Simulation of particulate
electrodynamic flows with the Subspace
Projection Method

Simulation von partikelbehafteten elektrodynamischen
Strömungen mit der Unterraumprojektionsmethode

Der Naturwissenschaftlichen Fakultät der
Friedrich–Alexander–Universität Erlangen–Nürnberg
zur Erlangung des Doktorgrades
Dr. rer. nat.

vorgelegt von
Rodolphe Prignitz
aus
Nürnberg
Für Kristina, Viviane und Annick
Acknowledgment

First, I would like to express my very great appreciation to my supervisor Eberhard Bänsch who supported me throughout my thesis with his knowledge especially on error estimation whilst allowing me to work in my own way.

My special thanks are extended to the whole AM3 team for the friendly atmosphere and their encouragement over all the years. The policy of open doors created a enjoyable work climate where I acquired a lot of my mathematical and numerical knowledge through a seemingly endless number of discussions. I attribute the level of my thesis to the insightful comments and hard questions of my colleagues. I lay great stress upon the advice concerning software development given by Stephan Weller, the crucial questions posed by Kathrin Bäumler and the all kind of questions asked by Steffen Basting.

In my daily work, I have been blessed with a group of assistant scientists, Harald Karpinski, Alicja Kerschbaum and Andreas Reitsam, who did fine work.

I am particularly grateful for the assistance in proof reading given by Eberhard Bänsch, Dmitri Kuzmin and Kristina Schwegler. Similarly, I wish to thank everyone who made a contribution to completing my thesis.

I should not forget to mention the very productive work with my project partners, especially Jochen Schmidt (LFG) and Georg Landmesser (Océ). I liked the scientific atmosphere where open discussions where always welcomed. The research presented here was funded by the Bayerische Forschungsstiftung (BFS), Grant No. 813-08. This is gratefully acknowledged.

Besides science, a special thank goes to Patrick Luff and all other players for the recreative table soccer sessions.

Last but not least I would like to thank my family for their ongoing support and especially Kristina for her love and everything else she gave me – Thank you :-****
Abstract

A novel finite element method for the 3d simulation of (many) particles in a Newtonian carrier liquid is presented. The method features the celebrated one domain approach to simplify the spatial discretization, a newly developed subspace projection method to account for the rigid body motion within the particles and an operator splitting to decouple the nonlinearities. Combined with local mesh refinement the method results in a fast and accurate algorithm which is, in addition conceptually simple to implement.

Validation is achieved using the sedimentation of a single particle and comparing the resulting drag coefficient with theoretical and experimental results. Furthermore, a viscometer is considered where the effective viscosity of a particle laden fluid is compared with analytic results.

Furthermore, a method for the solution of the Nernst–Planck–Poisson equations is presented. These equations describe the distribution of the concentration of charged substances in a fluid. Validation is carried out by stationary solutions of the equations.

Finally, both methods are combined for the simulation of particulate electrodynamic flows.
Zusammenfassung


Zur Validierung der Methode wird die Sedimentation eines Einzelpartikels betrachtet. Der daraus resultierende Widerstandsbeiwert wird mit theoretischen und experimentellen Ergebnissen verglichen. Mit der Simulation eines Viskometers wird die effektive Viskosität eines mit vielen Partikeln versetzten Fluids berechnet und ebenfalls mit analytischen Resultaten verglichen.


Zum Schluss werden beide Methoden zur Simulation von ionenhaltigen, partikelbeharteten Strömungen benutzt.

Partikelbehartete Strömungen, Finite Elemente, Navier–Stokes, Nernst–Planck–Poisson, Unterraumprojektion, Validierung
Contents

1 Introduction 3

2 An overview of the Subspace Projection Method (SPM) 7
  2.1 A General Framework 7
  2.2 Model Problem 11
    2.2.1 Definition 11
    2.2.2 Weak Formulation 12
    2.2.3 Discretization 13
    2.2.4 Representation of $P$ 16
    2.2.5 Error Estimate for the Model Problem in $X_c$ 17
    2.2.6 Example 18

3 Particulate Flows (PF) 21
  3.1 Mathematical Formulation 21
    3.1.1 Model 21
    3.1.2 Weak Formulation 23
  3.2 Numerical Method 27
    3.2.1 Splitting by Time Discretization 27
    3.2.2 Spatial Discretization 31
  3.3 Error Estimate 38
    3.3.1 Example 41
  3.4 Adaptivity 42
  3.5 Preconditioning 43
  3.6 Particle Interaction 49
    3.6.1 Newton’s Method for the Predictor 49
    3.6.2 Barnes–Hut Algorithm 51
    3.6.3 Short Term Repulsive Forces 53
  3.7 Validation 55
3.7.1 Sedimentation ................................................. 55
3.7.2 Viscometer .................................................. 56
3.7.3 Drafting Kissing Tumbling ................................. 60

4 Nernst–Planck–Poisson Model (NPP) .......................... 63
  4.1 Dimensionless Formulation ................................. 64
  4.2 Time Discretization .......................................... 65
  4.3 Spatial Discretization ........................................ 66
  4.4 Stabilization .................................................. 70
  4.5 Validation ..................................................... 72
    4.5.1 Weakly Coupled NPP Equations ...................... 72
    4.5.2 Strongly Coupled NPP Equations ...................... 73

5 Electrodynamical Particulate Flows (PFNPP) ............. 79
  5.1 Model ........................................................ 79
  5.2 Numerical Method .......................................... 80
  5.3 Validation .................................................. 83
    5.3.1 A Single Particle in a Capacitor .................... 83

6 Conclusions & Outlook .......................................... 87
  6.1 Conclusions ................................................. 87
  6.2 Outlook ..................................................... 87

A Appendix .......................................................... 89
Chapter 1

Introduction

Particulate flow (PF), i.e. the flow of a carrier fluid loaded with particles, plays an important role in many technical applications. Let us mention reactors, fluidized beds, production of nano particles and many more. There exists a hierarchy of models how to describe the particulate phase and how to describe the interaction between particles and fluid. For a comprehensive list of references we refer to the articles of Esmaelli & Tryggvason [25] and Hu [37].

For certain applications it is mandatory to describe the fluid–particle interaction and also a possible particle–particle interaction in full detail without simplified parametrizations. Computational methods based on such full models are called direct numerical simulations.

One of the most important points in simulating particulate flow is the numerical representation of the particles’ geometry. In Feng et al. and Johnson & Tezduyar [26, 39] a remeshing technique was used to explicitly follow the geometry in time; Wan and Turek [59] introduced a mesh deformation technique and Glowinski et al. [31] used Lagrange multipliers on regular grids. Also immersed boundary methods are very popular, for example LeVeque & Li and Veeramani et al. [44, 58]. Distributed Lagrange multipliers to account for the stress boundary condition are used in Bönisch & Heuveline and Bönisch et al. [15, 14]. In Maury [48] a projection based method was already introduced, still following explicitly the geometry, thus requiring remeshing.

Analytical results regarding existence, uniqueness and qualitative behavior of solutions can be found for instance in Galdi and Serre [29, 55].

The approach presented here is based on the one domain approach by [55, 31], but differs from the above mentioned articles in one or several aspects since it

• does not require an explicit meshing of the particles’ domain;

• does not need an explicit evaluation of forces;
• uses a *subspace projection method* to account for the constraint of rigid body motion within the particles, thus avoiding a saddle point problem for this constraint;

• uses time dependent adaptively refined meshes to provide the necessary geometric resolution.

The method was introduced very briefly in [51] and in much more detail in [52]. It turns out that this novel method is therefore easy to implement and rather efficient, see Fig. 1.1. Only few modules have to be added to an existing standard software.

In a lot of the aforementioned industrial application, the size of the particles is so small, that Coulomb forces and the interaction with solute charge carriers in the particle laden fluid play an important role. To take these effects into account the model is extended with the Nernst–Planck–Poisson model (NPP). For an introduction to the NPP model we refer to [19, 11, 42, 45, 61] and references therein. We present the derivation of a numerical method for the solution of the NPP model, including a justification for the algorithms used.

This thesis is organized as follows.

The mathematical foundation for the *Subspace Projection Method* is presented in Chapter 2. In order to better understand the numerical method, the SPM is applied to a simple model problem for which we also present an error bound and a numerical experiment to prove the sharpness of this bound.

In Chapter 3, the mathematical model and its dimensionless formulation as well as the weak formulation of the one domain approach are presented for the particulate flow model (PF). Large parts of this chapter are taken from our article [52]. In its sections, the numerical method is explained in detail. Each of them addresses the necessary steps of the overall procedure: the time splitting, the Navier–Stokes solver, the subspace projection method (SPM), adaptivity in space, preconditioning and the particle–particle interaction. Moreover, an error estimate for the core subproblem of the SPM is given. At the end of Chapter 3, we discuss two computational experiments for validating the code and its underlying methods: the sedimentation of a single particle as well as the rheological behavior of particle laden fluids are studied. As a qualitative benchmark we additionally include the well known drafting kissing tumbling experiment.
The numerical treatment of the Nernst–Planck–Poisson model (NPP) is derived in Chapter 4. We address the issues of order reduction for higher order time discretization, and stability of the space discretization for the convection dominated case. We finish Chapter 4 with two analytical test cases.

Both models, the PF model and the NPP model, are combined in Chapter 5. The numerical treatment based on the methods presented in the previous chapters is given. A simple test case shows reasonable results for the numerical treatment of the combined model.

Figure 1.1: *Demonstration of the methods efficiency. 10000 particles immersed in a driven cavity are simulated.*
Chapter 2

An overview of the Subspace Projection Method (SPM)

2.1 A General Framework

In this thesis the Subspace Projection Method (SPM) allows us to give a simple formulation of the discrete problem coming from the equations describing particulate flows. The weak solution of this problem is found in a function space which includes the rigid body restriction of the particle. The aim of the SPM is to incorporate this restriction in an easy way to the discrete setting, such that standard discrete spaces could be used. This allows an easy and fast implementation of the given problem. The SPM is not restricted to particulate flows, therefore the abstract formulation of the SPM is presented here. Let $L$ and $H_c$ be Hilbert spaces, $(\cdot, \cdot)_L$ and $(\cdot, \cdot)_H$ their respective scalar products and $a : H_c \times H_c \rightarrow \mathbb{R}$ a bounded, symmetric and coercive bilinear form. Moreover, let us assume a continuous embedding $H_c \hookrightarrow L$. According to the Lax–Milgram theorem, the following problem

Find $u$ in $H_c$ such that

$$a(u, v) = (f, v)_L \quad \forall v \in H_c$$

(2.1)

is well-posed for all $f \in L$, i.e. the solution $u$ exists and is unique. Additionally, it holds that $\|u\|_H \leq C\|f\|_L$ for a constant $C$, that means $u$ is continuously bounded by the data $f$. Let $\{X_c(h)\}_{h>0}$ be a family of finite dimensional Hilbert spaces which fulfill the approximation property

$$\forall v \in H_c, \quad \lim_{h \rightarrow 0} \left( \inf_{v_h \in X_c(h)} \|v - v_h\|_H \right) = 0,$$

(2.2)
for some quantity $h$ which measures the degree of approximation, e.g. the mesh width $h$ for a finite element approximation. Let $h$ be fixed. For the sake of simplicity we write $X_c$ instead of $X_c(h)$. The space $X_c$ is said to be conformal if $X_c \subset H_c$, and nonconformal otherwise. In the context of the SPM nonconformal settings are allowed. An approximate solution of (2.1) can be found by Galerkin’s method where the space $H_c$ is replaced by the finite dimensional Hilbert space $X_c$ in the weak formulation.

**Theorem 1** (Banach–Nečas–Babuška). *The well–posedness of the discrete problem*

\[
\text{Find } u_c \text{ in } X_c \text{ such that } \quad \forall v_c \in X_c \quad a(u_c, v_c) = (f, v_c)_L \tag{2.3}
\]

is equivalent to the two following conditions

\[
\exists \alpha_h > 0, \quad \inf_{u_c \in X_c} \sup_{v_c \in X_c} \frac{a(u_c, v_c)}{\|u_c\|_H \|v_c\|_H} \geq \alpha_h, \quad (BNB1)
\]

\[
\forall v_c \in X_c \quad (\forall u_c \in X_c, a(u_c, v_c) = 0) \Rightarrow (v_c = 0). \quad (BNB2)
\]

**Proof.** See [24], Theorem 2.6. In the situation above the solution space $X_c$ is equal to the test space $X_c$. Since $X_c$ is finite dimensional, we can conclude from the rank theorem that $(BNB1) \iff (BNB2)$. \hfill \Box

In the remainder we assume the two conditions to hold. Since $X_c$ is finite dimensional, owing to Riesz’s representation theorem the bilinear form $a$ induces a linear operator $A : X_c \to X_c$ with respect to the $L$ scalar product such that (2.3) is equivalent to

\[
\text{Find } u_c \text{ in } X_c \text{ such that } \quad (Au_c, v_c)_L = (f, v_c)_L \quad \forall v_c \in X_c. \tag{2.4}
\]

In practice, depending on the exact definition of $H_c$, it might be challenging to construct such a space $X_c$. Whenever a different space $X_c$ is considered it must be implemented separately. This occurs especially when time dependent problems are considered, where $H_c$, and with it $X_c$, are different from time step to time step. Even for stationary problems it might be desired to use standard finite element software with standard finite element spaces. To circumvent the explicit representation of the nonstandard space $X_c$, a convenient finite dimensional approximating subspace $X \hookrightarrow L$ with $X_c \subseteq X$ is taken. Let $P : X \to X_c$ be a projection. With this projection
it is possible to represent elements in $X_c$ by elements in $X$. Problem (2.4) is then rewritten

Find $u$ in $X$ such that

$$(APu, P v)_L = (f, P v)_L$$

$\forall v \in X,$ \hspace{1cm} (2.5)

and obtain the solution $u_c = Pu$. The solution $u$ of Eq. (2.5) is only determined up to the null space of $P$. However, the right solution $u_c = Pu$ of the initial Eq. (2.4) is still unique. Let $P^*$ be the adjoint of $P$ with respect to the $L$ scalar product, then the above problem is equivalent to

Solve

$$P^* APu = P^* f,$$ \hspace{1cm} (2.6)

and obtain the solution $u_c = Pu$.

