetilde{f}_2 \\ \widetilde{f}_3 \\ \widetilde{f}_4 \\ 0 \end{bmatrix}. \tag{14}$$

The extra row incorporates the integral constraint while the extra column has been added to obtain a square matrix. In our numerical experiments the resulting matrix is always regular, however, we are not aware of any proof that guarantees this for RBF-FD discretizations on scattered node sets. The unknown $\lambda \in \mathbb{R}$ may be discarded after solving the system.

Remark 2. If the component λ in the solution of Eq. (14) equals zero, then the original (singular) system Eq. (11) has been consistent. In our numerical experiments, we have $\lambda \neq 0$ (λ is typically multiple magnitudes larger than the machine precision or solver tolerance) when solving Eq. (14). This behavior has also been observed in [9]. On the positive side, $|\lambda|$ is decreasing as the number of unknowns is increased.

3.3. The influence of the viscosity ν on the discretization error

The following will be useful in explaining some of the phenomena observed in the numerical results in Section 4.

Let $p \in \mathscr{C}^1(\Omega, \mathbb{R})$, $\boldsymbol{u} \in \mathscr{C}^2(\Omega, \mathbb{R}^d) \cap \mathscr{C}(\overline{\Omega}, \mathbb{R}^d)$ with $\operatorname{div}(\boldsymbol{u}) = 0$ and $\boldsymbol{b} \in \mathscr{C}(\Omega, \mathbb{R}^d)$. If we set

$$\boldsymbol{f} := \boldsymbol{f}_{\nu,\boldsymbol{b},\boldsymbol{u},p} := -\nu\Delta\boldsymbol{u} + (\boldsymbol{b}\cdot\nabla)u + \nabla p \quad \text{in } \Omega, \quad \boldsymbol{g} := \boldsymbol{g}_{\boldsymbol{u}} := \boldsymbol{u} \quad \text{on } \partial\Omega,$$

then $(\boldsymbol{u}, p)$ solves the Oseen problem Eq. (10). In fact, $(\boldsymbol{u}, p)$ is the solution of the Oseen problem with viscosity $\widetilde{\nu}$ and force term $\boldsymbol{f} := \boldsymbol{f}_{\widetilde{\nu},\boldsymbol{b},\boldsymbol{u},p} := -\widetilde{\nu}\Delta\boldsymbol{u} + (\boldsymbol{b}\cdot\nabla)u + \nabla p$ for any $\widetilde{\nu} > 0$.

The following Theorem 1 will illustrate how this invariance with respect to ν will change in the discretized setting for the case $\boldsymbol{b} = \boldsymbol{0}$ (Stokes problem). In the following Proposition and Theorem, one may interpret the matrices L, D, G as discretized versions of the Laplacian, divergence and gradient, respectively. Given an exact (continuous) solution $(\boldsymbol{u}, p)$, the vectors ℓ, d, g may be interpreted as evaluations of $-\Delta\boldsymbol{u}, \operatorname{div}\boldsymbol{u}, \nabla p$ at the underlying (velocity/pressure) nodes. We will go into further interpretations after the Theorem 1. First we convert our system of Eq. (14) into a form that will be more convenient for the analysis in the following Proposition 1.

Proposition 1. *Let $\mathbf{1} \in \mathbb{R}^M$ be the vector of all ones and $\nu \in (0,\infty)$. Let $L \in \mathbb{R}^{N\times N}$, $G \in \mathbb{R}^{N\times M}$, $D \in \mathbb{R}^{M\times N}$, $\ell, g, u \in \mathbb{R}^N$, $d, p, y \in \mathbb{R}^M$ for $M, N \in \mathbb{N}$ with $y^T\mathbf{1} =: c \in (0,\infty)$ and $G\mathbf{1} = \mathbf{0}$. Then there holds*

$$\begin{bmatrix} u^* \\ p^* \\ \lambda^* \end{bmatrix} \text{solves} \begin{bmatrix} \nu L & G & 0 \\ D & 0 & y \\ 0 & \nu^{-1}y^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ g \\ 0 \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} u^* \\ p^* + \frac{\nu\lambda^*}{c}\mathbf{1} \end{bmatrix} \text{solves} \begin{bmatrix} \nu L & G \\ D & \nu^{-1}yy^T \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

and

$$\begin{bmatrix} u^+ \\ p^+ \end{bmatrix} \text{solves} \begin{bmatrix} \nu L & G \\ D & \nu^{-1}yy^T \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} u^+ \\ p^+ - \frac{y^Tp^+}{c}\mathbf{1} \\ y^Tp^+ \end{bmatrix} \text{solves} \begin{bmatrix} \nu L & G & 0 \\ D & 0 & y \\ 0 & \nu^{-1}y^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ g \\ 0 \end{bmatrix}.$$

Proof. Plug the solution into the system. □

Theorem 1. *Given the linear system*

$$\begin{bmatrix} \nu L & G \\ D & \nu^{-1}yy^T \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} \nu\ell + g \\ d \end{bmatrix} \tag{15}$$

where $\nu \in (0,\infty)$, $L \in \mathbb{R}^{N\times N}$, $G \in \mathbb{R}^{N\times M}$, $D \in \mathbb{R}^{M\times N}$, $\ell, g, u \in \mathbb{R}^N$, $d, p, y \in \mathbb{R}^M$ for $M, N \in \mathbb{N}$. Let $\mathbf{1} \in \mathbb{R}^M$ be the vector of all ones and assume that $y^T\mathbf{1} =: c \in (0,\infty)$ and $G\mathbf{1} = \mathbf{0}$. Assume that the system matrix as well as the first diagonal block L are nonsingular. Define

$$\bar{u} := L^{-1}\ell, \qquad \bar{p} := \arg\min_{q\in\mathbb{R}^M,\ y^Tq=0} \|g - Gq\| \tag{16}$$

where $\|\cdot\| : \mathbb{R}^N \to \mathbb{R}$ is an arbitrary norm. Further, we define the following two vectors which may be interpreted as errors (or deviations) in the divergence and gradient compared to the continuous setting,

$$e_d := d - DL^{-1}\ell = d - D\bar{u}, \qquad e_g := g - G\bar{p}.$$

Then the velocity and pressure components of the discrete solution can be expressed in terms of the vectors e_d, e_g and the viscosity ν as follows:

$$p = \bar{p} + \nu S^{-1}e_d + S^{-1}DL^{-1}e_g,$$
$$u = \bar{u} + L^{-1}GS^{-1}e_d + \nu^{-1}L^{-1}(I - GS^{-1}DL^{-1})e_g$$

where $S := DL^{-1}G - yy^T$ denotes the (scaled) Schur complement.

Proof. We transform the system into block upper triangular form by subtracting $\frac{1}{\nu}DL^{-1}$ times the first block equation from the second and obtain

$$\begin{bmatrix} \nu L & G \\ 0 & -\nu^{-1}S \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} \nu\ell + g \\ \widetilde{d} \end{bmatrix}$$

where

$$S = DL^{-1}G - yy^T$$

and

$$\begin{aligned} \widetilde{d} &= d - \nu^{-1}DL^{-1}(\nu l + g) \\ &= e_d - \nu^{-1}DL^{-1}G\bar{p} - \nu^{-1}DL^{-1}e_g \\ &= e_d - \nu^{-1}\widetilde{S}\bar{p} - \nu^{-1}DL^{-1}e_g \end{aligned}$$

where $\widetilde{S} = DL^{-1}G$. The Schur complement S is nonsingular in view of our assumption that the system matrix and L are nonsingular. Therefore, we obtain

$$p = -\nu S^{-1}\widetilde{d} = -\nu S^{-1}(\widetilde{d} - yy^T\bar{p}) = -\nu S^{-1}e_d + \bar{p} + S^{-1}DL^{-1}e_g.$$

We can then compute

$$\begin{aligned} u &= \nu^{-1}L^{-1}(\nu\ell + g - Gp) \\ &= L^{-1}\ell + \nu^{-1}L^{-1}\big(g - G(-\nu S^{-1}e_d + \bar{p} + S^{-1}DL^{-1}e_g)\big) \\ &= \bar{u} + \nu^{-1}L^{-1}e_g + L^{-1}GS^{-1}e_d - \nu^{-1}L^{-1}GS^{-1}DL^{-1}e_g \\ &= \bar{u} + L^{-1}GS^{-1}e_d + \nu^{-1}L^{-1}\big(I - GS^{-1}DL^{-1}\big)e_g. \end{aligned}$$

□

For the continuous case, we had argued that varying the factor ν for the Laplace term does not change the continuous solution $(\boldsymbol{u}, p)$ if the external force is adjusted accordingly to $\boldsymbol{f} = \nu\Delta\boldsymbol{u} + \nabla p$. This property carries over to the discrete setting only in the very special case when both $g \in \text{span}(G)$ and $d = D\bar{u}$ hold. In this case, the vectors e_g and e_d are zero vectors and there holds $u = \bar{u}$ and $p = \bar{p}$ Eqs. (15) and (16).