The space $X$ is finite dimensional, so there is a finite basis $\{\phi_1, \ldots, \phi_n\}$ and a canonical isomorphism $I : X \rightarrow \mathbb{R}^n$ mapping the elements of $X$ to a standard vector in $\mathbb{R}^n$. The system matrix $A \in \mathbb{R}^{n \times n}$ is then given by

$A_{i,j} = a(\phi_j, \phi_i) \hspace{1cm} i, j = 1, \ldots, n$ \hspace{1cm} (2.7)

and likewise the right hand side $\vec{f} = (f_1, \ldots, f_n)^T$, $f_i = (f, \phi_i)_L$. Moreover, denote by $P \in \mathbb{R}^{n \times n}$ the matrix representation of $P$ such that the following diagram commutes

$\begin{array}{c}
X \\
\downarrow I \\
\mathbb{R}^n \\
\uparrow P \\
X_c \\
\downarrow I \\
\mathbb{R}^n
\end{array}$

This is, $PIv = IPv$ for all $v \in X$. We have the following result:

**Lemma 2.** Eq. (2.4) is equivalent to Eq. (2.6). Let $\vec{e}_i \in \mathbb{R}^n$ be the $i$-th standard basis vector with $(e_i)_j = \delta_{ij}$. Assume $f \in X$ and $I \phi_i = \vec{e}_i$. Then, Eq. (2.6) in turn is equivalent to

$$P^T AP \vec{u} = P^T \vec{f},$$ \hspace{1cm} (2.8)

where $\vec{u} = Iu$.

**Remark 3.** The condition $I \phi_i = \vec{e}_i$ is true for Lagrange finite elements.
Proof. First, the equivalence of (2.4) and (2.6) is proved. For the forward direction use the adjoint operator property \((P^*u,v)_L = (u,Pv)_L\) for all \(u,v \in X\) on Eq. (2.4), yielding
\[
(P^*APu,v)_L = (P^*f,v)_L \quad \forall v \in X.
\]
This identity holds for all elements in \(X\), thus (2.6) follows. The backward direction is now straightforward.

For the second part of the proof we need to define the mass matrix \(M_{i,j} := (\phi_j,\phi_i)_L\). \(M\) is symmetric and fulfills the following relation between the \(X\) scalar product \((\cdot,\cdot)_L\) and the Euclidean \(\mathbb{R}^n\) scalar product \(\langle \cdot,\cdot \rangle\):
\[
\langle M\bar{I}u,\bar{I}v \rangle = (u,v)_L.
\]
To see this, consider
\[
(\phi_j,\phi_i)_L = M_{i,j} = \langle M\bar{e}_j,\bar{e}_i \rangle = \langle M\bar{I}\phi_j,\bar{I}\phi_i \rangle
\]
and use the linearity of each argument of the scalar products. From this relationship and with the definition of \(A\) we get \(A\bar{I}v = M\bar{I}A\bar{v}\) for all \(v \in X\) and \(\bar{f} = M\bar{I}f\) if \(f \in X\). In a similar manner we derive \(M\bar{I}P^*v = P^TM\bar{I}v\) for all \(v \in X\). To derive this, use \(P\bar{I}u = \bar{I}P\bar{u}\) and test with an arbitrary vector \(M\bar{I}v\) (\(M\) and \(\bar{I}\) are bijective), yielding \(\langle \bar{u},P^TM\bar{I}v \rangle\) for the left hand side and \(\langle \bar{u},M\bar{I}P^*v \rangle\) for the right hand side. Inserting these relations in (2.6) directly leads to (2.8) and vice versa. \(\square\)

With a Krylov space method as iterative solver, the evaluation of the matrix \(P^TAP\) is realized by pre- and post–multiplying the iterates by \(P\) and \(P^T\), respectively. Solvers of this type are not sensitive to the null space generated by \(P\) as long as the right hand side is in the range of the operator. Let \(\ker\) be the null space and \(\text{im}\) the range of an operator. It is easy to see that \(\ker(P) = \ker(P^TAP)\) and \(\text{im}(P^T) = \text{im}(P^TAP)\) if \(A\) is regular. From the fundamental theorem of linear algebra we have \(\ker(P) = \text{im}(P^T)^\perp\) and, consequently, \(\ker(P^TAP) = \text{im}(P^TAP)^\perp\). The \(n\)–th iterate of a Krylov space method is defined by
\[
u_n = u_0 + \sum_{i=1}^n \lambda_i (P^TAP)^i r_0,\tag{2.9}
\]
where \(u_0\) is the initial guess and \(r_0 = P^TAPu_0 - P^TF\) is the initial residual for a right hand side \(f\). The coefficients \(\lambda_i\) are determined by the precise Krylov space method, e.g. GMRes or BiCGStab. As a consequence of \(r_0 \in \text{im}(P^TAP)\), the update \(\sum_{i=1}^n \lambda_i (P^TAP)^i r_0\) is also an element of \(\text{im}(P^TAP)\) which is orthogonal to the null space \(\ker(P^TAP)\). That means the null space is automatically factored out by
any Krylov space method. If \( u_0 \in \text{im}(P^TAP) \) then \( u_n \) is the unique solution (for large \( n \)), else the application of \( P \) to the iterative solution \( u_n \) does not only give the right solution of the discrete system but also factors out the null space.

Solving the linear system (2.6) with the system matrix \( A \) obtained from the space \( X \) and the help of the projection \( P \) is called Subspace Projection Method (SPM).

### 2.2 Model Problem

The subspace projection method is applied to a model problem in this section. In the weak formulation a restricted function space \( H_c \) will naturally arise. After the mathematical treatment error bounds will be derived, followed by a numerical example.

#### 2.2.1 Definition

Consider the geometric situation from Figure 2.1. \( \Omega \) is an open domain with an outer boundary \( \Gamma \), and \( \Pi \) is an open domain inside of \( \Omega \). \( \Pi \) and \( \Omega \) share the same boundary \( \partial \Pi \) but they do not intersect, \( \Pi \cap \Omega = \emptyset \). Suppose \( u \in H^2(\Omega) \), then the following set of coupled equations is called the model problem:

\[
\begin{align*}
-\Delta u &= f & \text{in } \Omega, \\
u &= 0 & \text{on } \Gamma, \\
u &= U & \text{on } \partial \Pi, \\
U &= F - \int_{\partial \Pi} \partial_n u \, ds.
\end{align*}
\]

The unknown function \( u \) and the scalar value \( U \) are strongly coupled by (2.10c) and (2.10d). This model problem is artificial but it is mathematically similar to the PF model considered in the next chapter.
2.2.2 Weak Formulation

For the derivation of the weak formulation an appropriate function space is defined

\[ H_c(\Omega) = \{ (v, V) \in H^1(\Omega) \times \mathbb{R} | v|_\Gamma = 0, v|_\Pi = V \}. \]  \hspace{1cm} (2.11)

The weak formulation is derived by testing (2.10a) with \( v \in \{ w | (w, V) \in H_c(\Omega) \} \) and integrating over \( \Omega \).

\[
\int_{\Omega} f v \, dx = - \int_{\Omega} \Delta u v \, dx = \int_{\Omega} \nabla u \cdot \nabla v \, dx - \int_{\partial \Pi} \partial_n u v \, dx = \int_{\Omega} \nabla u \cdot \nabla v \, dx - \int_{\partial \Pi} \partial_n u \, dx V = \int_{\Omega} \nabla u \cdot \nabla v \, dx + UV - FV. \]  \hspace{1cm} (2.12)

Integrating by parts and inserting (2.10d) in the last expression, the final formulation is received.

Find \((u, U) \in H_c(\Omega)\) such that

\[
\int_{\Omega} \nabla u \cdot \nabla v \, dx + UV = \int_{\Omega} f v \, dx + FV \quad \forall (v, V) \in H_c(\Omega). \]  \hspace{1cm} (2.13)

For numerical reasons, which are important for the simulation of particulate flows, a one domain approach should be considered in the weak formulation. That means the weak formulation is not only valid on \( \Omega \), but also in the combined domain \( \Omega_c = \Omega \cup \Pi \cup \partial \Pi \). To this end, formulation (2.13) is extended. First we define an appropriate function space on the combined domain \( \Omega_c \)

\[ H_c(\Omega_c) := \{ (v, V) \in H^1(\Omega_c) \times \mathbb{R} | v|_\Gamma = 0, v|_\Pi = V \}. \]

This space equipped with the scalar product

\[ ((u, U), (v, V))_H := (\nabla u, \nabla v)_{L^2} + UV \]  \hspace{1cm} (2.14)

is a Hilbert space. Additionally, the \( L \) scalar product is defined as

\[ ((u, U), (v, V))_L := (u, v)_{L^2} + UV. \]  \hspace{1cm} (2.15)

With \( \nabla v|_\Pi = 0 \) for all \( v \in \{ w | (w, V) \in H_c(\Omega_c) \} \) and \( \tilde{f}|_\Omega = f \) a continuation of \( f \), we receive

Find \((u, U) \in H_c(\Omega_c)\) such that

\[
\int_{\Omega_c} \nabla u \cdot \nabla v \, dx + UV = \int_{\Omega_c} \tilde{f} v \, dx + \tilde{F} V \quad \forall (v, V) \in H_c(\Omega_c), \hspace{1cm} (2.16a)
\]

\[ \tilde{F} = F - \int_{\Pi} \tilde{f} \, dx. \]  \hspace{1cm} (2.16b)
This weak formulation in terms of bilinear forms then reads

\[
\text{Find } (u, U) \in H_c(\Omega_c) \text{ such that } \quad ((u, U), (v, V))_H = ((\tilde{f}, \tilde{F}), (v, V))_L.
\] (2.17)

From the Lax–Milgram theorem we conclude well–posedness of (2.17).

### 2.2.3 Discretization

Let \( T \) be a triangulation of the combined domain \( \Omega_c \) with typical mesh width \( h \) and \( \mathcal{K}^p \) the space of polynomials of degree \( p \). The standard linear finite element space augmented by one dimension to represent the scalar variable \( U \) is then defined as

\[
X := \{ (v, V) \in C^0(\Omega_c) \times \mathbb{R} \mid v_{|T} = 0, v_{|T \in \mathcal{T}} \in \mathcal{K}^1(T) \forall T \in \mathcal{T} \}, \quad (2.18)
\]

where \( X \) is a finite dimensional space and \( \mathbb{R}^{nd} \) its canonical isomorphic representation. The basis of \( X \) is

\[
\{ (\tilde{\phi}_1, 0), \ldots, (\tilde{\phi}_{nk}, 0), (0, 1) \} = \{ \phi_1, \ldots, \phi_{nd} \}. \quad (2.19)
\]

\( \tilde{\phi}_i \) are the Lagrange nodal basis functions corresponding to the grid while \( \phi_i \) are the basis elements of \( X \). For the model problem we have \( nd = nk + 1 \) where \( nk \) is the number of inner nodes of the mesh \( T \). Let

\[
\mathcal{I} := \left( (v, V) \mapsto \vec{v} := [v_1, \ldots, v_{nk}, V]^T : X \to \mathbb{R}^{nd}, \right) \quad (2.20)
\]

be the canonical isomorphism which maps \( X \) to the standard vector space \( \mathbb{R}^{nd} \). The coefficients \( v_i \) are well defined by the unique decomposition \( v = \sum_i v_i \tilde{\phi}_i \). \( \langle \vec{u}, \vec{v} \rangle \) is the standard scalar product on \( \mathbb{R}^{nd} \). Two index sets \( I_1 \) and \( I_2 \) are needed to define the projection operator \( \mathcal{P} \).

\[
I_1 := \left\{ i \mid \text{supp}(\tilde{\phi}_i) \cap \{ T \in \mathcal{T} \mid T \cap \Pi \neq \emptyset \} \neq \emptyset \right\}, \quad (2.21)
\]

\[
I_2 := \left\{ i \mid \text{supp}(\tilde{\phi}_i) \cap \{ T \in \mathcal{T} \mid T \subseteq \Pi \} \neq \emptyset \right\}. \quad (2.22)
\]

For a visualization, see Figure 2.2. The respective index sets contain all nodes which belong to the gray areas. This leads to two different spaces

\[
X_{c,1} := \{ (v, V) \in X \mid (v, V) = \mathcal{P}_1(v, V) \} = \ker_X(I - \mathcal{P}_1), \quad (2.23)
\]

\[
X_{c,2} := \{ (v, V) \in X \mid (v, V) = \mathcal{P}_2(v, V) \} = \ker_X(I - \mathcal{P}_2), \quad (2.24)
\]
where

\[ P_k(v, V) := \left( \sum_{i=1}^{n_k} v_i \tilde{\phi}_i - \sum_{i \in I_k} (v_i - V) \tilde{\phi}_i, V \right), \]  

(2.25)

and \( \ker_X(I - P_i) \) is the null space of \( I - P_i \) with respect to \( X \). The first variant \( X_{c,1} \) is a subspace of \( H_c(\Omega_c) \) while \( X_{c,2} \) is not. The reason to use \( X_{c,2} \) and therefore a nonconformal approximation is its simpler evaluation. Only nodes inside the particle must be considered. Both spaces also fulfill the approximation property (2.2) as the gray areas in Fig. 2.2 will converge to the black boundary from both directions. Besides the algorithmic advantage of alternative \( X_{c,2} \) there is no analytic disadvantage. The following 1d result emphasizes this.