However, typically a discretized system satisfies neither one of the above two conditions, and hence the discrete solution vector $(u, p)^T$ changes when ν changes in Eq. (15). Then the (ν-invariant) vectors $\bar{u}$ and $\bar{p}$ in Eq. (16) can be viewed as approximations to the (ν-invariant) continuous solution that share this invariance property. We will later use the representations obtained for u and p from Theorem 1 to explain the numerically observed dependence of the discrete pressure and velocity errors on ν. A related argument is also well-known for non-divergence-free mixed Finite Element methods [34,35].

The proof of Theorem 1 has not required any information on the discretization that was used to obtain the linear system Eq. (15). It can therefore be applied to any discretization which leads to this type of system.

We now show that Theorem 1 is applicable to our RBF-FD discretized system Eq. (14). The assumption $G\mathbf{1} = \mathbf{0}$ is fulfilled by construction of the RBF-FD method (see Remark 1). Note that setting the value of the pressure to $\gamma \in \mathbb{R}$ at the i-th pressure node is equivalent to setting $y = e_i$ where e_i is the i-th unit vector and requiring $y^T p = \gamma$. However, we can restrict ourselves to the case $y^T p = 0$, because the following holds under the assumptions of Proposition 1:

$$\begin{bmatrix} u_0 \\ p_0 \\ \lambda_0 \end{bmatrix} \text{solves} \begin{bmatrix} \nu L & G & 0 \\ D & 0 & y \\ 0 & \nu^{-1}y^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ g \\ 0 \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} u_0 \\ p_0 + \frac{\gamma}{c}\mathbf{1} \\ \lambda_0 \end{bmatrix} \text{solves} \begin{bmatrix} \nu L & G & 0 \\ D & 0 & y \\ 0 & \nu^{-1}y^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ g \\ \gamma \end{bmatrix}.$$

For the three pressure constraints which we discussed to obtain the vector y, there holds $y^T\mathbf{1} > 0$. Therefore, in all three cases we can apply Proposition 1 to transform the system Eq. (14) into the form Eq. (15). If we further assume the diagonal block L and the system matrix in Eq. (15) to be regular, we can apply Theorem 1 to our RBF-FD discretization. Our numerical results in Section 4.6 indicate that this regularity assumption is justified in our test cases.

A special case is discussed in the following Remark.

Remark 3. If the pressure is a polynomial of degree at most $\ell \in \mathbb{N}$, i.e., $p \in \Pi_\ell(\mathbb{R}^d)$, and the degree of polynomial augmentation used in the PHS RBF-FD method to assemble the matrix G is also (at least) ℓ, then $g \in \text{span}(G)$. This results from the PHS RBF-FD weights being constructed such that Eq. (1) holds exactly for all $q \in \Pi_\ell(\mathbb{R}^d)$ evaluated at points in $X_{u,I}$. Therefore, in Theorem 1, the error e_g will vanish. One can argue analogously that e_d will vanish if each component of the velocity $\boldsymbol{u}$ is a polynomial of at most the degree of polynomial augmentation.

4. Numerical results

In this Section, we show numerical tests for the Oseen Eq. (10) in $d = 3$ spatial dimensions where the domain Ω is a unit cube $(0, 1)^3$ or a bunny-shaped polyhedron (scaled to fit within the unit cube) shown in Fig. 3.

We will set the boundary conditions and force function in the Oseen equations such that the exact solution is known to us and can be used to measure the discretization error and in particular assess convergence for decreasing step widths h_u, h_p (and hence increasing system sizes). In Section 4.1, we list those exact solutions and give some general information regarding the parameters of the RBF-FD method.

In the subsequent subsections, we show our numerical results, in particular

- for discretizations with different ratios $s = \frac{h_p}{h_u}$ of the step widths (Section 4.2),
- for different pressure constraints (Section 4.3),
- for different convection directions (Section 4.4),
- for different combinations of degrees of polynomial augmentation (Section 4.5) and
- for different viscosities (Section 4.6).

Furthermore, we will discuss scalibility aspects in Section 4.7.

4.1. General information

We will perform numerical tests for manufactured problems whose analytical solution is known to us. In particular, we select solutions that will illustrate different convergence behavior of the RBF-FD discretization depending on whether the solution is of

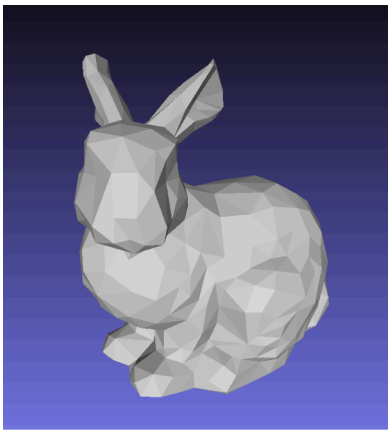


Fig. 3. Bunny domain, originally obtained at https://github.com/OpenGP/OpenGP/blob/master/data/bunny.off, scaled version available at https://github.com/mkochh/OseenRBFFD/Tests/OFF_Files/bunny.off.

polynomial or non-polynomial type. The velocity will be one of the following two (divergence-free) functions,

$$\boldsymbol{u}^1(x) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \qquad \boldsymbol{u}^2(x) = \begin{pmatrix} \cos(x) \cdot \sin(y) \\ x \cdot \cos(y) \\ z \cdot (x + \sin(x)) \cdot \sin(y) \end{pmatrix}.$$

For the pressure, we use one of the following two functions

$$p^1(x) = (x - 0.5) \cdot (y - 0.5) \cdot (z - 0.5) + c_2(\Omega),$$
$$p^2(x) = \sin(\pi x) \cdot \cos(\pi y) + c_1(\Omega)$$

where the constants $c_j(\Omega)$ are set such that there holds $\int_\Omega p(x)\,dx = 0$ (up to machine precision), in particular,

$$c_1(\text{cube}) = 0, \qquad c_1(\text{bunny}) = -0.260568484878944,$$
$$c_2(\text{cube}) = 0, \qquad c_2(\text{bunny}) = -0.001363726470335.$$

Tests will be performed for the following three continuous solutions:

Solution 1: $(\boldsymbol{u}^1, p^2)$, (polynomial velocity, nonpolyn. pressure),
Solution 2: $(\boldsymbol{u}^2, p^2)$, (nonpolynomial velocity and pressure),
Solution 3: $(\boldsymbol{u}^2, p^1)$, (nonpolyn. velocity, polynomial pressure).

Further we will perform tests for the following three divergence-free convection directions,

$$\boldsymbol{b}_1(\mathbf{x}) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \qquad \boldsymbol{b}_2(\mathbf{x}) = c \begin{pmatrix} 2x(1-x)(2y-1)z \\ (2x-1)y(y-1) \\ (2x-1)(2y-1)z(z-1) \end{pmatrix}, \qquad \boldsymbol{b}_3(\mathbf{x}) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{17}$$

with $c = 6.40936351829$ such that $\int_{[0,1]^3} \|\boldsymbol{b}_2(\boldsymbol{x})\|_2 \,\mathrm{d}\boldsymbol{x} \approx 1 = \int_{[0,1]^3} \|\boldsymbol{b}_1(\boldsymbol{x})\|_2 \,\mathrm{d}\boldsymbol{x}$. The third option $\boldsymbol{b}_3$ leads to the Stokes equations which will be useful for comparison in some test cases.

For the choice of the exponent k of the PHS(k) (see Definition 3), the degree of polynomial augmentation ℓ and the stencil size n, we follow the recommendations in [24]. That is to say, we first choose the degree of polynomial augmentation $\ell \in \mathbb{N}$ as it determines the convergence order [27]. Then we set the stencil size to

$$n = 2M + \lfloor \ln(2M) \rfloor \text{ with } M = \binom{\ell + d}{d}$$

and the parameter k in PHS(k) to

$$k = \begin{cases} \max\left\{ D, \frac{\ell+1}{2} \right\}, & \text{if } \ell \text{ is odd}, \\ \max\left\{ D, \frac{\ell}{2} \right\}, & \text{if } \ell \text{ is even} \end{cases}$$

where D is the order of the differential operator, in our case $D = 2$ for the Laplacian and $D = 1$ for convection, gradient and divergence operators. Stencils are formed by choosing n nodes which are closest to the center node in the euclidean norm. We use the C++ library `nanoflann` [36] which uses a kd-tree to find the nearest neighbors.

We choose the order ℓ of polynomial augmentation separately for each (Laplacian, convection, gradient, divergence) operator and use the shorthand notation $lcgd$ for the respective orders, e.g., $lcgd = 3222$ stands for using $\ell = 3$ for the Laplacian, $\ell = 2$ for the convection, etc.

Table 1
Number of pressure nodes N_p and velocity nodes $N_{u,I}$ for different values of h_u and h_p (step widths rounded to fourth decimal place).

s	2.0			1.4			1.0		
h_u	0.05	0.0152	0.0089	0.05	0.0152	0.0089	0.05	0.0152	0.0089
h_p	0.1	0.0303	0.0179	0.07	0.0212	0.0125	0.05	0.0152	0.0089
$N_{u,I}$	5,976	230187	1,139,097	5,976	230,187	1,139,097	5,976	230,187	1,139,097
N_p	715	28483	49,183	2,173	84,023	415,879	6,161	232,677	1,149,010

In our results we show relative 2-norm errors: Given a numerical solution $x_{num} \in \mathbb{R}^N$ and an exact solution $x_{exact} \in \mathbb{R}^N$ (analytical solution evaluated at the nodes), we compute

$$e_2 := e_2(x_{num}, x_{exact}) := \frac{\|x_{num} - x_{exact}\|_2}{\|x_{exact}\|_2}. \tag{18}$$

We solve the discrete linear systems iteratively with a preconditioned BiCGstab method which stops when the 2-norm relative residual is reduced below 10^{-12} (or a maximum number of 1024 iterations is reached). While the analysis of iterative solvers for RBF-FD systems is a subject of its own interest, our focus of this paper lies in the analysis of the discretization error. We have chosen a stopping criterion with a rather small relative residual norm such that we expect the measured errors e_2 to be of the same order as errors obtained with exact solvers.

4.2. Discretizations with different step widths

In the following we discuss the impact of the quotient $s = h_p/h_u$ as well as the distance of the pressure nodes to the boundary on the discretization error. The distance of the pressure nodes to the boundary is determined by the ghost node distance h_g (see Section 3.1). We first show results for a fixed ratio $h_p/h_g = 2.5$ (i.e., $h_g = h_p/2.5$) and vary the values for the ratio $s = \frac{h_p}{h_u}$. Since our experiments are performed in $d = 3$ spatial dimensions, it holds that $s^3 N_p \approx N_{u,I}$. In Table 1, we show the number of pressure nodes N_p and velocity nodes $N_{u,I}$ for the smallest, a medium and the largest velocity node step widths ($h_u = 0.05$, $h_u = 0.0152$ and $h_u = 0.0089$, resp.) used in our numerical tests.

Fig. 4 shows the convergence for a range of different values of s. The exact solution in these experiments was either Solution 1 (Fig. 4(a) and (b)) or Solution 3 (Fig. 4(c) and (d)) with convection $\boldsymbol{b}_1$ on the cube with polynomial augmentation $lcgd = 4333$ and viscosity $\nu = 10^{-2}$. The results were (qualitatively) similar for other degrees of polynomial augmentation and viscosities ν. Results for the bunny-shaped domain were also qualitatively similar. Tests using Solution 2 showed behavior comparable to those with Solution 1.

For Solution 1, we see in Fig. 4(a) that reducing s from 2.4 down to 1.4 (and therefore increasing the number of pressure nodes N_p) gradually reduces the relative pressure error $e_{2,p}$. At around $s = 1$ ($N_{u,I} \approx N_p$), the improvements stagnate. For $s = 0.8$, our iterative solver did not reach the solver tolerance of 10^{-12}. These observations are similar to the findings in [9] where the authors proposed that the condition $N_p/N_u < 1$ should hold to guarantee stability of the numerical scheme.

The situation changes somewhat when we look at the results for Solution 3 (polynomial pressure) in Fig. 4(c). For $h_u < 0.02$, the relative errors are very similar when using $s = 1.4, 2.0, 2.4$. For larger values of h_u (i.e., coarser node sets), choosing $s = 1.4$ results in smaller relative errors compared to $s = 2.0, 2.4$. Lowering s beyond 1.4 leads to an increase in relative errors. The iterative solver again did not converge for the case $s = 0.8$. The case of polynomial pressure was not considered in [9].

We further observe that the relative velocity error $e_{2,u}$ behaves qualitatively very similar to the relative pressure error $e_{2,p}$, showing that the two errors are coupled to each other. For example, the spike for $s = 1$ at $h_u \approx 0.3$ for Solution 1 (Fig. 4(a) and (b)) occurs for both the pressure and the velocity errors. This may be explained partially by Theorem 1 where both errors e_d and e_g appear in the representation of both solution vectors u and p.

We now fix $s = h_p/h_u$ to be 1, i.e., $h_u = h_p$, and vary the way we place the pressure nodes X_p. The first option is to set $X_p = X_{u,I}$. The second option uses the ghost node distance $h_g = 0$ while the third uses $h_g = h_p/2.5$ (and hence allows pressure nodes closer to the boundary).

In Fig. 5 we see that $h_g = 0$ achieves slightly worse relative errors compared to the other two options for both exact solutions. The option $h_g = 0$ also shows a somewhat irregular convergence behavior in both test cases. For $h_g = h_p/2.5$ and $X_p = X_{u,I}$ the convergence is more comparable and convergence is more regular for Solutions 1 and 3, except for two outliers. The reason for these outliers is not clear, but we hypothesize that matrices become more ill-conditioned as s is lowered.

Obviously we cannot cover every possible node placement and merely want to show that even minor changes in the discretization can lead to a quite different convergence behavior. As a conclusion from the tests in this subsection, we will use $s = 1.4$ and $h_g = h_p/2.5$ for all remaining tests since this setting performed well across the range of our experiments.

For comparison purposes we also present results obtained using the second and third order Taylor-Hood mixed finite element discretization [37,38], i.e., we choose finite element spaces

$$V_h = H_0^1(\Omega)^d \cap \mathcal{P}_k(\mathcal{T})^d \text{ for the velocity,}$$
$$Q_h = H^1(\Omega) \cap \mathcal{P}_{k-1}(\mathcal{T}) \text{ for the pressure,}$$

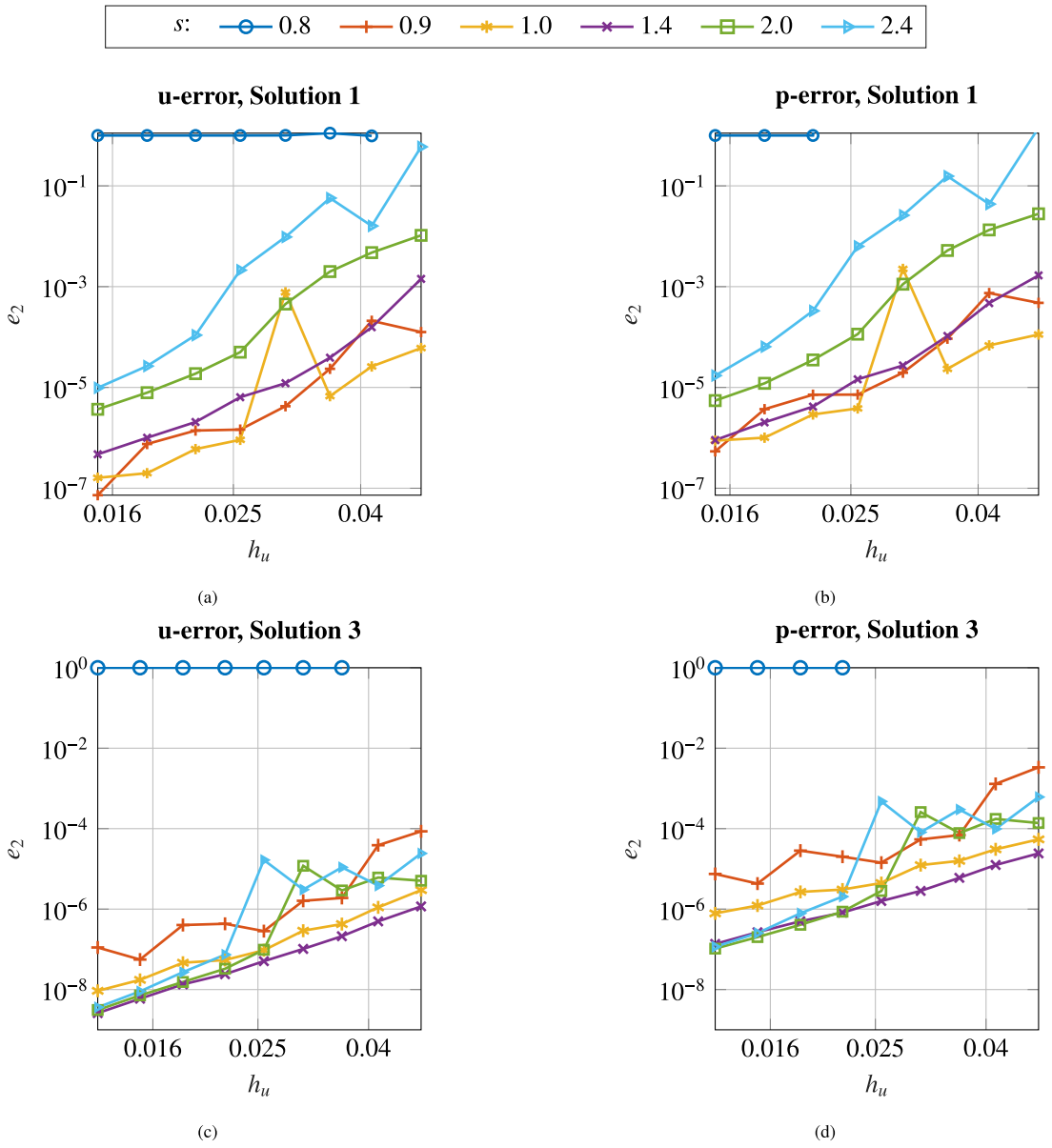


Fig. 4. Discretization errors Eq. (18) for different values of $s = h_p/h_u$ for the exact Solution 1 ((a), (b)) and exact Solution 3 ((c), (d)) with convection $\boldsymbol{b}_1$ on the cube with polynomial augmentation lcgd = 4333 and viscosity $\nu = 0.01$.