![Figure 2.2: Visualization of the two different function spaces \( X_{c,1} \) and \( X_{c,2} \). The condition \( v = V \) holds in the gray areas, the black circle is the boundary \( \partial \Pi \).](image1)

![Figure 2.3: Visualization of an interpolation in the two finite element spaces \( X_{c,1} \) and \( X_{c,2} \).](image2)

**Lemma 4.** Let \( \Omega_c \subset \mathbb{R} \) be a domain. The two interpolation operators \( I_1 : H_c(\Omega_c) \to \)
$X_{c,1}$ and $I_2 : H_c(\Omega_c) \to X_{c,2}$ have the error bounds

$$\|v - I_1 v\|_H \leq c h + c_1 h^{\frac{1}{2}}, \quad \|v - I_2 v\|_H \leq c h + c_2 h^{\frac{1}{2}},$$  

(2.26)

with constants $c$, $c_1$ and $c_2$. Additionally, $c_1 = c_2$ holds true.

**Proof.** An interpolation operator for $H_c(\Omega_c)$ has the form $I(v, V) = (\hat{I}v, V)$, i.e. it acts only on the $v$ part of the elements of $H_c(\Omega_c)$. In the remainder we will abuse this notation and only use $I$, disregarding if it acts on $v$ or on $(v, V)$. The boundary $\partial \Pi$ contains two points. Without loss of generality, assume that these two points are not points of the triangulation $T$ of $\Omega_c$. Then the triangulation $T_{\partial \Pi}$ is $T$ extended by these two points. $T_{\partial \Pi}$ defines a new finite element space $X_{\partial \Pi}$. For this space standard interpolation $I_C : H_c(\Omega_c) \to X_{\partial \Pi}$ with its usual error bounds can be applied. Now we can split the error in two parts

$$\|v - I_{1,2} v\|_H \leq \|v - I_C v\|_H + \|I_C v - I_{1,2} v\|_H.$$

The first part can be estimated by $\|v - I_C v\|_H \leq c h$ for a constant $c$, see [1]. The second part can be proved directly. For that see the notation in Figure 2.3. By elementary calculus we get

$$\|I_C v - I_1 v\|_H \leq \left( m_a^2 + (m_a - m_1)^2 \right)^{\frac{1}{2}} h^{\frac{1}{2}},$$

$$\|I_C v - I_2 v\|_H \leq \left( m_b^2 + m_2^2 \right)^{\frac{1}{2}} h^{\frac{1}{2}},$$

where the $m$'s are the slopes of the corresponding linear functions. For the last statement to be true we have to show $(m_a - m_1)^2 = m_2^2$, which can easily be computed.

**Remark 5.** The fact that we can show $c_1 = c_2$ is only valid in the 1d case as the situation is fully symmetric around the border $\partial \Pi$. More importantly, this result shows that one can only expect a convergence order of $h^{\frac{1}{2}}$ and that this bound appears locally near the border $\partial \Pi$.

In the remainder of this section we use $X_c = X_{c,2}$. With Lemma 2 the discretized version of (2.17) can be written as

Solve

$$P^T A \tilde{u} = P^T \tilde{f},$$  

(2.27)

and set $\bar{u}_c = P \tilde{u}$.
where $A$ is the stiffness matrix in $X$ as defined before. In the real application, we use an iterative solver to solve this system. Therefore it is not necessary and also not recommended to explicitly construct the matrix $P^TAP$. The iterative solver just has to know the action of the matrix $P^TAP$ on a vector. In the next subsection we will see that $P$ is a very simple and sparse matrix, even not worth to explicitly store it.

### 2.2.4 Representation of $P$

The construction of the nodal operator $P$ is simple with the help of the former subsection. Figure 2.4 shows a triangulation of a part of the domain $\Omega_c$. The large circle is the boundary $\partial \Pi$, while the domain $\Pi$ is inside the circle and $\Omega$ is outside. The nodes inside $\Pi$ are colored black, while the nodes outside are colored gray. Suppose the nodes are numbered according to their color: first the gray nodes in $\Omega$, then the black nodes inside $\Pi$. With this numbering and from the general formula $P^Iv = IPv$ and the given expressions $I$ and $P$ we conclude that an element $\vec{v} \in \mathbb{R}^{nd} = \mathcal{I}(X)$ has the form

\[
[v_*, \ldots, v_*, v_*, \ldots, v_*, V]^T,
\]

(2.28)

while an element $\vec{v} \in \mathbb{R}^{nd} = \mathcal{I}(X_c)$ has the form

\[
[v_*, \ldots, v_*, V, \ldots , V, V]^T.
\]

(2.29)
The function of the projection operator $P$ is to transform $P\vec{v} = \vec{v}$. Thus, the explicit form of $P$ is

$$
P\vec{v} = \begin{bmatrix} 1 \cdots 1 \\
0 \cdots 0 \\
\vdots & \vdots \\
0 & 1 \\
1 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\
v_2 \\
\vdots \\
v_n \\
V \end{bmatrix} = \begin{bmatrix} v_1 \\
v_2 \\
\vdots \\
v_n \\
V \end{bmatrix} = \vec{v}.
$$

(2.30)

All degrees of freedom (dofs) located on the gray nodes are left unchanged, while the dofs on the black nodes inside $\Pi$ are set to $V$. This operation is so simple that it is not worth to explicitly construct the matrix $P$. Instead, in our code we use subroutines to compute the action of $P$ and $P^T$ on a vector, see Program listing 1.

```
subroutine pmul(vec_in , vec_out)
    do i = 1, nk
        if( color(i) == BLACK ) then
            vec_out(i) = vec_in(nd)
        else
            vec_out(i) = vec_in(i)
        end if
    end do
    vec_out(nd) = vec_in(nd)
end
```

Program listing 1: Subroutine to perform the action of $P$.

### 2.2.5 Error Estimate for the Model Problem in $X_c$

**Theorem 6.** Let $(u_h, U_h) = (\sum_i u_i \phi_i, U_h)$ be the solution of (2.27) and $(u, U)$ the solution of (2.16a). Suppose $u \in H^{2,2}(\Omega)$, $u \in H^{2,2}(\Pi)$ and $u \in H^{1,\infty}(\Omega_c)$, then
there exist constants $c_1$ and $c_2$ such that

$$
\|(u, U) - (u_h, U_h)\|_H \leq c_1 h + c_2 h^{1/2},
$$

(2.31)

with mesh width $h$. Additionally, the second term originates from a small band near the boundary $\partial \Pi$.

**Proof.** In Section 3.3 the proof is given for a more general situation. \qed

### 2.2.6 Example

To support the sharpness of the error estimate in the previous section, we present a numerical example. For the data

$$
f(x) = 1, \ F = 3/16, \ \Omega = B_1(0) \text{ and } \Pi = B_{1/2}(0) \subset \mathbb{R}^2
$$

(2.32)

the exact solution of the model problem (2.10a)–(2.10d) is

$$
u = (1 - x^2 - y^2)/4 \text{ and } U = 3/16.
$$

(2.33)

A plot of the exact solution can be seen in figure 2.5. The results of the calculations

![Figure 2.5: Plot of the exact solution of (2.10a)–(2.10d) with the given data.](image)

are printed in Table 2.1 and Table 2.2. In the first table, regular refinement is applied on each level. As a consequence the mesh width $h$ is reduced by a factor of 2 from one level to the next one. Clearly the error $\|(u, U) - (u_h, U_h)\|_H$ is reduced by a factor of
two if the level is increased twice, that means proportionality to $\sqrt{h}$. In the second table, additional local refinement at the boundary $\partial \Pi$ is applied to the mesh. At the boundary $\partial \Pi$, the mesh width is reduced by a factor of four, elsewhere it is reduced by a factor of two. The error estimates predict, that the convergence rate of $\sqrt{h}$ is caused from a small area near the boundary $\partial \Pi$. If this is true, then a reduction by a factor of two of the error should be observed on each level. The numbers of the error $\|(u, U) - (u_h, U_h)\|_H$ support this statement. In addition, a comparison between the two tables show that local refinement is a good idea anyhow. The rows printed with a bold font have nearly the same amount of dofs. In the locally refined version the error is much smaller. Figures 2.6 and 2.7 exemplarily show the results of such calculations.

<table>
<thead>
<tr>
<th>Level</th>
<th>dofs</th>
<th>$|\nabla(u - u_h)|_{L^2(\Omega_h)}$</th>
<th>$|(u, U) - (u_h, U_h)|_H$</th>
<th>EOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2097</td>
<td>4.17e-02</td>
<td>4.18e-02</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>8257</td>
<td>2.97e-02</td>
<td>2.98e-02</td>
<td>0.49</td>
</tr>
<tr>
<td>3</td>
<td>32769</td>
<td>2.08e-02</td>
<td>2.08e-02</td>
<td>0.52</td>
</tr>
<tr>
<td>4</td>
<td>130561</td>
<td>1.47e-02</td>
<td>1.47e-02</td>
<td>0.50</td>
</tr>
<tr>
<td>5</td>
<td>521217</td>
<td>1.05e-02</td>
<td>1.05e-02</td>
<td>0.49</td>
</tr>
</tbody>
</table>

Table 2.1: Errors of the numerical scheme with regular grid refinement applied to each level.

<table>
<thead>
<tr>
<th>Level</th>
<th>dofs</th>
<th>$|\nabla(u - u_h)|_{L^2(\Omega_h)}$</th>
<th>$|(u, U) - (u_h, U_h)|_H$</th>
<th>EOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>997</td>
<td>4.18e-02</td>
<td>4.19e-02</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>5809</td>
<td>2.08e-02</td>
<td>2.08e-02</td>
<td>1.01</td>
</tr>
<tr>
<td>3</td>
<td>30593</td>
<td>1.05e-02</td>
<td>1.05e-02</td>
<td>0.99</td>
</tr>
<tr>
<td>4</td>
<td>151889</td>
<td>5.23e-03</td>
<td>5.23e-03</td>
<td>1.01</td>
</tr>
<tr>
<td>5</td>
<td>726133</td>
<td>2.60e-03</td>
<td>2.60e-03</td>
<td>1.01</td>
</tr>
</tbody>
</table>

Table 2.2: Errors of the numerical scheme with regular grid refinement and additional local refinement at the boundary $\partial \Pi$ applied to each level.
Figure 2.6: Numerical solution of the model problem (2.10a)–(2.10d) on a grid with nearly constant mesh width.

Figure 2.7: Numerical solutions of the model problem (2.10a)–(2.10d) on a grid with local refinement at the boundary $\partial \Pi$. 
Chapter 3

Particulate Flows (PF)

3.1 Mathematical Formulation

3.1.1 Model

In this section we introduce the mathematical model for particulate flows. For the ease of presentation, this will be done for the 3d case with only one particle. The extension to more particles is straightforward, simply by adding an index. The model also holds for the 2d case, one just has to adapt the definition of the cross-product involved in the equations.

Denote by $\Omega(t) \subset \mathbb{R}^3$ the time–dependent domain occupied by an incompressible, Newtonian fluid with velocity $u$ and pressure $p$. Its motion is described by the incompressible Navier–Stokes–equation. A homogeneous no–slip condition is prescribed on the outer boundary $\Gamma_D$. $\Pi(t) \subset \mathbb{R}^3$ is the time–dependent domain of a rigid particle, with its center of mass given by $X = \frac{1}{|\Pi(t)|} \int_{\Pi(t)} x \, dx$ while $r = x - X$ is its relative coordinate. The particle’s motion, being a rigid body motion, is governed

![Diagram](https://via.placeholder.com/150)

Figure 3.1: Particle $\Pi(t)$ of arbitrary shape inside the fluid domain $\Omega(t)$.
by Newton’s law, describing values for the translational and angular velocities \( U, \omega \), respectively, and the position \( X \). The orientation in space is given by a complete system of orthogonal unit vectors \( \Theta \). Since the particle is impermeable, we assume \( \Omega(t) \cap \Pi(t) = \emptyset \) for all times \( t > 0 \). Finally, we assume (for simplicity) that the whole volume \( \Omega_c = \Omega(t) \cup \Pi(t) \cup \partial \Pi(t) \) is time–independent. See also Fig. 3.1.1 for a sketch of the situation.

The motion of fluid and particle is coupled on one hand by the no–slip condition on the particle boundary, see (3.1d), and on the other hand by the stress and pressure forces of the fluid acting on the particle in the right hand sides of (3.1e) and (3.1g). The mathematical model consists of a coupled system of partial differential equations (PDE) for \( u, p \) and of ordinary differential equations (ODE) for \( U, \omega, X \) and \( \Theta \) reading in dimensional form

\[
\begin{align*}
\rho \partial_t u + \rho (u \cdot \nabla) u - \nabla \cdot (\eta D[u] - p I) &= 0 & & \text{in } \Omega(t), \quad (3.1a) \\
\nabla \cdot u &= 0 & & \text{in } \Omega(t), \quad (3.1b) \\
\n\mathbf{u} &= 0 & & \text{on } \Gamma_D, \quad (3.1c) \\
\mathbf{u} &= \mathbf{U} + \omega \times \mathbf{r} & & \text{on } \partial \Pi(t), \quad (3.1d) \\
\frac{d}{dt} (MU) &= M \frac{dU}{dt} = \mathbf{F} - \int_{\partial \Pi(t)} \sigma \mathbf{n} ds, \quad (3.1e) \\
\frac{d\mathbf{X}}{dt} &= \mathbf{U}, \quad (3.1f) \\
\frac{d}{dt} (I\omega) &= I \frac{d\omega}{dt} + \omega \times (I\omega) = - \int_{\partial \Pi(t)} \mathbf{r} \times \sigma \mathbf{n} ds, \quad (3.1g) \\
\frac{d\Theta}{dt} &= R[\omega] \Theta. \quad (3.1h)
\end{align*}
\]

The system has to be closed by appropriate initial conditions. Here, \( \rho \) is the constant density and \( \eta \) the dynamic viscosity of the fluid; \( \sigma \) is the stress tensor defined by \( \sigma := \eta D[u] - p I \), where \( D[\cdot] \) is the deformation tensor \( D[u]_{i,j} = \partial_j u_i + \partial_i u_j \) and \( I_{i,j} = \delta_{ij} \) the identity.

We assume the particle’s density \( \rho_\Pi \) to be constant, then \( M = \rho_\Pi \int_{\Pi(t)} 1 d\mathbf{x} \) and \( I = \rho_\Pi \int_{\Pi(t)} \mathbf{r} \cdot \mathbf{r} - \mathbf{r} \otimes \mathbf{r} d\mathbf{x} \) are the respective mass and inertia tensor of the particle where \( \otimes \) is the usual outer product defined by \( (\mathbf{r} \otimes \mathbf{r})_{i,j} = r_i r_j \). \( \mathbf{n} \) is the outer normal of the domain \( \Omega(t) \). Note that it is opposite to the particle’s outer normal on the particle boundary \( \partial \Pi(t) \). This is the reason for the minus sign in the right hand sides of equations (3.1e) and (3.1g). \( \mathbf{F} \) describes an external force acting on the particle like gravity, particle–particle (in case of more than 1 particle) or particle–wall
interaction. \( R[\cdot] \) is the cross-product operator defined by

\[
R[\omega] = \begin{bmatrix}
0 & -\omega_3 & \omega_2 \\
\omega_3 & 0 & -\omega_1 \\
-\omega_2 & \omega_1 & 0
\end{bmatrix}.
\] (3.2)

For a characteristic length \( x_c \) and velocity \( u_c \), we introduce the dimensionless variables

\[
\tilde{x} = \frac{x}{x_c}, \quad \tilde{u} = \frac{u}{u_c}, \quad \tilde{t} = \frac{tu_c}{x_c}, \quad \tilde{p} = \frac{p}{\rho u_c^2}, \quad \tilde{U} = \frac{U}{u_c} \quad \text{and} \quad \tilde{\omega} = \frac{\omega x_c}{u_c}.
\]

Mass, inertia and force are also scaled by

\[
\tilde{M} = \frac{M}{\rho x_c^3}, \quad \tilde{I} = \frac{I}{\rho x_c^5} \quad \text{and} \quad \tilde{F} = \frac{1}{\rho u_c^2 x_c^2} F.
\]

Finally, define the Reynolds number by

\[
Re = \frac{\rho x_c u_c}{\eta}.
\]

Inserting this into the previous set of equations and skipping the tildes results in a dimensionless model for particulate flows, where the model parameters are the Reynolds number \( Re \), the density ratio \( \alpha = \frac{\rho}{\rho_{\Pi}} \) (in \( M \) and \( I \)) and the external force \( F \). “\( \dot{\cdot} \)” is used as abbreviation for \( \frac{d}{dt} \). We arrive at the dimensionless formulation:

\[
\begin{align*}
\partial_t u + (u \cdot \nabla) u - \nabla \cdot \left( \frac{1}{Re} D[u] - p \right) &= 0 \quad \text{in } \Omega(t), \quad (3.3a) \\
\nabla \cdot u &= 0 \quad \text{in } \Omega(t), \quad (3.3b) \\
\nabla \cdot v &= 0 \quad \text{on } \Gamma_D, \quad (3.3c) \\
\n\nabla \cdot v &= 0 \quad \text{on } \partial \Pi(t), \quad (3.3d) \\
M \dot{U} &= F - \int_{\partial \Pi(t)} \sigma \, n \, ds, \quad (3.3e) \\
\dot{X} &= U, \quad (3.3f) \\
I \dot{\omega} + \omega \times (I \omega) &= -\int_{\partial \Pi(t)} r \times \sigma \, n \, ds, \quad (3.3g) \\
\dot{\Theta} &= R[\omega] \Theta. \quad (3.3h)
\end{align*}
\]

### 3.1.2 Weak Formulation

In this section, the weak formulation of system (3.3a)–(3.3h) is derived. We follow the idea and presentation in [31]. This formulation perfectly fits to the context of the Subspace Projection Method presented in the previous chapter. We start with the definition of the function space of combined velocities:

\[
H_c(\Omega(t)) = \left\{ (v, V, \xi) \mid v \in (H^1(\Omega(t)))^3, \ V \in \mathbb{R}^3, \ \xi \in \mathbb{R}^3, \ v = 0 \text{ on } \Gamma_D, \ v = V + \xi \times r \text{ on } \partial \Pi(t) \right\}.
\] (3.4)

This space is called the space of combined velocities, since it combines the fluid velocity \( u \), the translational particle velocity \( V \) and the angular particle velocity \( \xi \).
Elements in this space are typically denoted by \( u = (u, U, \omega) \) and \( v = (v, V, \xi) \). Sometimes the abbreviation \( v \in H_c(\Omega(t)) \) is used instead of the correct form \( v \in \{ w \mid (w, V, \xi) \in H_c(\Omega) \} \). We also define an inner product

\[
(u, v) := \int_{\Omega_c} u \cdot v \, dx + U \cdot V + \omega \cdot \xi. \tag{3.5}
\]

Note that the space \( H_c \) equipped with this inner product is not a Hilbert space as it is not complete. In order to derive the weak formulation, we perform the symbolic calculation

\[
\int_{\Omega(t)} (3.3a) \cdot v \, dx + (3.3e) \cdot V + (3.3g) \cdot \xi \tag{3.6}
\]

for a test function \( (v, V, \xi) \in H_c(\Omega(t)) \), yielding

\[
\begin{align*}
&\int_{\Omega(t)} \partial_t u \cdot v + u \cdot \nabla u \cdot v \, dx + \int_{\Omega(t)} \nabla p \cdot v - \frac{1}{\text{Re}} \nabla \cdot D[u] \cdot v \, dx \\
&\quad + M \dot{U} \cdot V + I \dot{\omega} \cdot \xi + \omega \times (I \omega) \cdot \xi = \\
&\quad F \cdot V - \int_{\partial \Pi(t)} \sigma \, n \cdot V + [r \times \sigma \, n] \cdot \xi \, ds. \tag{3.7}
\end{align*}
\]

Integration by parts of the stress term and observing that \( v = V + \xi \times r \) on \( \partial \Pi(t) \) leads to

\[
- \int_{\Omega(t)} \frac{1}{\text{Re}} \nabla \cdot D[u] \cdot v - \nabla p \cdot v \, dx =
\]

\[
\int_{\Omega(t)} \frac{1}{2\text{Re}} D[u] : D[v] - p \nabla \cdot v \, dx
\]

\[
- \int_{\partial \Pi(t)} \left( \frac{1}{\text{Re}} D[u] \cdot n - pn \right) \cdot (V + \xi \times r) \, ds. \tag{3.8}
\]

Using the relation \( a \cdot (b \times c) = (c \times a) \cdot b \) for the boundary term in (3.8), we get

\[
- \int_{\partial \Pi(t)} \sigma \, n \cdot (V + \xi \times r) \, ds = - \int_{\partial \Pi(t)} \sigma \, n \cdot V + [r \times \sigma \, n] \cdot \xi \, ds,
\]

which is exactly the last term in (3.7), and hence, both terms cancel out. Thus we arrive at:
For almost all \( t > 0 \), \( q \in L^2(\Omega(t)) \) and \( v \in H_c(\Omega(t)) \) find \( p \) and \( u = (u, U, \omega) \) such that

\[
\int_{\Omega(t)} \partial_t u \cdot v + u \cdot \nabla u \cdot v \, dx + \int_{\Omega(t)} \frac{1}{2\text{Re}} \text{D}[u] : \text{D}[v] - p \nabla \cdot v \, dx \\
+ M \dot{U} \cdot V + I \dot{\omega} \cdot \xi + \omega \times (I \omega) \cdot \xi = F \cdot V \\
\int_{\Omega(t)} \nabla \cdot u q \, dx = 0, \\
\dot{X} = U, \\
\dot{\Theta} = R[\omega] \Theta.
\]

(3.9a) (3.9b) (3.9c) (3.9d)

The forces at the interface are incorporated in the weak formulation, and there is no need for an explicit computation of the stresses. A second important point is the appearance of all three velocities within one equation. These two points will enable us to derive an efficient solution strategy in Section 3.2. The next step is to extend the weak formulation on the time independent domain \( \Omega_c \), yielding the one domain approach. To this end, we naturally extend the velocity \( v \) to \( \Pi(t) \) by prescribing

\[
v = V + \xi \times r \text{ in } \Pi(t).
\]

(3.10)

This gives rise to the following definition:

\[
H_c(\Omega_c) = \left\{ (v, V, \xi) \mid v \in (H^1(\Omega_c))^3, V \in \mathbb{R}^3, \xi \in \mathbb{R}^3,
\begin{align*}
&v = 0 \text{ on } \Gamma_D, \\
&v = V + \xi \times r \text{ in } \Pi(t)
\end{align*}
\}. 
\]

(3.11)

The velocity \( v \) in \( H_c(\Omega_c) \) is defined on the combined domain \( \Omega_c \) and is restricted to the rigid body velocity \( V + \xi \times r \) inside the particle. This space is also used in proofs for the existence and uniqueness of single particle flows, see for example [55, 29].

Equation (3.10) implies

\[
\nabla \cdot v = 0 \quad \text{and} \quad \text{D}[v] = 0 \quad \text{in } \Pi(t).
\]

(3.12)

Then for \( u \in H_c(\Omega_c) \)

\[
\int_{\Pi(t)} \nabla \cdot u q \, dx = 0 \quad \text{for all } q \in L^2(\Pi(t))
\]

(3.13)

as well as

\[
\int_{\Pi(t)} \frac{1}{2\text{Re}} \text{D}[u] : \text{D}[v] - p \nabla \cdot v \, dx = 0 \quad \text{for all } v \in H_c(\Omega_c).
\]

(3.14)
For the total time derivative of the velocity, we compute
\[
\frac{d}{dt} u = \partial_t u + (u \cdot \nabla) u = \dot{U} + \dot{\omega} \times r + \omega \times (\omega \times r) \quad \text{in } \Pi(t). \quad (3.15)
\]
Multiplying Eq. (3.15) by \( v \), \( v_{\Pi(t)} = V + \xi \times r \) and integrating over \( \Pi(t) \) lead to (for a detailed step by step calculation see Appendix A)
\[
\int_{\Pi(t)} (\partial_t u + (u \cdot \nabla) u) \cdot v \, dx =
\int_{\Pi(t)} \dot{U} \cdot (V + \xi \times r) \, dx + \int_{\Pi(t)} (\dot{\omega} \times r + \omega \times (\omega \times r)) \cdot (V + \xi \times r) \, dx =
\frac{\alpha}{\alpha} \int_{\Pi(t)} \dot{U} \cdot V \, dx + \frac{\alpha}{\alpha} \int_{\Pi(t)} (\dot{\omega} \times r + \omega \times (\omega \times r)) \cdot (\dot{\xi} \times r) \, dx =
\alpha M \dot{U} \cdot V + \alpha (I \dot{\omega} + \omega \times (I \omega)) \cdot \xi. \quad (3.16)
\]
By adding (3.14) and (3.16) to (3.9a) and also (3.13) to (3.9b), the weak formulation on the combined domain finally reads

\[
\text{Find } (u, p) \text{ with } u(t) \in H_c(\Omega_c), p(t) \in L^2_0(\Omega_c) \text{ such that for all } (v, q) \in (H_c(\Omega_c) \times L^2_0(\Omega_c))
\]
\[
\int_{\Omega_c} \partial_t u \cdot v + (u \cdot \nabla) u \cdot v + \frac{1}{2Re} D[u] : D[v] - p \nabla \cdot v \, dx
+ (1 - \alpha) M \dot{U} \cdot V + (1 - \alpha) (I \dot{\omega} + \omega \times (I \omega)) \cdot \xi = F \cdot V, \quad (3.17a)
\]
\[
\int_{\Omega_c} \nabla \cdot u \, q \, dx = 0, \quad (3.17b)
\]
\[
\dot{X} = U, \quad (3.17c)
\]
\[
\dot{\Theta} = R[\omega] \Theta. \quad (3.17d)
\]
Note that in the above formulation we did not specify the precise parabolic spaces. For a shorter notation, we introduce the bi– and trilinear forms
\[
m(u, v) = \int_{\Omega_c} u \cdot v \, dx, \quad (3.18a)
\]
\[
s(u, v) = \frac{1}{2Re} \int_{\Omega_c} D[u] : D[v] \, dx, \quad (3.18b)
\]
\[
k(w; u, v) = \int_{\Omega_c} (w \cdot \nabla) u \cdot v \, dx, \quad (3.18c)
\]
\[
b(q, v) = \int_{\Omega_c} q \nabla \cdot v \, dx, \quad (3.18d)
\]
and the variable \( \beta = 1 - \alpha \). Then, (3.17a)–(3.17d) can be compactly written as
Find \((u, p)\) with \(u(t) \in H_c(\Omega_c), \ p(t) \in L^2_0(\Omega_c)\) such that for all \((v, q) \in (H_c(\Omega_c) \times L^2_0(\Omega_c))\)

\[
m(\dot{u}, v) + k(u; v, u) + s(u, v) - b(p, v) + \\
\beta M \dot{U} \cdot V + \beta (I \dot{\omega} + \omega \times (I \omega)) \cdot \xi = F \cdot V,
\]

\[
b(q, u) = 0,
\]

\[
\dot{X} = U,
\]

\[
\hat{\Theta} = R[\omega] \Theta.
\]

Eqs. (3.19a) and (3.19b) are called the combined Navier–Stokes equations. The time dependence of \(\Omega(t)\) and \(\Pi(t)\) is completely hidden in the time–dependent definition of \(H_c(\Omega_c)\).