where $k \in \{2,3\}$, $\mathcal{T}$ is a triangulation of Ω consisting of simplices τ and

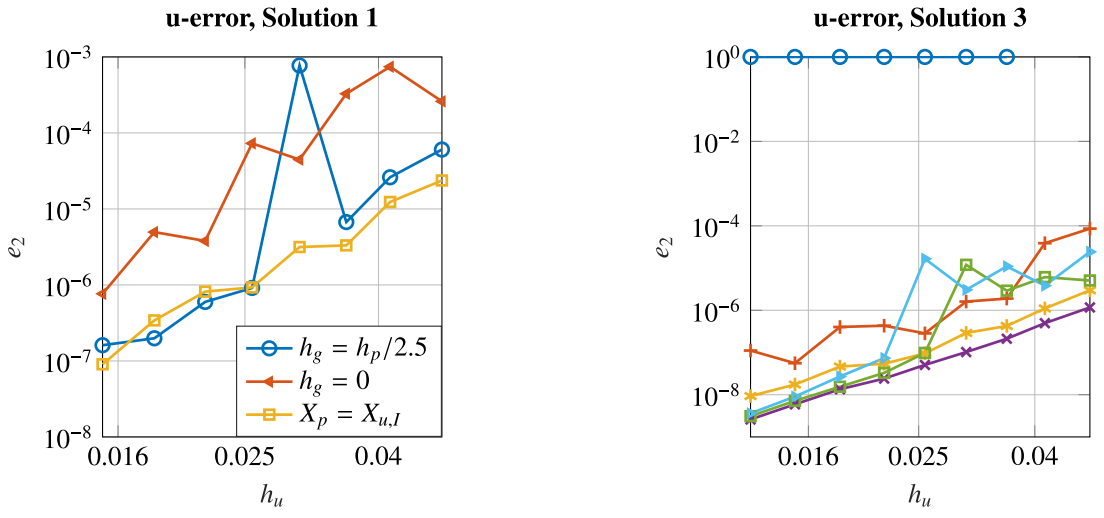
$$\mathcal{P}_k(\mathcal{T}) = \{v \in \mathscr{C}(\Omega) : v|_\tau \in \Pi_k, \text{ for all } \tau \in \mathcal{T}\}.$$

We denote the k-th order Taylor-Hood space by TH(k) and define the maximum diameter of the simplices of the triangulation as

$$h_{max} := \max_{\tau \in \mathcal{T}} \operatorname{diam}_2(\bar{\tau}).$$

Analogously to the definition of $N_{u,I}$ and N_p (see Section 3.1), we denote the number of degrees of freedom for the velocity per dimension by N_V and for the pressure by N_Q. The total number of (finite element) degrees of freedom is then given by $N_{DOF} := 3 \cdot N_V + N_Q$ (compared to $N_{DOF} := 3 \cdot N_{u,I} + N_p$ for RBF-FD).

We use the Python interface of the open-source C++ library `NGSolve` (see https://ngsolve.org/) [39] to generate a (non-uniform) mesh, assemble the FEM matrices and then solve the linear system of equations. The results for Solution 1 on the cube with convection b_1 and viscosity $\nu = 10^{-2}$ can be seen in Table 2.



(a) (b)

Fig. 5. Discretization errors Eq. (18) of the velocity for different placements of the pressure nodes X_p for the exact Solution 1 Fig. 5(a) and Solution 3 Fig. 5(b) with convection b_1 on the cube with lcgd = 4333 and $\nu = 0.01$, $s = 1$.

Table 2
Errors for the Taylor-Hood mixed finite element discretization compared to an RBF-FD discretization.

TH(2)						
$h_{\max}$	N_V	N_Q	nnz	nnz/N_{DOF}	$e_{2,u}$	$e_{2,p}$
0.158	5,062	1,177	607,451	37.1	5.7e−2	1.5e−2
0.082	39,933	7,124	5,124,853	40.4	9.3e−3	3.1e−3
0.044	333,215	49,838	44,230,804	42.1	1.3e−3	9.5e−4
TH(3)						
$h_{\max}$	N_V	N_Q	nnz	nnz/N_{DOF}	$e_{2,u}$	$e_{2,p}$
0.306	2,178	1,257	495,440	63.6	1.4e−1	1.5e−1
0.158	18,511	7,824	4,666,872	73.7	1.4e−2	3.3e−2
0.082	140,818	51,067	37,311,720	78.8	1.3e−3	7.0e−3
RBF-FD ($lcgd$ =3222, $s = 1.4$)						
h_u	$N_{u,I}$	N_p	nnz	nnz/N_{DOF}	$e_{2,u}$	$e_{2,p}$
0.050	5,976	2,173	1,234,497	61.1	1.0e−3	1.6e−3
0.030	27,926	10,191	5,912,130	62.8	1.3e−4	5.6e−4
0.015	230,187	84,023	49,642,210	64.0	2.4e−5	1.6e−4

A full comparison, which we do not provide here, would require an analysis of the computational time involved per degree of freedom. Our comparison here should merely indicate that RBF-FD discretization is competitive with some standard finite element discretization techniques.

4.3. *Different pressure constraints*

Here we numerically compare the three methods described in Section 3.2 to make the pressure unique which we denote as follows:

- "set": Set the value of the pressure at the node that was generated last (which is typically near the center of the domain) to the value of the exact solution at that node.
- "average": Use the simple (averaging) quadrature rule Eq. (12) with all quadrature weights y_i set equal to one.
- "quad poly": Use the more complicated quadrature rule with quadrature weights computed via Eq. (13).

Fig. 6 shows numerical results for the three different options on the bunny domain for Solution 2, convection $\boldsymbol{b}_1$, viscosity $\nu = 10^{-2}$ and polynomial augmentation degrees $lcgd = 4333$. In Fig. 6(b), we see that the advanced option "quad poly" leads to significantly smaller pressure errors compared to the other two options, especially for larger numbers of degrees of freedom (i.e., smaller h_u). Fig. 6(a) shows that all three variants produce similar relative errors for the velocity when $h_u \geq 0.016$. For $h_u < 0.016$, fixing the value

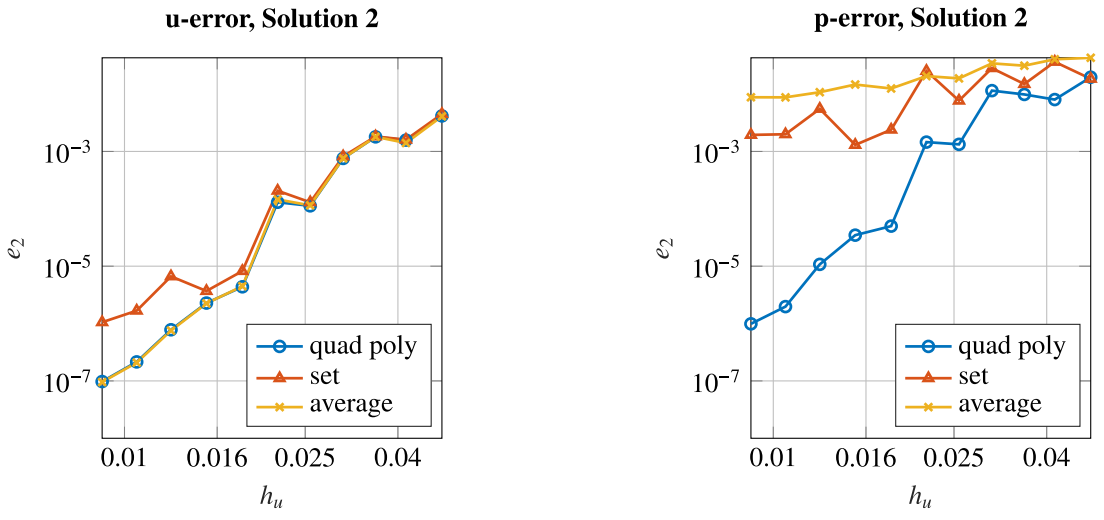


Fig. 6. Discretization errors Eq. (18) when using the different strategies (quad poly, set, average) to make the pressure unique.

of the pressure at one node leads to slightly larger errors compared two the other two methods. Therefore, different pressure constraints may lead to different solutions for the velocity (as well as the pressure). The velocity errors in Fig. 6(a) are indistinguishable for the options "quad poly" and "average". However, there are indeed small deviations between the velocity solutions obtained with "quad poly" or "average" constraints. The relative 2-norm difference between these two solutions is about one to three orders of magnitude smaller than the relative 2-norm error to the exact velocity solution e_2 shown in Fig. 6(a). In [9] it is mentioned that setting the value of the pressure at one node may produce local distortions in the solution. These distortions could be due to the inconsistency of the system (see Remark 2). As mentioned in Section 3.3, the "set" option can be viewed as a particular case with zero quadrature weights except for one non-zero entry in y at the chosen pressure node. Therefore, the Lagrange multiplier $\lambda \cdot y$ corrects the right-hand side vector $\tilde{f}_4$ in Eq. (14) only locally at the chosen pressure node. In contrast, for the other two methods the quadrature weights are typically all nonzero such that the divergence inconsistency is distributed globally instead of only locally. In our following numerical experiments, we will always use "quad poly" as it clearly performs best out of the tested options.