### 3.2 Numerical Method

The numerical scheme to solve the weak problem (3.19a)–(3.19d) derived in the previous section consists of the following six points:

1. splitting scheme to decouple the unknowns;
2. a pressure correction projection scheme based on a BDF2 method to efficiently solve the combined Navier–Stokes equations;
3. subspace projection to incorporate the restrictions given by the function space \(H_c(\Omega_c)\);
4. adaptivity in space;
5. preconditioning;

#### 3.2.1 Splitting by Time Discretization

We start by introducing a splitting scheme to solve the highly coupled, highly non-linear system of equations. The presented algorithm decouples the position of the particles \((X, \Theta)\) from the combined Navier–Stokes equations \((u, U, \omega, p)\). These are then further decoupled by a pressure correction projection method. Thus, the philosophy here is to finally split the complex system into a cascade of simple
subproblems rather than using a (maybe more accurate but much more expensive) monolithic approach.

In order to discretize in time, the time interval \((0, T)\) is subdivided by discrete time instants: \(0 = t^0 < t^1 < \cdots < t^N = T\). Denote by \(\tau_{k+1} := t^{k+1} - t^k\). For simplicity, a fixed time step size \(\tau\) is used: \(\tau_k = \tau\) for all \(k = 1, \ldots, N\). Moreover, define \(\gamma = \frac{2}{3} \tau\).

To begin time integration we need initial values \(X^0, \Theta^0, u^0, U^0\) and \(\omega^0\) with \(\nabla \cdot u^0 = 0\). Furthermore, to start the BDF2 method we additionally need values for \(u^{-1}, p^0\), and we set the auxiliary variable \(\chi^0 = \chi^{-1} = 0\).

Then, in each time step Eqs. (3.19a)–(3.19d) are split into three substeps. The first is a predictor step for the preliminary particle position and translational velocity, \(\bar{X}\) and \(\bar{U}\), respectively. In the second step values for \(u^{k+1}, U^{k+1}, \omega^{k+1}\) and \(p^{k+1}\) are computed based on the BDF2 projection scheme in rotational form. The last step is a corrector for \(X^{k+1}\) and \(\Theta^{k+1}\).

The predictor step is done by the implicit midpoint rule which is of second order in time [21] and preserves quadratic first integrals, i.e. energy.

\[
\begin{align*}
\text{Predictor} & \quad \text{Given } F, X^k \text{ and } U^k. \\
\bar{X} &= X^k + \frac{\tau}{2} \left( U^k + \bar{U} \right), & (3.20a) \\
\bar{U} &= U^k + \frac{\tau M}{2} F \left( X^k + \bar{X} \right). & (3.20b)
\end{align*}
\]

The combined Navier Stokes equations are discretized by a projection method in rotational form, see [33]. The time derivative \(\partial_t u\) is replaced by a BDF2 scheme having good stability properties while the equations for \(\bar{U}\) and \(\bar{\omega}\) are approximated by the implicit midpoint rule, respectively. The particle domain \(\Pi(t)\) is described by its current position \(\bar{X}\) and \(\Theta^k\):

**Combined Navier Stokes**

**Step 1** (Burgers problem)
Choose one of the options for \( u^* \) and \( \omega^* \).

\[
\begin{align*}
  u^* &= \begin{cases} 
    u^k & \text{for} \\
    2u^k - u^{k-1} & \text{for} \\
    u^{k+1} & \text{for}
  \end{cases}, \\
  \omega^* &= \begin{cases} 
    \omega^k & \text{for} \\
    2\omega^k - \omega^{k-1} & \text{for} \\
    \omega^{k+1} & \text{for}
  \end{cases}.
\end{align*}
\]  
(3.21)

Given \( u^k, u^{k-1}, p^k, \chi^k, \chi^{k-1}, U, \omega^k \).

Find \( u^{k+1} \in H_c(\Omega_c) \) such that for all \( \mathbf{v} \in H_c(\Omega_c) \)

\[
\begin{align*}
  m(u^{k+1}, \mathbf{v}) + \gamma k(u^*; u^{k+1}, \mathbf{v}) + \gamma s(u^{k+1}, \mathbf{v}) + \\
  \frac{2}{3} \beta M u^{k+1} \cdot \mathbf{v} + \frac{2}{3} \beta I \omega^{k+1} \cdot \mathbf{v} + \frac{2}{3} \beta \omega^* \times \left( I \omega^{k+1} \right) \cdot \xi = \\
  \gamma b(p^k, \mathbf{v}) + m\left( \frac{4}{3} u^k - \frac{1}{3} u^{k-1}, \mathbf{v} \right) + \gamma b\left( \frac{4}{3} \chi^k - \frac{1}{3} \chi^{k-1}, \mathbf{v} \right) + \\
  \frac{2}{3} \beta M \dot{U} \cdot \mathbf{v} + \frac{2}{3} \beta \dot{I} \omega^k \cdot \xi - \frac{\gamma}{2} \beta \omega^* \times \left( I \omega^k \right) \cdot \xi.
\end{align*}
\]  
(3.22)

**Step 2** (Computation of pressure correction)

Find \( \chi^{k+1} \in H^1(\Omega_c) \) such that for all \( \Psi \in H^1(\Omega_c) \)

\[
m(\nabla \chi^{k+1}, \nabla \Psi) = \frac{1}{\gamma} b(\Psi, u^{k+1}).
\]  
(3.23)

**Step 3** (Pressure update in rotational form)

Find \( p^{k+1} \in L^2_0(\Omega_c) \) such that for all \( q \in L^2_0(\Omega_c) \)

\[
m(p^{k+1}, q) = m(p^k + \chi^{k+1}, q) - b(q, \frac{2}{\text{Re}} u^{k+1}).
\]  
(3.24)

The corrector also uses the implicit midpoint rule for time discretization.

**Corrector**

Given \( \Theta^k, X^k, \omega^k, \omega^{k+1}, U^k \) and \( U^{k+1} \).

\[
\begin{align*}
  \Theta^{k+1} &= \left( I - \frac{\tau}{2} R \left[ \omega^{k+1} \right] \right)^{-1} \left( I + \frac{\tau}{2} R \left[ \omega^k \right] \right) \Theta^k, \\
  X^{k+1} &= X^k + \frac{\tau}{2} \left( U^k + U^{k+1} \right).
\end{align*}
\]  
(3.25a, b)

Remarks 7.
i) In some applications the time scale of the particle dynamics (for instance because of a particular particle–particle interaction) is much smaller than the time scale of the particle–fluid interaction. In such a case, in the predictor step, one might divide the time step \( \tau \) into smaller time–sub–steps \( \tau_s = \tau/s \) and repeat Eqs. (3.20a)-(3.20b) for the smaller time step \( s \) times.

ii) In fact, the projection scheme solves for the nondivergence free quantities denoted by \( \tilde{u}^{k+1} \) in Section 3.5 in [33], since the corrected solenoidal velocity is not a finite element function anymore in general. Moreover, as pointed out in [33], \( \tilde{u} \) is of the same order as \( u \).

iii) Depending on the choice in (3.21), Eq. (3.22) is linear or nonlinear. Furthermore, the second and third options give rise to a formally second order in time approximation of the nonlinearity. We made good experience with linearization by extrapolation, which gives almost the same results as the implicit equation. Thus, in what follows, we fix \( u^* := 2u^k - u^{k-1} \) and \( \omega^* := 2\omega^k - \omega^{k-1} \).

iv) A critical point in discretizing (3.19a) is the compatibility of the respective time discretizations. To see this, assume \( F \) to be linear, and let us write

\[
\begin{align*}
\text{BDF2} := & m(u^{k+1}, v) + \gamma k(u^*; u^{k+1}, v) + \gamma s(u^{k+1}, v) \\
& + \frac{2}{3} \beta I \omega^{k+1} \cdot \xi + \frac{2}{3} \beta \omega^* \times (I \omega^{k+1}) \cdot \xi \\
& - \gamma b(p^k, v) - m\left(\frac{4}{3} u^k - \frac{1}{3} u^{k-1}, v\right) - \gamma b\left(\frac{4}{3} \chi^k - \frac{1}{3} \chi^{k-1}, v\right) \\
& - \frac{2}{3} \beta I \omega^k \cdot \xi + \frac{2}{3} \beta \omega^k \times (I \omega^k) \cdot \xi
\end{align*}
\]

as abbreviation. Now insert Equation (3.20b) into (3.22):

\[
0 = \text{BDF2} + \frac{2}{3} \beta M U^{k+1} \cdot V - \frac{2}{3} \beta M \bar{U} \cdot V =
\]

\[
\text{BDF2} + \frac{2}{3} \beta M U^{k+1} \cdot V - \frac{2}{3} \beta M \left( F \left( \frac{X^k + \bar{X}}{2} \right) \right) \cdot V =
\]

\[
\text{BDF2} + \frac{2}{3} \beta M U^{k+1} \cdot V - \frac{2}{3} \beta M U^k \cdot V - \frac{\tau}{3} \left( F^k + F^{k+1} \right) \cdot V. \quad (3.26)
\]

The last line is exactly the equation we would have received if we had discretized Eq. (3.19a) in time without the decoupling. This property does not hold for arbitrary combinations of time discretization scheme. If, one uses for example, the BDF2 method in Eq. (3.22) for \( U \) instead of the implicit midpoint rule, then this leads to

\[
\text{BDF2} + \frac{2}{3} \beta M U^{k+1} \cdot V - \frac{2}{3} \beta M U^k \cdot V - \frac{2\tau}{3} \left( F^k + F^{k+1} \right) \cdot V. \quad (3.27)
\]

Clearly, here the force term is weighted by the wrong factor of 2.
v) The corrector step takes care that the particle position is also influenced by the fluid friction, which stabilizes the overall algorithm. It can be computed directly, one has only to invert a regular $3 \times 3$ matrix for each particle. The predictor is a fully implicit system of dimension six times number of particles and solved by Newton’s method including GMRes for inverting the Jacobian. A more detailed description will follow in Subsection 3.6.1.

vi) With some further consideration also the case $\beta = 0 \ (\rho = \rho_\Pi)$ can be handled without greater difficulties. For gravitational forces only, one can derive a version of the predictor without the appearance of $\beta$. If other forces are used the splitting is not valid anymore and we suggest to evaluate the forces directly in Eq. (3.19a), similar to Eq. (3.26) in Remark (iv).

3.2.2 Spatial Discretization

The decoupled time discretized model includes the PDEs (3.22)–(3.23) and one $L^2$–projection (3.24). The key point in the spatial discretization is to define a discrete counterpart of $H_c(\Omega_c)$ and, moreover, the concrete realization of this nonstandard finite element space.

Finite Element Spaces

To this end, fix a time instant $t^{k+1}$. The location of the particle after time discretization is defined by $\Pi^{k+1} := \bigcup \{ T \in T^{k+1} \mid T \subseteq \bar{\Pi} \}$. Where $T^{k+1}$ denotes a conforming triangulation of $\Omega_c$ with typical mesh width $h$ after the predictor step, and $\bar{\Pi}$ is the particle domain based on $\bar{X}$ and $\Theta^k$. $K^p$ is the space of polynomials of degree $p$.

Define the following finite element spaces

\[ \tilde{X} := \left\{ v \in (C^0(\bar{\Omega}_c))^3 \mid v|_T \in (K^2(T))^3 \quad \text{for all} \quad T \in \mathcal{T}, \ v = 0 \ \text{on} \ \Gamma_D \right\}, \]

\[ X := \left\{ v = (v, V, \xi) \mid v \in \tilde{X}, \ V \in \mathbb{R}^3, \ \xi \in \mathbb{R}^3, \ v = 0 \ \text{on} \ \Gamma_D \right\}, \]

\[ X_c := \left\{ v_c = (v_c, V, \xi) \in X \mid v_c|_T = V + \xi \times r \ \text{for all} \quad T \in \mathcal{T} \quad \text{with} \quad T \subseteq \bar{\Pi} \right\}, \]

\[ Y := \left\{ v \in C^0(\bar{\Omega}_c) \mid v \in K^1(T) \quad \text{for all} \quad T \in \mathcal{T} \right\}. \]

$\tilde{X}$ is the Lagrange finite element space of globally continuous, piecewise quadratics with homogeneous Dirichlet boundary conditions. These functions are augmented by six additional degrees of freedom in $X$. Three of them for the particle velocity $V$ and three for the particle angular velocity $\xi$. In the case of more than one particle, the
space $X$ has to be extended by additionally six dofs for each extra particle. The pair $\bar{X} \times Y$ is the LBB–stable Taylor–Hood element to approximate the Navier–Stokes equations.

$X_c$ is an approximation of $H_c$ as it includes the rigid body constraint (3.1d) for all elements that completely lie within $\bar{\Pi}$, see Fig 3.2. Computationally, it is not straightforward how to directly work with the space $X_c$. Instead, we will work with $X$ and circumvent the explicit representation of $X_c$ by using a subspace projection as presented in Chapter 2.

$$\mathcal{P} : X \to X_c.$$ (3.28)

With the notation

$$a(u, v) = a((u, U, \omega), (v, V, \xi)) := m(u, v) + \gamma k(u^\ast; u, v) + \gamma s(u, v) + \frac{2}{3} \beta M U \cdot V + \frac{2}{3} \beta I \omega \cdot \xi + \frac{\gamma}{2} \beta \omega^\ast \times (I \omega) \cdot \xi,$$ (3.29)

and

$$l(v) = l((v, V, \xi)) := \gamma b(p^k, v) + m\left(\frac{4}{3} u^k - \frac{1}{3} u^{k-1}, v\right) + \gamma b\left(\frac{4}{3} \chi^k - \frac{1}{3} \chi^{k-1}, v\right) + \frac{2}{3} \beta M \dot{U} \cdot V + \frac{2}{3} \beta I \omega^k \cdot \xi - \frac{\gamma}{2} \beta \omega^k \times (I \omega^k) \cdot \xi,$$ (3.30)

we arrive at the finite element discretization of Eqs. (3.22) – (3.24):
Discretized Combined Navier Stokes

Step 1 (Burgers problem)
Given \( \mathbf{u}^k, \mathbf{u}^{k-1}, p^k, \chi^k, \chi^{k-1}, \bar{U}, \omega^k \).