4.4. Different convection directions

In this section, we look at results for the different convection directions $\boldsymbol{b}_i$ with $i = 1, 2, 3$ (see Eq. (17)). We set the viscosity to $\nu = 10^{-3}$, use polynomial augmentation $lcgd = 4333$, $s = h_u/h_p = 1.4$ and Solution 1 on the bunny-shaped domain. The results are shown in Fig. 7.

Convection $\boldsymbol{b}_3$ corresponds to the Stokes equations and results in the smallest relative errors e_2 for pressure and velocity when compared to the convections $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$. This is explained by the fact that there is no convection for $\boldsymbol{b}_3$ and therefore convection dominance cannot negatively impact the solution. Meanwhile, for convections $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$, the velocity error e_2 is up to an order of magnitude larger compared to the one for $\boldsymbol{b}_3$. Our discretization does not use a specific strategy to deal with the convection dominance, such as upwind stencils or hyperviscosity. Therefore, this behavior is expected. As seen before, the pressure error behaves qualitatively similar to the velocity error and quantitatively appears to be even slightly more affected by the convection dominance than the velocity error. The pressure errors for $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ can be up to two orders of magnitude larger than for $\boldsymbol{b}_3$ at small step widths h_u. Overall, the error discrepancy between $\boldsymbol{b}_3$ and the other two convections increases as h_u decreases.

4.5. Different degrees of polynomial augmentation

In this section we investigate the convergence behavior for different degrees of polynomial augmentation (which imply different stencil sizes). In Fig. 8 we show our results for Solution 2 (Fig. 8(a) and (b)) and Solution 3 (Fig. 8(c) and (d)) which behave qualitatively quite similar. Only for $lcgd = 2121$ do we see a clear difference: For Solution 2, using $lcgd = 2121$ or $lcgd = 3222$ achieve very similar errors whereas for Solution 3 the errors are indistinguishable for $lcgd = 2121$ and $lcgd = 2111$. A similar behavior has also been observed for higher degrees (e.g. $lcgd = 3232$) but is not included here. We currently do not have an explanation for this. However, we once more see a strong coupling between the velocity and pressure errors.

The spikes in the convergence behavior observed for $lcgd = 5444$ might be due to higher condition numbers of the system matrices. In [24] it was shown through numerical experiments for the Poisson equation that the condition number of the system matrix increases with the total number of nodes as well as the order of polynomial augmentation.

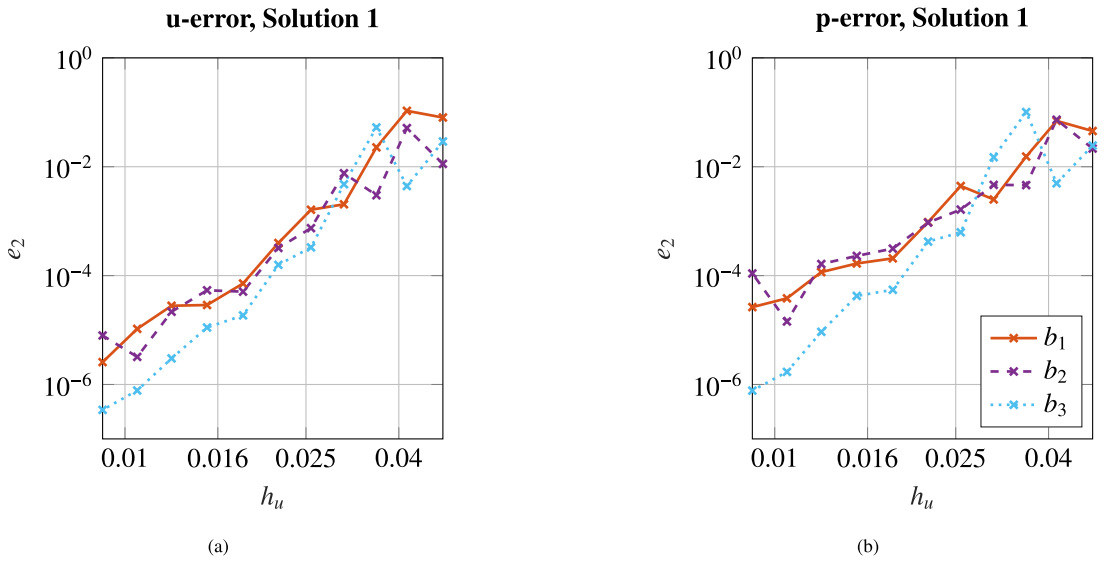


Fig. 7. Discretization errors Eq. (18) for different convections on the bunny for $\nu = 10^{-3}$, $lcgd = 4333$, $s = 1.4$, Solution 1.

Table 3
EOC for various values of polynomial augmentation $lcgd$ for Solution 2 (see Fig. 8(a) and (d)) and Solution 3 (see Fig. 8(c) and (d)).

	Solution 2		Solution 3	
$lcgd$	EOC velocity	EOC pressure	EOC velocity	EOC pressure
2111	2.0064	1.9371	1.7217	0.8125
2121	2.3794	1.4934	2.3769	1.2488
3222	2.0159	1.9600	2.2584	2.1984
4333	3.8760	3.8598	4.3550	3.3738
5444	6.9997	6.3637	4.6932	4.3559

We have used our numerical results to compute experimental orders of convergence (EOC) which are listed in Table 3. The EOC is calculated by fitting the function $f(x) = a \cdot x^b$ to the data points (h_u, e_2) which correspond to the five smallest values of h_u using `MATLAB's nlinfit` function. The EOC is then given by the parameter b.

While the general trend is for the EOC to increase with increasing degrees $lcgd$ of polynomial augmentation, we do not recognize a clear pattern how this increase depends on these parameters, especially since the EOCs are different for the two test problems with different exact solutions.

It seems though that even polynomial degrees should be preferred for the Laplacian as odd degrees do not give much improvement as can be seen when going from $lcgd = 2111$ to $lcgd = 3222$ in Table 3. For even degrees of augmentation for the Laplacian, the EOC of the velocity seems to be equal to said degree. This was also observed for a Poisson problem in [23] and a general elliptic equation in [12].

Similarly, in [11] the Oseen equations were solved on a square domain using a right-angled and a hexagonal node layout. For both node layouts, the EOCs for the velocity were roughly smaller by one than the polynomial augmentation degree. On the hexagonal node layout, the EOC of the pressure was equal to the degree of polynomial augmentation and on the rectangular node layout the EOC of was smaller by one.

In [27], it is proven that PHS can achieve at most a convergence rate of $\ell + 1 - D$ when approximating pointwise derivatives of order D with polynomial augmentation of degree ℓ in Sobolev spaces. These rates are also obtained experimentally in [9] for the Stokes equations.

Overall, the convergence orders observed in our experiments are in good agreement with the literature up to some smaller deviations.

4.6. *Varying viscosities*

We will now discuss the impact of the viscosity parameter ν on the errors. In Fig. 9(a) and (b) we see results for Solution 2 (nonpolyn. pressure) with convection b_1 and for $\nu = 10^0, 10^{-1}, 10^{-2}, 10^{-3}$.

We see a clear dependence of the velocity error on ν. In particular reducing ν by an order of magnitude increases the relative velocity error by roughly an order of magnitude. As discussed in Section 3.3, ideally we would want the error to remain unchanged

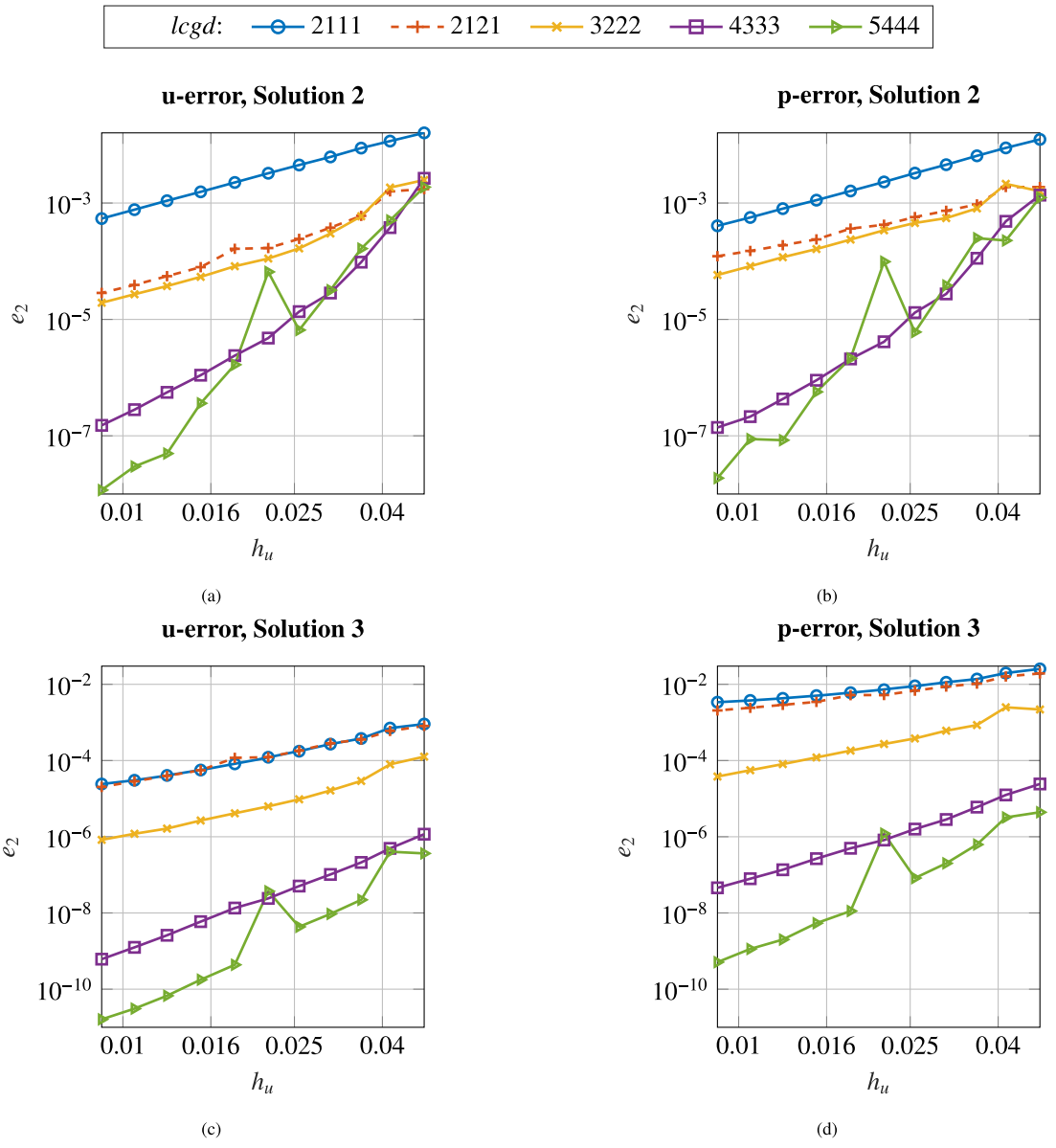


Fig. 8. Discretization errors Eq. (18) for different degrees of polynomial augmentation for Solution 2 (Fig. 8(a), 8(b)) and Solution 3 (Fig. 8(c) and (d)) with convection b_1 and $\nu = 10^{-2}$ on the cube, $s = 1.4$.

as our exact solution also did not change. For $\nu = 10^{-3}$ errors are even larger, which as mentioned in Section 4.4 is most likely due to effects associated with convection dominance. (All iterations converged in the sense that a relative residual below 10^{-12} was reached before 1024 steps.) The pressure errors remain almost the same for small ν. Only for $\nu = 10^{-3}$ do we see larger pressure errors.

In Fig. 9(c) and (d) results for Solution 3 (polynomial pressure) and convection b_1 are shown. Now the previously seen effect almost completely reverses and we see that the velocity errors remain almost the same when varying ν (except again $\nu = 10^{-3}$). Instead, the pressure error now changes when varying ν. Though contrary to the first case, now lowering ν also reduces the pressure error.

In Theorem 1 we predicted this behavior for the Stokes equations. We now see that the Oseen equations show a similar behavior at least for small viscosities ν where the behavior of Stokes and Oseen equations is expected to be similar.

We also performed the same test for the Stokes equations for viscosities $\nu \in [10^{-4}, 1]$. The results can be seen in Fig. 10. Here we see exactly the behavior predicted by Theorem 1.

As explained in Remark 3 if the pressure is a polynomial of certain degree and the degree of polynomial augmentation at least matches this then the norm of the error e_g vanishes. Therefore, the relative error for the velocity becomes independent of ν. This can be seen in Fig. 10(c) for the Stokes equations. Otherwise, we have a dependence of the velocity error on ν^{-1} as can be seen in

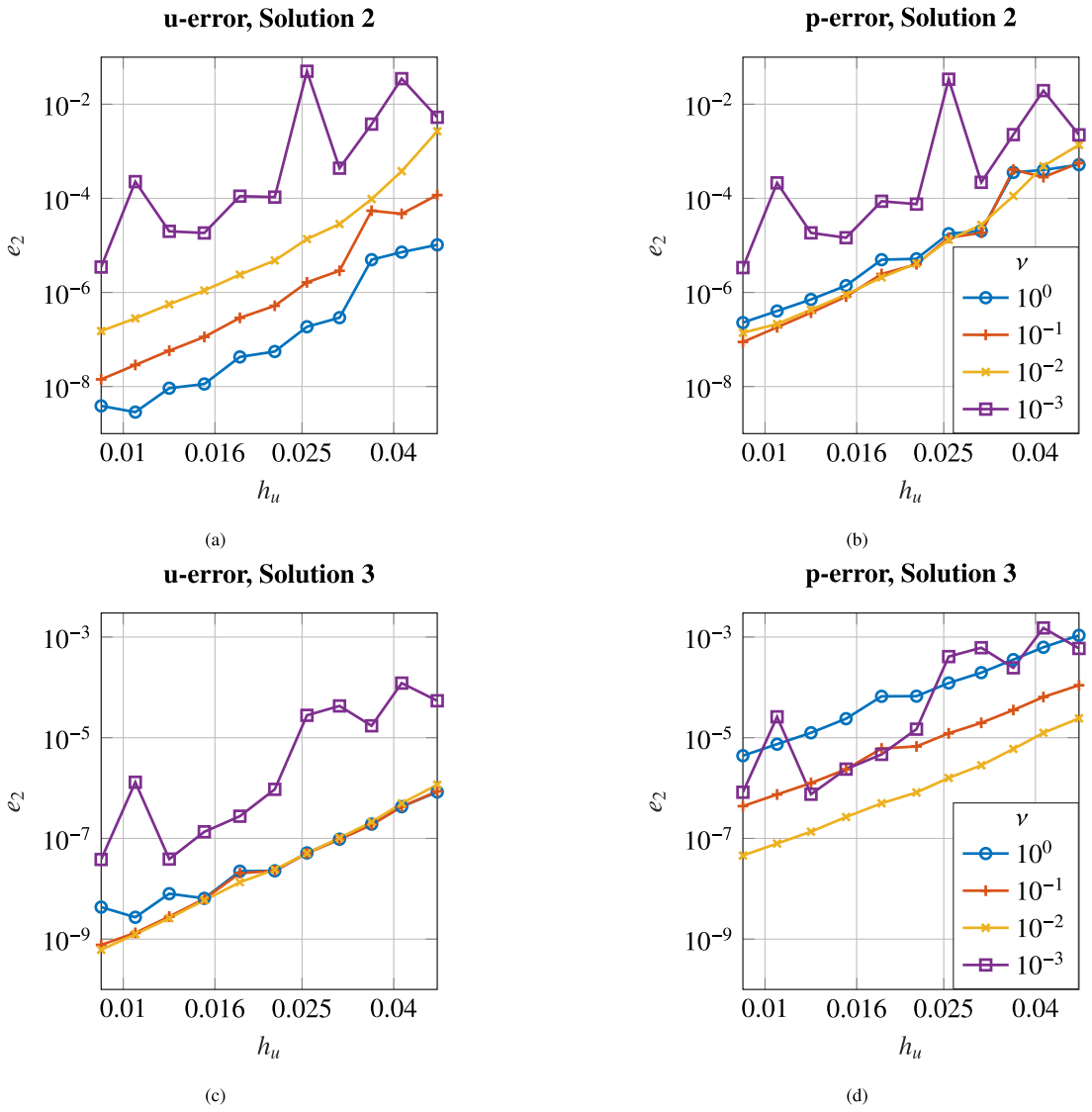


Fig. 9. Influence of parameter ν for Solution 2 (Fig. 9(a) and (b)) and Solution 3 (Fig. 9(c) and (d)) with convection b_1, $lcgd = 4333$, domain = cube.

Table 4
1-norm condition estimates for system matrix from Eq. (14) on the unit cube with convection b_1 and two different viscosities ν.

			1-norm condition estimate of Eq. (14)	
$N_{u,I}$	N_p	$3N_{u,I} + N_p$	$\nu = 10^{-2}$	$\nu = 10^{-3}$
5,976	2,173	20,102	2.059584e+06	2.059584e+06
8,042	2,933	27,060	3.321600e+06	3.321600e+06
10,429	3,809	35,097	4.842553e+06	2.688118e+07
20,788	7,614	69,979	1.161410e+07	1.508755e+07
33,346	12,236	112,275	2.175120e+07	5.311748e+07

Fig. 10(b). We also see the dependence of the pressure error on ν for Solution 3 as expected (see Fig. 10(d)). We believe that this not visible for Solution 2 (see Fig. 10(b)) because $\|e_d\| \ll \|e_g\|$.