Find \( \tilde{\mathbf{u}}^{k+1} \in X \) such that for all \( \tilde{\mathbf{v}} \in X \)
\[
a(\mathcal{P}\tilde{\mathbf{u}}^{k+1}, \mathcal{P}\tilde{\mathbf{v}}) = (\tilde{f}, \mathcal{P}\tilde{\mathbf{v}}),
\]
(3.31)
and set \( (\mathbf{u}^{k+1}, \mathbf{U}^{k+1}, \omega^{k+1}) = \mathbf{u}^{k+1} = \mathcal{P}\tilde{\mathbf{u}}^{k+1} \). Hereby we set \( \tilde{f} = (f, F, T) \) such that
\[
l(\mathbf{v}) = (f, \mathbf{v}) \quad \text{for all} \ \mathbf{v} \in X_c.
\]
(3.32)

Step 2 (Computation of the pressure correction)

Find \( \chi^{k+1} \in Y \) such that for all \( \Psi \in Y \)
\[
m(\nabla \chi^{k+1}, \nabla \Psi) = \frac{1}{\gamma} b(\Psi, \mathbf{u}^{k+1}).
\]
(3.33)

Step 3 (Pressure update in rotational form)

Find \( p^{k+1} \in Y \) such that for all \( q \in Y \)
\[
m(p^{k+1}, q) = m(p^k + \chi^{k+1}, q) - b(q, \frac{2}{\text{Re}} \mathbf{u}^{k+1}).
\]
(3.34)

Eqs. (3.33), (3.34) can be solved in a standard way. It remains to give a formulation of (3.31) in terms of nodal values. By using the Subspace Projection Method, Lemma 2, we arrive at
\[
P^T A \bar{\mathbf{u}}_{k+1} = P^T \bar{f},
\]
(3.35)
where \( \bar{\mathbf{u}}_{k+1} = \mathcal{P}\tilde{\mathbf{u}}^{k+1} \).

Now, let \( \phi_i, i = 1, \ldots, 3n k \) be the Lagrange nodal basis of \( \tilde{X} \) with \( n k \) the number of inner Lagrange nodes. We extend this basis to a basis \( \phi_i, i = 1, \ldots, n d = n k + 6 \) of \( X \) by setting
\[
\{\phi_1, \ldots, \phi_{n d}\} = \{(\phi_1, 0)^T, \ldots, (\phi_{n k}, 0)^T, (0, \psi_1)^T, \ldots, (0, \psi_6)^T\}
\]
(3.36)
for appropriate vectors $\psi_1, \ldots, \psi_6$. There is a canonical isomorphism

$$\mathcal{I} : X \to \mathbb{R}^{nd}$$

mapping finite element functions in $X$ to the vector of nodal coefficients. The coefficients $u_i$ are well defined by the unique decomposition $u = \sum u_i \phi_i$. The system matrix $A \in \mathbb{R}^{nd \times nd}$ is then given by

$$A_{i,j} = a(\phi_j, \phi_i) \quad i, j = 1, \ldots, nd$$

and likewise the right hand side $\vec{f} = (f_1, \ldots, f_{nd})^T$, $f_i = (j, \phi_i)$.

**Construction of $P$**

It is worth to mention again that it is not necessary and also not recommended to explicitly construct the matrix $P^TAP$. Using an iterative solver, only the action of the matrix-vector has to be known.

Like in the previous chapter Fig. 3.3 shows a triangulation of a part of the domain $\Omega_c$. The solid circle indicates the boundary $\partial \Pi$ with $\Pi$ lying inside the circle and $\Pi = \Omega_c \setminus (\Pi \cup \partial \Pi)$ outside. The nodes inside $\Pi$ are colored black while the nodes outside are colored gray. Suppose the nodes are numbered according to their color, i.e. the gray nodes in $\tilde{\Omega}$ first, then the black nodes inside $\Pi$. If $nf$ is the number of Lagrange nodes located outside the particle, and $np$ the number of nodes inside the particle, we have $nk = nf + np$. With this numbering an element $\vec{v} \in \mathbb{R}^{nd} = \mathcal{I}(X)$

![Figure 3.3: Triangulation of $\Omega_c$.](image)

has the form

$$[v_*, \ldots, v_*, v_*, \ldots, v_*, V, \xi]^T.$$  

(3.38)
Note that in this formula, all entries \( v \), \( V \) and \( \xi \) are vectors of dimension three. We define the projection \( P \) in the following way. \( (P\vec{v})_i = v_i \) for \( i = 1, \ldots, 3nf \) and also for the last six entries which represent the particle velocity and angular velocity, respectively. For the 3np basis function whose Lagrange nodes are inside \( \tilde{\Pi} \), the resulting value is the particle velocity. This velocity is the sum of the translational and the angular velocity and is given by \( \vec{v} = \vec{V} + \vec{\xi} \times \vec{r} \) where \( \vec{r} \) is the local particle coordinate. The nodal vector corresponding to the basis (3.36) is

\[
[v_1, \ldots, v_{3nf}, v_{3nf+1}, \ldots, v_{3(nf+np)=3nk}, V_1, V_2, V_3, \xi_1, \xi_2, \xi_3]^T \in \mathbb{R}^{nd}. \quad (3.39)
\]

The same vector transformed by \( P \) must be

\[
[v_1, \ldots, v_{3nf}, V_1 + \xi_2 r_3 - \xi_3 r_2, V_2 + \xi_3 r_1 - \xi_1 r_3, V_3 + \xi_1 r_2 - \xi_2 r_2, \ldots, v_{3(nf+np)=3nk}, \ldots, V_1 + \xi_2 r_3 - \xi_3 r_2, V_2 + \xi_3 r_1 - \xi_1 r_3, V_3 + \xi_1 r_2 - \xi_2 r_2, \ldots, v_{3(nf+np)=3nk}, V_1, V_2, V_3, \xi_1, \xi_2, \xi_3]^T \in \mathbb{R}^{nd}. \quad (3.40)
\]

Recalling that \( E_i := R[\vec{r}_i]^T \in \mathbb{R}^{3 \times 3} \), the nodal projection \( P \) takes the form

\[
P\vec{v} = \begin{bmatrix}
\begin{bmatrix} I_3 \cdots I_3 \end{bmatrix} & 0 \cdots 0 \\
0 \cdots 0 & I_3 \cdots I_3 & E_3 \cdots E_3 \\
0 \cdots 0 & 0 \cdots 0 & I_3 \cdots I_3 & E_3 \\
0 \cdots 0 & 0 \cdots 0 & 0 \cdots 0 & I_3 \\
\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix} v_* \\
v_* \\
v_* \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} v_* \\
v_* \\
v_* \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} v_* \\
v_* \\
v_* \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} v_1 \\
v_1 \\
v_1 \\
\xi_1 \\
\end{bmatrix} \\
\begin{bmatrix} v_2 \\
v_2 \\
v_2 \\
\xi_2 \\
\end{bmatrix} \\
\begin{bmatrix} v_3 \\
v_3 \\
v_3 \\
\xi_3 \\
\end{bmatrix} \\
\begin{bmatrix} V \\
V \\
V \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} 0 \\
0 \\
0 \\
0 \\
\end{bmatrix} \\
\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix} v_* \\
v_* \\
v_* \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} v_* \\
v_* \\
v_* \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} v_* \\
v_* \\
v_* \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} v_1 \\
v_1 \\
v_1 \\
\xi_1 \\
\end{bmatrix} \\
\begin{bmatrix} v_2 \\
v_2 \\
v_2 \\
\xi_2 \\
\end{bmatrix} \\
\begin{bmatrix} v_3 \\
v_3 \\
v_3 \\
\xi_3 \\
\end{bmatrix} \\
\begin{bmatrix} V \\
V \\
V \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} 0 \\
0 \\
0 \\
0 \\
\end{bmatrix} \\
\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix} v_* \\
v_* \\
v_* \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} v_* \\
v_* \\
v_* \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} v_* \\
v_* \\
v_* \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} v_1 \\
v_1 \\
v_1 \\
\xi_1 \\
\end{bmatrix} \\
\begin{bmatrix} v_2 \\
v_2 \\
v_2 \\
\xi_2 \\
\end{bmatrix} \\
\begin{bmatrix} v_3 \\
v_3 \\
v_3 \\
\xi_3 \\
\end{bmatrix} \\
\begin{bmatrix} V \\
V \\
V \\
\xi \\
\end{bmatrix} \\
\begin{bmatrix} 0 \\
0 \\
0 \\
0 \\
\end{bmatrix} \\
\end{bmatrix}
\end{bmatrix} = \vec{v}. \quad (3.41)
\]

As mentioned before, the effect of \( P \) on a vector is calculated by a simple subroutine and it is not necessary to store \( P \) explicitly as a matrix. This fact is one of the great advantages of the method presented here. In order to upgrade an existing flow–solver to handle fully coupled particulate flows, there are only minor features to add. Besides the administration of the particle data, an ODE–solver for the particles, the selection of the nodes inside a particle (i.e. the particle position \( \tilde{\Pi} \)) and possibly the particle–particle interactions, the only feature that has to be changed in the finite element code is to adapt the matrix*vector product in order to incorporate the projection \( P \), see Program listings 2 and 3. A description of an efficient implementation of these ingredients can be found in [53].
Program listing 2: Subroutine to perform the action of $P$. 
Action of $P^T$

```fortran
subroutine ptmul(u, U, ω, v, V, ξ)
    do d=1,ndim
        V(d) = U(d)
        ξ(d) = ω(d)
    end do
    do i =1,nk
        if( COLOR(i) == BLACK ) then
            do d=1,ndim
                r(d) = x(d, i) − X(d)
            end do
            do d=1,ndim
                v(d, i) = 0
                V(d) = V(d) + u(d, i)
            end do
            ξ(1) = ξ(1) + r(2) * u(3, i) − r(3) * u(2, i)
            ξ(2) = ξ(2) + r(3) * u(1, i) − r(1) * u(3, i)
            ξ(3) = ξ(3) + r(1) * u(2, i) − r(2) * u(1, i)
        else
            do d=1,ndim
                v(d, i) = u(d, i)
            end do
        end if
    end do
end subroutine
```

Program listing 3: Subroutine to perform the action of $P^T$. 
3.3 Error Estimate

In this section we prove an error estimate for the following (quasi–) stationary model problem. For simplicity set $u^* = 0$, $\omega^* = 0$ in the definition of the bilinear form Eq. (3.29) and define the corresponding energy norm by

$$||v||_c^2 := a(v, v).$$  (3.42)

Let $u \in H_c(\Omega_c)$ be the continuous solution of

$$a(u, v) = l(v) \quad \text{for all } v \in H_c(\Omega_c)$$  (3.43)

and $u_h \in X_c$ the corresponding finite element solution fulfilling

$$a(u_h, v_h) = l(v_h) \quad \text{for all } v_h \in X_c.$$  (3.44)

If $u$ is sufficiently smooth, integrating back by parts shows that $u$ solves the following system of equations in strong form (similar as (3.3a)–(3.3g)):}

$$u - \frac{\gamma}{\text{Re}} \nabla \cdot D[u] = \text{rhs}_1 \quad \text{in } \Omega,$$

$$u = 0 \quad \text{on } \Gamma_D,$$

$$u = U + \omega \times r \quad \text{in } \Pi,$$

$$\int_{\Pi} u \, dx + \frac{\gamma}{\text{Re}} \int_{\partial \Pi} D[u] \, n \, ds + \frac{2}{3} \beta M U = \text{rhs}_2 + \int_{\Pi} \text{rhs}_1 \, dx,$$

$$\int_{\Pi} r \times u \, dx + \frac{\gamma}{\text{Re}} \int_{\partial \Pi} r \times D[u] \, n \, ds + \frac{2}{3} \beta I \omega = \text{rhs}_3 + \int_{\Pi} r \times \text{rhs}_1 \, dx,$$

where the terms $\text{rhs}_1$, $\text{rhs}_2$ and $\text{rhs}_3$ are the corresponding terms in Eq. (3.30).

The error estimate is plagued by two problems: the velocity $u$ is not smooth across the interface $\partial \Pi$ in general and since $X_c \not\subset H_c(\Omega_c)$, the finite element method is a nonconforming approximation. In what follows, we assume that $\Omega_c$ is triangulated by a family of conforming, shape regular quasi–uniform triangulations $(T_h)_{h>0}$, where $h$ denotes the mesh size of $T_h$. For a given $h$ we define $X_c$ as in the beginning of this section.

We use Strang’s Second Lemma to bound the error. With the above notation we have:
**Theorem 8** (Strang’s Second Lemma). Assume the following:

1. The bilinear form $a(\cdot, \cdot)$ is bounded on $X_c \times X_c$.

2. There exists $\eta > 0$ such that

$$\inf_{0 \neq u_h \in X_c} \sup_{0 \neq v_h \in X_c} \frac{a(u_h, v_h)}{\|u_h\|_c \|v_h\|_c} \geq \eta.$$ 

Then, the following error estimate holds:

$$\|u_h - u\|_c \leq \left(1 + \frac{\|a\|_{X_c \times X_c}}{\eta}\right) \inf_{v_h \in X_c} \|u - v_h\|_c$$

$$+ \frac{1}{\eta} \sup_{v_h \in X_c} \frac{|l(v_h) - a(u, v_h)|}{\|v_h\|_c}. \quad (3.46)$$

**Proof.** See for instance [24].

**Theorem 9** (Error estimate). Let $u = (u, U, \omega)$ be the continuous solution. With the above notation and additionally assuming that $u \in (H^{2,2}(\Omega))^3$, $u \in (H^{2,2}(\Pi))^3$ as well as $u \in (H^{1,\infty}(\Omega_c))^3$ the error $e := u - u_h$ is bounded by

$$\|e\|_c \leq C(u)h^{1/2}.$$

**Proof.** Clearly, the assumptions of Strang’s 2nd Lemma are fulfilled in the present situation with $\eta = 1$ by definition. It remains to bound the two error terms in the right hand side of Eq. (3.46).

We first bound the interpolation error, i.e. the first term on the right hand side of (3.46). Set $v_h := (v_h, U, \omega) = (I_h u, U, \omega)$, where $I_h$ denotes the standard Lagrange interpolation operator. Thanks to the definition of $X_c$, it follows that $v_h \in X_c$. By standard interpolation estimates it follows for $T \cap \Pi = \emptyset$:

$$\|\nabla(u - v_h)\|_T^2 \leq C h^2 \|\nabla^2 u\|_T^2, \quad (3.47)$$

where $\|\cdot\|_T$ denotes the $L^2$-norm on $T$. If $T \subseteq \Pi$ then $u = U + \omega \times r = v_h$ in $T$ yielding

$$\|\nabla(u - v_h)\|_T^2 = 0. \quad (3.48)$$

For elements at the particle boundary, i.e.

$$T \in \partial \Pi_T := \{ T \in \mathcal{T} | T \cap \Pi \neq \emptyset, T \not\subseteq \Pi \}, \quad (3.49)$$

we proceed as follows. By standard scaling we have

$$\|\nabla(u - u_h)\|_T^2 \leq C h^2 \|\nabla(u - u_h)\|_{\infty,T}^2 \quad (3.50)$$
and then
\[
\sum_{T \in \partial \Pi_T} \|\nabla (u - u_h)\|_T^2 \leq Ch^3 \sum_{T \in \partial \Pi_T} \|\nabla (u - u_h)\|_{\infty, \Omega_c}^2.
\] (3.51)

Since the cardinality \(#\partial \Pi_T \leq C/h^2\) we arrive at
\[
\sum_{T \in \partial \Pi_T} \|\nabla (u - u_h)\|_T \leq Ch^3 \|\nabla (u - u_h)\|_{\infty, \Omega_c}.
\] (3.52)

Because \(u \in H^{1,\infty}(\Omega_c)\), it easily follows that \(\|\nabla (u - u_h)\|_{\infty, \Omega_c}\) is bounded. Thus, taking the square root yields
\[
(\sum_{T \in \partial \Pi_T} \|\nabla (u - u_h)\|_T^2)^{1/2} \leq Ch^{1/2}.
\] (3.53)

Together with (3.47) we get
\[
\|u - v_h\|_c \leq C(u)h^{1/2}.
\] (3.54)

To bound the second term on the right hand side of Eq. (3.46), take an arbitrary \(v_h = (v_h, \mathbf{V}, \xi) \in X_c\). Using Eq. (3.45a)–(3.45e) integration by parts yields
\[
a(u, v_h) - l(v_h) = \int_{\Pi} \left( \frac{u - \gamma}{Re} \nabla \cdot \mathbf{D}[u] - rhs_1 \right) \cdot v_h \, dx + \int_{\partial \Pi} \mathbf{D}[u] \cdot \mathbf{n} \cdot v_h \, ds \\
+ \int_{\Pi} (u - rhs_1) \cdot v_h \, dx + \int_{\Pi} \left( \frac{2}{3} \beta M \mathbf{U} - rhs_2 \right) \cdot \mathbf{V} + \left( \frac{2}{3} \beta I \omega - rhs_3 \right) \cdot \xi.
\] (3.55)

Multiplying Eq. (3.45d) by \(\mathbf{V}\) and Eq. (3.45e) by \(\xi\), subtracting both resulting equations from Eq. (3.55) and then using again \(a \cdot (b \times c) = (c \times a) \cdot b\) we arrive at
\[
a(u, v_h) - l(v_h) = \int_{\Pi} (u - rhs_1) \cdot (v_h - V - \xi \times r) \, dx + \frac{\gamma}{Re} \int_{\partial \Pi} \mathbf{D}[u] \cdot (v_h - V - \xi \times r) \, ds.
\] (3.56)

Thus the consistency error manifests itself by the term \(v_h - V - \xi \times r\) on \(\partial \Pi\), which is not zero in general. We introduce the notation \(\Pi_h := \cup \{T \in T \mid T \subseteq \Pi\}\) and \(\delta \Pi := \Pi \backslash \Pi_h\). Note that \(v_h - V - \xi \times r = 0\) in \(\Pi_h \subseteq \Pi\). Since the strip \(\delta \Pi\) has a
width of the order of $h$ and since $\mathbf{v}_h - V - \xi \times \mathbf{r} = 0$ on $\partial \Pi_h$, Eq. (3.56) can be estimated
\[
a(u, \mathbf{v}_h) - l(\mathbf{v}_h) = \int_{\partial \Pi} (\mathbf{u} - rhs_1) \cdot (\mathbf{v}_h - V - \xi \times \mathbf{r}) \, d\mathbf{x} \\
+ \frac{\gamma}{\text{Re}} \int_{\partial \Pi} D[u] \mathbf{n} \cdot (\mathbf{v}_h - V - \xi \times \mathbf{r}) \, ds \\
\leq C \left( \frac{h^{1/2}}{\text{Re}} \|D^2 \mathbf{u}\| + h \|\mathbf{u}\| + h \|rhs_1\| \right) \|\nabla (\mathbf{v}_h - V - \xi \times \mathbf{r})\| \\
\leq h^{1/2} C \left( \frac{\gamma}{\text{Re}} \|D^2 \mathbf{u}\| + \|\mathbf{u}\| + \|rhs_1\| \right) \|\mathbf{v}_h\|_c,)
\]
where we have used Poincaré’s inequality on $\delta \Pi$, Korn’s inequality and the embeddings $H^1(\delta \Pi) \hookrightarrow L^2(\Pi)$ as well as $H^1(\delta \Pi) \hookrightarrow L^2(\partial \Pi)$.

### 3.3.1 Example

The following computational example shows that the above error estimate is sharp. For $\Omega = B_1(0), \Pi = B_{1/2}(0) \subset \mathbb{R}^3$ and the data
\[
\text{rhs}_1 = \begin{cases} 
[-y(a - br^{-3}), x(a - br^{-3}), 0]^T & \text{in } \Omega \setminus \Pi, \\
[0, 0, 0]^T & \text{in } \Pi,
\end{cases}
\]
\[
\text{rhs}_2 = [0, 0, 0]^T, \text{ rhs}_3 = [0, 0, -\frac{23\pi}{2100}]^T,
\]
$\beta = 3/2, M = 1, I = I, \frac{\gamma}{\text{Re}} = 0.01, a = 3/7, b = -1/14,$

the exact solution of the model problem Eqs. (3.45a)-(3.45e) is
\[
\mathbf{u} = \begin{cases} 
[-y(a - br^{-3}), x(a - br^{-3}), 0]^T & \text{in } \Omega \setminus \Pi, \\
[-y, x, 0]^T & \text{in } \Pi,
\end{cases}
\]
as well as
\[
\mathbf{U} = [0, 0, 0]^T \text{ and } \omega = [0, 0, 1]^T.
\]
Here we have used the notation $r^2 := x_1^2 + x_2^2 + x_3^2$. The rotational symmetric solution $\mathbf{u}$ is plotted on the line $x_2 = x_3 = 0$ in Fig. 3.4.

The computational results are summarized in Tabs. 3.1 and 3.2. In the first table regular refinement is applied on each level, resulting in a global halving of the mesh size from one level to the next one. Clearly the error $\|\mathbf{u} - \mathbf{u}_h\|_c$ is reduced by a factor of 2 if the level is increased by 2, i.e. the EOC is about 1/2. The proof of the error estimates suggests that the reduction in the order of the error estimate to $\sqrt{h}$ applies only near the boundary $\partial \Pi$. To verify this fact, in a further computation additional
Figure 3.4: Plot of exact and numerical solutions of Eqs. (3.45a)–(3.45e) on the line \( x_2 = x_3 = 0 \). The shown solutions correspond to the last line in Tabs. 3.1 and 3.2.

local refinement at the boundary \( \partial \Pi \) is applied to the mesh. Far away from \( \partial \Pi \) the mesh width is reduced by a factor of 2 on each level, while at the boundary it is reduced by a factor of 4. The results are listed in Table 3.2. Representative meshes are shown in Fig. 3.5.

<table>
<thead>
<tr>
<th>Level</th>
<th>DOFs</th>
<th>( |u - u_h|_c )</th>
<th>EOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15k</td>
<td>0.1136</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>108k</td>
<td>0.0790</td>
<td>0.52</td>
</tr>
<tr>
<td>2</td>
<td>824k</td>
<td>0.0545</td>
<td>0.54</td>
</tr>
<tr>
<td>3</td>
<td>6440k</td>
<td>0.0357</td>
<td>0.61</td>
</tr>
<tr>
<td>4</td>
<td>51M</td>
<td>0.0254</td>
<td>0.49</td>
</tr>
</tbody>
</table>

Table 3.1: Errors of the numerical scheme with regular grid refinement applied to each level.

### 3.4 Adaptivity

One of the most important issues in simulating particulate flow is the numerical resolution of the particle’s geometry. In [37] a remeshing technique was used to
<table>
<thead>
<tr>
<th>Level</th>
<th>DOFs</th>
<th>(|u - u_h|_c)</th>
<th>EOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15k</td>
<td>0.1136</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>194k</td>
<td>0.0636</td>
<td>0.84</td>
</tr>
<tr>
<td>2</td>
<td>3310k</td>
<td>0.0315</td>
<td>1.01</td>
</tr>
<tr>
<td>3</td>
<td>56M</td>
<td>0.0154</td>
<td>1.03</td>
</tr>
</tbody>
</table>

Table 3.2: Errors of the numerical scheme with regular grid refinement and additional local refinement at the boundary \(\partial \Pi\) applied to each level.

explicitly follow the geometry in time, Wan and [59] introduced a mesh deformation technique and [31] used Lagrange multipliers. All these techniques have some pros and cons, in particular in 3d. In contrast to the above mentioned methods, we use time–dependent adaptively refined grids based on the bisection method by [12] to sufficiently resolve the region around the particle. This naturally fits the subspace projection method described in the previous subsection. The space \(X_c\) of \(H_c(\Omega_c)\) introduces an error proportional to the square root of the local mesh size on the particle boundary, see Subsection 3.3. This loss of approximation order can be overcome by locally refining the grid at the boundary of the particle. In practical applications, it turned out that a local refinement of 2–3 regular levels was sufficient.

The finite element flow solver NAVIER was used to implement the algorithms presented in this article. NAVIER features built–in local adaptivity, includes linear and quadratic elements and supports a wide range of solvers and preconditioners. For more details see [13].

### 3.5 Preconditioning

In general, the matrix \(P^TAP\) is ill conditioned so that preconditioning is mandatory for an efficient solution strategy. With the notation from Section 3.2.2, \(A\) can be decomposed as

\[
A = \begin{bmatrix}
\bar{A} & 0 \\
0 & C
\end{bmatrix},
\]

(3.57)

where

\[
\bar{A} = \begin{bmatrix}
A_{ff} & A_{fp} \\
A_{pf} & A_{pp}
\end{bmatrix}
\]

(3.58)
Figure 3.5: One eights of the mesh used in the calculations for the model problem (3.45a)–(3.45e). (a) and (b) show one step of global refinement and (c), (d) the same mesh with additional local refinement.

is the system matrix arising from the finite element discretization of \( m(u, v) + \gamma k(u^*, u, v) + \gamma s(u, v) \) and

\[
C = \begin{bmatrix}
\frac{2}{3} \beta M \Pi & 0 \\
0 & \frac{2}{3} \beta I + \frac{2}{3} \beta \omega^* \times I
\end{bmatrix} \in \mathbb{R}^{6 \times 6}.
\] (3.59)

\( p \) refers to the indices inside \( \Pi \) and \( f \) to the indices inside \( \Omega \). For more than one particle, \( C \) is a block diagonal matrix with a \( 6 \times 6 \) block for each particle. For the
Figure 3.6: Successive refinement by bisection of one tetrahedron. In the last step (d) the length of all edges is reduced by a factor of two.

Figure 3.7: Mesh around a particle. For an accurate representation the mesh is locally refined on the particle boundary.

ease of presentation, we will assume a spherical particle so that the term $\frac{3}{2} \beta \omega^* \times I$ in the definition of $C$ can be omitted.

Because of the projection $P$ there is a coupling between the $\tilde{A}$ part of $A$ and the $C$ part. One computes:

$$P^TAP = \begin{bmatrix} \mathbb{I} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & E^T & \mathbb{I} \end{bmatrix} \begin{bmatrix} A_{ff} & A_{fp} & 0 \\ A_{pf} & A_{pp} & 0 \\ 0 & 0 & C \end{bmatrix} \begin{bmatrix} \mathbb{I} & 0 & 0 \\ 0 & 0 & E \\ 0 & 0 & \mathbb{I} \end{bmatrix} = \begin{bmatrix} A_{ff} & 0 & A_{fp}E \\ 0 & 0 & 0 \\ E^T A_{pf} & 0 & C + E^T A_{pp}E \end{bmatrix}.$$  \hspace{1cm} (3.60)

Now, the simplest choice of a preconditioner is a diagonal scaling. According to Eq. (3.60) the diagonal of $P^TAP$ is given by

$$\text{diag}(P^TAP) = [\text{diag}(A_{ff}), 0, \text{diag}(C) + \text{diag}(E^T A_{pp}E)]^T,$$ \hspace{1cm} (3.61)
where $E$ and $E^T$ are the cross–product terms $R[r]$ and its transposed, respectively. Note that $P^TAP$ has a null space corresponding to the nodes inside the particle. As suggested in Section 3.2.2, the system $P^TAP$ is solved with a Krylov–space based iterative solver.