4.7. Scalibility

In this section we briefly comment on time and storage complexity of the proposed method. Let $N := \max(N_u, N_p)$. Then the domain discretization described in Section 3.1 is of complexity $\mathcal{O}(N \log N)$ [16]. A kd-tree is used here which has storage complexity

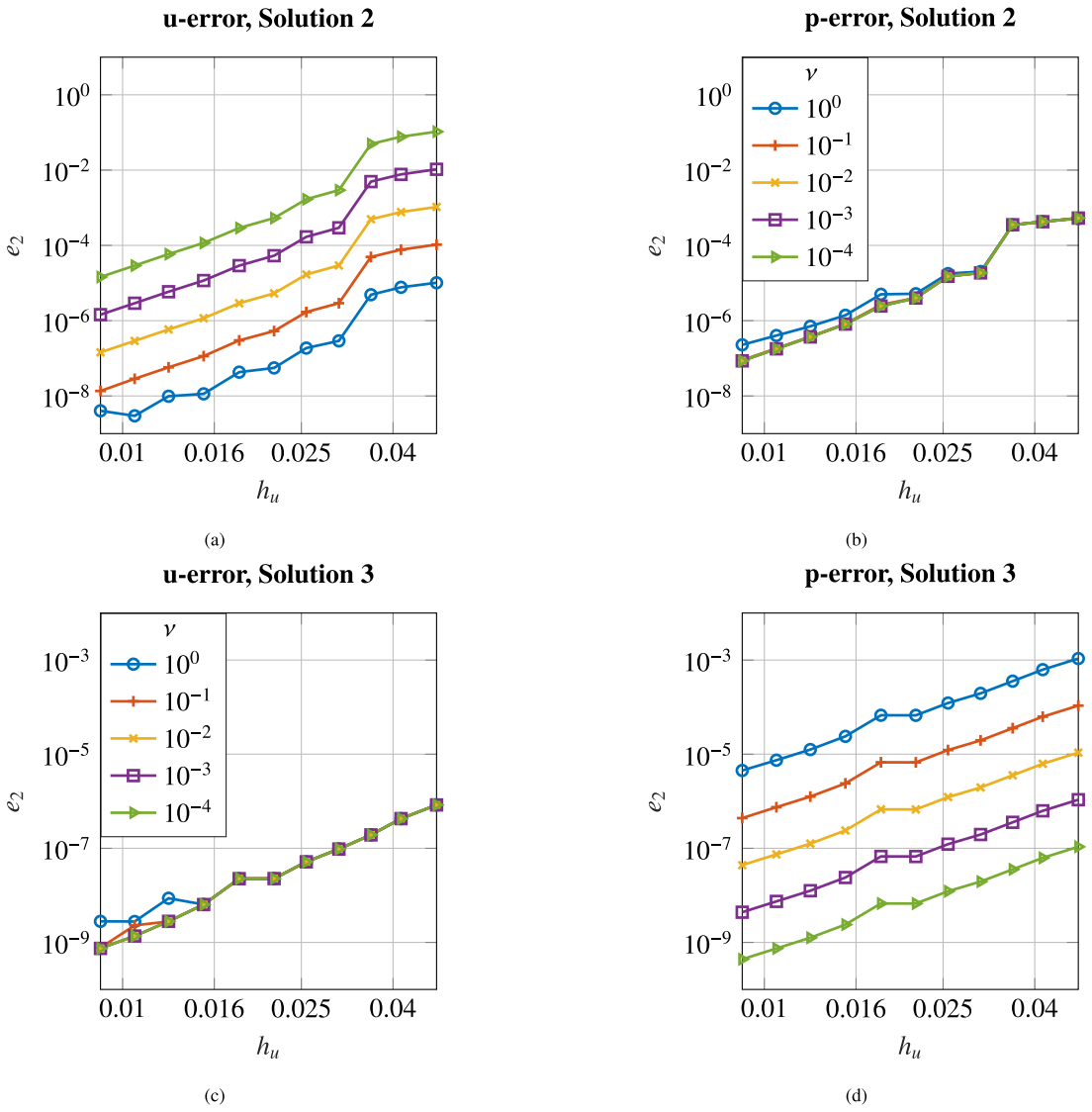


Fig. 10. Influence of parameter ν for Solution 2 (Fig. 9(a) and (b)) and Solution 3 (Fig. 9(c) and (d)) with convection b_3 (corresponds to Stokes equation), $lcgd = 4333$, domain = cube.

$\mathcal{O}(N)$ [40]. Storing the velocity and pressure nodes is also of storage complexity $\mathcal{O}(N)$. For constant step widths h_u, h_p a uniform-grid based spatial search structure could also be used which would improve the time complexity to $\mathcal{O}(N)$ (see [16] for details). Let n be the maximal stencil size used in the discretization of the Oseen equations. The task of finding the stecils of size n for each node can be done in $\mathcal{O}(nN \log N)$ when a kd-tree as used [40]. This is the case for the library nanoflann [36] which we use. The storage complexity for the kd-tree is once again $\mathcal{O}(N)$ and storing the stencils is $\mathcal{O}(nN)$. Computing the stencil weights for all nodes with a direct method such as partially pivoted LU factorization as we do is of complexity $\mathcal{O}(n^3 N)$. This task is also easily parallelized and `Medusa` [28] automatically parallelizes if compiled with OpenMP. Storing the weights in a sparse matrix is then of storage complexity $\mathcal{O}(nN)$. In total, we achieve time complexity $\mathcal{O}(N \log N)$ and storage complexity $\mathcal{O}(N)$. We conclude that the discretization is therefore scalable to large problems. Without going into too much detail, we mention that in our experience solving the resulting linear system Eq. (14) requires by far the most computational resources. The solution process has also been identified as the bottleneck in terms of computational time in [23, Section 3.4] which presented the RBF-FD method for solving Poisson's equation also utilizing `Medusa` [28] and the node generation algorithm from [16]. We leave exploring solution procedures for the resulting saddle point systems Eq. (14) for future work (see [41] for an overview). Another aspect related to the solution procedure is the condition number of the linear system. It is well known that the condition number increases in finite element methods as the size of the elements is decreased (the size of the linear system increases) or the discretization order is increased (leading to additional nonzero entries per row). We also observe this for our method and give a small example in Table 4. We present 1-norm condition estimates which are computed using Algorithms 4.1 from [42] which is implemented in the C++ library `Eigen` [43].

The condition estimates scale roughly linearly with the matrix size (with more variance for the case viscosity $\nu = 10^{-3}$).

5. Conclusion

We presented an almost completely meshless discretization (only requiring a surface mesh of the domain) of the Oseen equations in three dimensions using the PHS RBF-FD method. Different node arrangements and different strategies to make the pressure unique were described and illustrated numerically. Further we provided a novel theoretical insight into the behavior of the discretization error for the Stokes equations, showing a dependence of the discrete pressure and velocity on the viscosity parameter ν in Theorem 1. We performed numerical tests in three dimensions for the unit cube and a more complex bunny-shaped domain for a multitude of parameters and confirmed the theoretical predictions of Theorem 1. Moreover, we were able to reach small relative errors and high orders of convergence for all our tests.

We have neither discussed the iterative solution procedure nor the computational time to obtain discrete solutions. This will be part of future work where we will investigate different preconditioners to solve the linear systems arising from the presented discretization. We might also consider to include timings for the discretization procedure.

CRediT authorship contribution statement

Michael Koch: Writing – original draft, Visualization, Validation, Software, Investigation, Formal analysis; **Sabine Le Borne:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

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

Acknowledgements

The authors acknowledge the support by the Deutsche Forschungsgemeinschaft (DFG) within the Research Training Group GRK 2583 “Modeling, Simulation and Optimization of Fluid Dynamic Applications”.

We would also like to thank Dr. Jonas Grams for helpful discussions regarding implementation issues and letting us use part of his codebase.

References

[1] R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars, Mon. Not. R. Astron. Soc. 181 (3) (1977) 375–389. https://doi.org/10.1093/mnras/181.3.375
[2] T. Liszka, J. Orkisz, The finite difference method at arbitrary irregular grids and its application in applied mechanics, Comput. Struct. 11 (1–2) (1980) 83–95. https://doi.org/10.1016/0045-7949(80)90149-2
[3] J.-S. Chen, C. Pan, C.-T. Wu, W.K. Liu, et al., Reproducing kernel particle methods for large deformation analysis of non-linear structures, Comput. Methods Appl. Mech. Eng. 139 (1–4) (1996) 195–227. https://doi.org/10.1016/S0045-7825(96)01083-3
[4] A.I. Tolstykh, D.A. Shirobokov, On using radial basis functions in a finite difference mode with applications to elasticity problems, Comput. Mech. 33 (1) (2003) 68–79. https://doi.org/10.1007/s00466-003-0501-9
[5] N. Flyer, E. Lehto, S. Blaise, G.B. Wright, A. St-Cyr, et al., A guide to RBF-generated finite differences for nonlinear transport: shallow water simulations on a sphere, J. Comput. Phys. 231 (11) (2012) 4078–4095. https://doi.org/10.1016/j.jcp.2012.01.028
[6] V. Shankar, A.L. Fogelson, Hyperviscosity-based stabilization for radial basis function-finite difference (RBF-FD) discretizations of advection-diffusion equations, J. Comput. Phys. 372 (2018) 616–639. https://doi.org/10.1016/j.jcp.2018.06.036
[7] N. Flyer, G.A. Barnett, L.J. Wicker, et al., Enhancing finite differences with radial basis functions: experiments on the navier-stokes equations, J. Comput. Phys. 316 (2016) 39–62. https://doi.org/10.1016/j.jcp.2016.02.078
[8] P.K. Mishra, G.E. Fasshauer, K. Sen Mrinal, L. Ling, et al., A stabilized radial basis-finite difference (RBF-FD) method with hybrid kernels, Comput. Math. Appl. 77 (9) (2019) 2354–2368. https://doi.org/10.1016/j.camwa.2018.12.027
[9] A. Westermann, O. Davydov, A. Sokolov, S. Turek, et al., Stability and accuracy of a meshless finite difference method for the stokes equations, Internat. J. Numer. Methods Eng. 126 (3) (2025). https://doi.org/10.1002/nme.70000
[10] T. Sun, J. Li, J. Zhao, X. Feng, et al., Least-squares RBF-FD method for the incompressible stokes equations with the singular source, Numer. Heat Transf. Part A Appl. 75 (11) (2019) 739–752. https://doi.org/10.1080/10407782.2019.1608766
[11] L. Mu, X. Feng, Radial basis function finite difference method based on oseen iteration for solving two-dimensional navier-stokes equations, Entropy 25 (5) (2023) 804. https://doi.org/10.3390/e25050804
[12] V. Bayona, N. Flyer, B. Fornberg, G.A. Barnett, et al., On the role of polynomials in RBF-FD approximations: II. numerical solution of elliptic PDEs, J. Comput. Phys. 332 (2017) 257–273. https://doi.org/10.1016/j.jcp.2016.12.008
[13] D. Strzelczyk, M. Rot, G. Kosec, M. Matyka, et al., On h-refined meshless solution to Navier-Stokes problem in porous media: comparing meshless Lattice Boltzman Method with ACM RBF-FD approach, 2024, arXiv preprint arXiv:2404.14195.
[14] R. Zamolo, E. Nobile, Solution of incompressible fluid flow problems with heat transfer by means of an efficient RBF-FD meshless approach, Numer. Heat Transf. Part B Fundament. 75 (1) (2019) 19–42. https://doi.org/10.1080/10407790.2019.1580048
[15] T. Chu, O.T. Schmidt, RBF-FD discretization of the Navier-Stokes equations on scattered but staggered nodes, J. Comput. Phys. 474 (2023) 111756. https://doi.org/10.1016/j.jcp.2022.111756

[16] J. Slak, G. Kosec, On generation of node distributions for meshless PDE discretizations, SIAM J. Sci. Comput. 41 (5) (2019) A3202–A3229. https://doi.org/10.1137/18M1231456
[17] R. Zamolo, E. Nobile, Two algorithms for fast 2D node generation: application to RBF meshless discretization of diffusion problems and image halftoning, Comput. Math. Appl. 75 (12) (2018) 4305–4321. https://doi.org/10.1016/j.camwa.2018.03.031
[18] H. Wendland, Scattered data approximation, Cambridge Monogr. Appl. Comput. Math., Cambridge University Press, 2004. https://doi.org/10.1017/CBO9780511617539
[19] Y.L. Chan, L.H. Shen, C.T. Wu, D.L. Young, et al., A novel upwind-based local radial basis function differential quadrature method for convection-dominated flows, Comput. Fluids 89 (2014) 157–166. https://doi.org/10.1016/j.compfluid.2013.10.032
[20] Y.T. Gu, G.R. Liu, Meshless techniques for convection dominated problems, Comput. Mech. 38 (2) (2006) 171–182. https://doi.org/10.1007/s00466-005-0736-8
[21] Z. Vaupotič, M. Rot, G. Kosec, et al., Hyperviscosity stabilisation of the RBF-FD solution to natural convection, J. Phys. Conf. Ser. 2766 (1) (2024) 012160. https://doi.org/10.1088/1742-6596/2766/1/012160
[22] B. Fornberg, E. Lehto, Stabilization of RBF-generated finite difference methods for convective PDEs, J. Comput. Phys. 230 (6) (2011) 2270–2285. https://doi.org/10.1016/j.jcp.2010.12.014
[23] M. Jančič, J. Slak, G. Kosec, et al., Monomial augmentation guidelines for RBF-FD from accuracy vs. computational time perspective, J. Sci. Comput. 87 (1) (2021) 9. https://doi.org/10.1007/s10915-020-01401-y
[24] S. Le Borne, W. Leinen, Guidelines for RBF-FD discretization: numerical experiments on the interplay of a multitude of parameter choices, J. Sci. Comput. 95 (1) (2023) 8. https://doi.org/10.1007/s10915-023-02123-7
[25] B. Fornberg, N. Flyer, A primer on radial basis functions with applications to the geosciences, CBMS-NSF Regional Conf. Ser. in Appl. Math., SIAM, 2015. https://doi.org/10.1137/1.9781611974041
[26] R. Zamolo, E. Nobile, B. Šarler, et al., Novel multilevel techniques for convergence acceleration in the solution of systems of equations arising from RBF-FD meshless discretizations, J. Comput. Phys. 392 (2019) 311–334. https://doi.org/10.1016/j.jcp.2019.04.064
[27] O. Davydov, R. Schaback, Optimal stencils in sobolev spaces, IMA J. Numer. Anal. (2017). https://doi.org/10.1093/imanum/drx076
[28] J. Slak, G. Kosec, Medusa: a c++ library for solving PDEs using strong form mesh-free methods, ACM Trans. Math. Softw. 47 (3) (2021) 1–25. https://doi.org/10.1145/3450966
[29] A.P. Lawrence, M.E. Nielsen, B. Fornberg, et al., Node subsampling for multilevel meshfree elliptic PDE solvers, Comput. Math. Appl. 164 (2024) 79–94. https://doi.org/10.1016/j.camwa.2024.03.022
[30] J. Glaubitz, Stable high-order cubature formulas for experimental data, J. Comput. Phys. 447 (2021) 110693. https://doi.org/10.1016/j.jcp.2021.110693
[31] E.B. Chin, N. Sukumar, An efficient method to integrate polynomials over polytopes and curved solids, Comput. Aided Geom. Des. 82 (2020) 101914. https://doi.org/10.1016/j.cagd.2020.101914
[32] G.H. Golub, C.F. Van Loan, Matrix computations, Johns Hopkins Studies in the Mathematical Sciences, The Johns Hopkins University Press, Baltimore, 4th ed., 2013. https://doi.org/10.1137/1.9781421407944.
[33] O. Davydov, B.D. Esposti, Meshless quadrature formulas arising from numerical differentiation, 2024, arXiv preprint arXiv:2409.03567.
[34] V. John, A. Linke, C. Merdon, M. Neilan, L.G. Rebholz, et al., On the divergence constraint in mixed finite element methods for incompressible flows, SIAM Rev. 59 (3) (2017) 492–544. https://doi.org/10.1137/15M1047696
[35] A. Linke, On the role of the Helmholtz decomposition in mixed methods for incompressible flows and a new variational crime, Comput. Methods Appl. Mech. Eng. 268 (2014) 782–800. https://doi.org/10.1016/j.cma.2013.10.011
[36] J.L. Blanco, P.K. Rai, nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees, 2014, https://github.com/jlblancoc/nanoflann.
[37] V. John, Finite Element Methods for Incompressible Flow Problems, Springer International Publishing, 2016. https://doi.org/10.1007/978-3-319-45750-5
[38] H. Elman, D. Silvester, A. Wathen, Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics, Oxford University PressOxford, 2014. https://doi.org/10.1093/acprof:oso/9780199678792.001.0001
[39] J. Schöberl, C++11 Implementation of Finite Elements in NGSolve, ASC Reports (2014). http://hdl.handle.net/20.500.12708/28346.
[40] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517. https://doi.org/10.1145/361002.361007
[41] M. Benzi, G.H. Golub, J. Liesen, et al., Numerical solution of saddle point problems, Acta Numerica 14 (2005) 1–137. https://doi.org/10.1017/S0962492904000212
[42] N.J. Higham, FORTRAN Codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation, ACM Trans. Math. Softw. 14 (4) (1988) 381–396. https://doi.org/10.1145/50063.214386
[43] G. Guennebaud, B. Jacob, et al., Eigen v3, 2010, http://eigen.tuxfamily.org.