In order to compute the diagonal, the only nonstandard term is the term

$$\text{diag}(E^TA_{pp}E). \quad (3.62)$$

Using the notation “val” as the values of the matrix $A$ stored in the compressed row storage format of Navier and the diagonal entries stored in “diag”, Program listing 4 computes the diagonal of $P^TAP$ (for a scalar case for the ease of presentation).

The reduction in iteration counts using a diagonal preconditioner is listed in table 3.3. This already shows a significant drop in the number of iterations for a typical choice of discretization parameters. However, more sophisticated preconditioners are desirable. Note that standard matrix based preconditioners like SOR, ILU etc. are not applicable since the full matrix $P^TAP$ is not at hand. Instead we will develop a preconditioner based on inexact factorization.

To this end, we first observe that the matrix $C$ is only block diagonal and thus easy to invert. The same property holds for

$$\tilde{D} = C + E^T A_{pp} E, \quad (3.63)$$

(which is also block diagonal with blocks of size $6 \times 6$) as long as no neighboring particles share different nodes from the same tetrahedron. In the latter, there are two additional symmetric off diagonal blocks. The values of these blocks are very small compared to the values in the diagonal blocks. The first approximation is to drop this off diagonal blocks, denote the resulting matrix $D$ and define the Schur complement $S := A_{ff} - A_{fp} E D^{-1} E^T A_{pf}$. Then an approximate decomposition of the matrix $P^TAP$ is given by

$$P^TAP \approx \begin{bmatrix} A_{ff} & 0 & A_{pf}E \\ 0 & 0 & 0 \\ E^T A_{pf} & 0 & D \end{bmatrix} = \begin{bmatrix} \mathbb{I} & 0 & A_{pf} E D^{-1} \\ 0 & 0 & 0 \\ 0 & 0 & \mathbb{I} \end{bmatrix} \begin{bmatrix} S & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} \mathbb{I} & 0 & 0 \\ 0 & 0 & 0 \\ D^{-1} E^T A_{pf} & 0 & \mathbb{I} \end{bmatrix}. \quad (3.64)$$

In order to factor out the null space, for any matrix $T \in \mathbb{R}^{nd \times nd}$ we denote by $T[p,p] \in \mathbb{R}^{(nd-np) \times (nd-np)}$ the matrix obtained by eliminating all columns and rows
Diagonal of $P^TAP$

```fortran
subroutine precon(diag)

    diag(1:nd)= 0

    do i=1,nk
        if( color(i) == BLACK ) then
            ii= particle_number(i)
            do it=rowptr(i),rowptr(i+1)-1
                j= colind(it)
                jj= particle_number(j)
                if( jj.eq.ii ) then
                    diag(ii)= diag(ii) + val(it)
                    diag(ii)= diag(ii) + Et(i)*val(it)*E(i)
                end if
            end do
        end if
    end do
else
    do it=rowptr(i),rowptr(i+1)-1
        if( colind(it) == i ) then
            diag(i)= val(it)
            exit
        end if
    end do
end if
end do
end
```

Program listing 4: Computation of diagonal entries of the product $P^TAP$ using the CRS sparse matrix format.
corresponding to the \( p \)-degrees of freedom. Then

\[
(P^T A P)^{-1}_{[p,p]} \approx \begin{bmatrix}
I & 0 \\
-D^{-1}E^T A_{pf} & I
\end{bmatrix}
\begin{bmatrix}
S^{-1} & 0 \\
0 & D^{-1}
\end{bmatrix}
\begin{bmatrix}
I & -A_{pf} E D^{-1} \\
0 & I
\end{bmatrix}.
\] (3.65)

Since it is not straightforward to invert \( S \), we replace \( S \) by \( A_{ff} \) or with its incomplete LU–factorization \( \text{ILU}(A_{ff}) \) [54], resulting in the two different preconditioners

\[
V_1^{-1} = \begin{bmatrix}
I & 0 \\
-D^{-1}E^T A_{pf} & I
\end{bmatrix}
\begin{bmatrix}
\text{ILU}(A_{ff})^{-1} & 0 \\
0 & D^{-1}
\end{bmatrix}
\begin{bmatrix}
I & -A_{pf} E D^{-1} \\
0 & I
\end{bmatrix},
\] (3.66)

\[
V_2^{-1} = \begin{bmatrix}
I & 0 \\
-D^{-1}E^T A_{pf} & I
\end{bmatrix}
\begin{bmatrix}
A_{ff}^{-1} & 0 \\
0 & D^{-1}
\end{bmatrix}
\begin{bmatrix}
I & -A_{pf} E D^{-1} \\
0 & I
\end{bmatrix}.
\] (3.67)

A comparison of the iteration number obtained by these preconditioners is shown in Table 3.3. A restarted GMRes–Method was used to solve the linear system. The Krylov–space dimension was set to 17, the iteration numbers in the table denote the number of inner iterations until the residuum was smaller than \( 10^{-8} \). The mesh was locally refined two times around the particles’ boundaries, which means the mesh width \( h \) was reduced by a factor of four near the boundaries. With no preconditioner applied, the iteration count was much beyond 1000, which is impractical for real simulations. With the diagonal preconditioner the iteration count is dramatically reduced by at least a factor of 10. Further improvements are made with the inexact factorization preconditioners \( V_1 \) and \( V_2 \). In terms of iteration numbers, \( V_2 \) is clearly superior to \( V_1 \). However, \( V_2 \) is expensive in terms of computational costs because one has to invert \( A_{ff} \) and then carry out a full backward substitution in each iteration. Especially with a huge number of unknowns it might be more efficient to use an iterative solver to invert \( A_{ff} \) in each iteration but it is also very expensive. In contrast, \( V_1 \) has low computational costs as one only has to perform one incomplete factorization, and then apply a sparse backward substitution in each iteration. In addition one could replace \( A_{ff} \) in \( V_2 \) with different multilevel approximations \( \text{BPX}(A_{ff}) \) and \( \text{MGV}(A_{ff}) \) denoting the application of one BPX [8] iteration and one multigrid V–cycle [54, 34] on \( A_{ff} \), respectively. This was analyzed in detail in [53]. The observations made therein indicates, that the results in means of iteration counts are quite good for both multilevel approximations. In terms of overall runtime, however, the ILU based preconditioner \( V_1 \) performs best.
\[ \alpha = \rho / \rho_\Pi \]

<table>
<thead>
<tr>
<th>#Particles</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>100</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>1442</td>
<td>1401</td>
<td>1442</td>
<td>1364</td>
<td>1378</td>
<td>1402</td>
<td>1274</td>
<td>1262</td>
<td>1228</td>
</tr>
<tr>
<td>Diagonal</td>
<td>116</td>
<td>130</td>
<td>143</td>
<td>114</td>
<td>126</td>
<td>136</td>
<td>105</td>
<td>112</td>
<td>116</td>
</tr>
<tr>
<td>V_1</td>
<td>44</td>
<td>50</td>
<td>55</td>
<td>43</td>
<td>48</td>
<td>53</td>
<td>39</td>
<td>42</td>
<td>45</td>
</tr>
<tr>
<td>V_2</td>
<td>23</td>
<td>26</td>
<td>27</td>
<td>22</td>
<td>24</td>
<td>26</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 3.3: Number of iterations to solve the system \( P^T A P \) with the proposed preconditioner for different density ratios and different number of particles. The time step size was 0.02 and the Reynolds number 10. The number of grid nodes was 103053, 234210 and 316275 for 100, 500 and 1000 particles, respectively.

3.6 Particle Interaction

The algorithm for one particle extends straightforward to a method for many particles. Forces between particles then play an important role. For a huge number of particles, the evaluation of these forces can cost a large amount of computation time. Therefore, a crucial point in the implementation is to find an optimal algorithm for the evaluation of the forces. In the remainder, a detailed description of the particle interactions will be given.

3.6.1 Newton’s Method for the Predictor

Let the interparticulate force \( F_\Pi \) and the global force \( F_G \) only be governed by the particle positions, and not by the velocities. Then the predictor for \( N \) interacting particles is described by the set of equations

\[
\ddot{\mathbf{X}}_i = \mathbf{X}_i^k + \frac{\tau}{2} \left( \mathbf{U}_i^k + \ddot{\mathbf{U}}_i \right) \quad i = 1, \ldots, N,
\]

\[
\ddot{\mathbf{U}}_i = \mathbf{U}_i^k + \frac{\tau}{M_i} \left( F_G \left( \frac{\mathbf{X}_i^k + \ddot{\mathbf{X}}_i}{2} \right) + \sum_{j \neq i} F_\Pi \left( \frac{\mathbf{X}_i^k + \ddot{\mathbf{X}}_i, \mathbf{X}_j^k + \ddot{\mathbf{X}}_j}{2} \right) \right) \quad i = 1, \ldots, N,
\]

see also Eqs. (3.20a)-(3.20b). The vector of unknowns is

\[
[\ddot{\mathbf{U}}, \ddot{\mathbf{X}}]^T := [\ddot{\mathbf{U}}_1, \ldots, \ddot{\mathbf{U}}_N, \ddot{\mathbf{X}}_1, \ldots, \ddot{\mathbf{X}}_N]^T.
\]
Note that each entry is a vector of dimension three. The nonlinear vector valued function $g : \mathbb{R}^{6N} \to \mathbb{R}^{6N}$ is defined as

$$g\left(\begin{bmatrix} U \\ X \end{bmatrix}\right) := \begin{bmatrix} U_N - U^k_N - \frac{\tau}{M^k} \left( F_G \left( \frac{X^k_N + X^N}{2} \right) + \sum_{j \neq N} F_G \left( \frac{X^k_j + X^j}{2} \right) \right) \\
\vdots \\
X^k_N - X^k - \frac{\tau}{2} (U^k + U_1) \\
\vdots \\
X^N - X^N - \frac{\tau}{2} (U^N + U_N) \end{bmatrix}.$$  \hspace{1cm} (3.70)

If a unique solution of problem (3.68a)-(3.68b) exists, then $[U^{k+1}, X^{k+1}]^T$ is the only zero of $g$ and the identity $g([U^{k+1}, X^{k+1}]^T) = 0$ holds. Newton’s method is applied to search the zero of $g$. For a smooth function $g$, the algorithm is defined as

$$J_g(x_n) (x_{n+1} - x_n) = -g(x_n),$$  \hspace{1cm} (3.71)

where $J_g$ is the Jacobian of $g$ with respect to the variable $x$ and $x_0$ is an initial guess.

With the definition

$$f_{X,i}(X_j) := \begin{cases} \frac{\partial}{\partial X_i} F_G \left( \frac{X_j + X_i}{2} \right) & \text{for } i = j \\
\frac{\partial}{\partial X_i} F_G \left( \frac{X_j + X_i}{2} , \frac{X^k_j + X^k}{2} \right) & \text{for } i \neq j \end{cases},$$  \hspace{1cm} (3.72)

the Jacobian for $g$ in (3.70) is given by

$$J_g\left(\begin{bmatrix} U \\ X \end{bmatrix}\right) = \begin{bmatrix} \mathbb{I} \\
& \cdots \\
& -\frac{\tau}{2M_i} f_{X_1}(X_1) \\
& \vdots \\
& \vdots \\
& \frac{\mathbb{I}}{2M_N} f_{X_N}(X_N) \end{bmatrix}.$$  \hspace{1cm} (3.73)

In each iteration, the Jacobian is solved with the GMRes algorithm. In the final algorithm, Armijo’s rule [2] is applied for damping

1. Set $\ell = 1$ and $[U_\ell, X_\ell] = [U^k, X^k]$.

2. While $\|g(U_\ell, X_\ell)\| \geq \text{tol.}$

3. Solve for

$$J_g(U_\ell, X_\ell) \begin{bmatrix} \Delta U \\ \Delta X \end{bmatrix} = -g(U_\ell, X_\ell).$$  \hspace{1cm} (3.74)
4. Find the minimum \( i \in \{0, 1, \ldots, i_{\text{max}}\} \) such that
\[
\|g(U_\ell + 2^{-i} \Delta U, X_\ell + 2^{-i} \Delta X)\| \leq \left(1 - 2^{-(i+1)}\right) \|g(U_\ell, X_\ell)\| \tag{3.75}
\]
is true. If the statement is always false, set \( i = i_{\text{max}} \). Compute
\[
\begin{bmatrix}
U_{\ell+1} \\
X_{\ell+1}
\end{bmatrix} = \begin{bmatrix}
U_\ell \\
X_\ell
\end{bmatrix} + \frac{1}{2^\ell} \begin{bmatrix}
\Delta U \\
\Delta X
\end{bmatrix}, \tag{3.76}
\]
increase \( \ell \to \ell + 1 \), go to 2.

Depending on the number of particles and the magnitude of the force only one to four Newton iterations are needed to fulfill the termination condition. This condition is cheap to evaluate, as its result can be used as right hand side for the next iteration and it gives good control over the error. The condition for damping is most often true in situations where a lot of particles are about to collide. Then in general one damping step is sufficient and in the next Newton iteration damping is not required anymore. The inversion of the Jacobian seems to be an easy task for GMRes, except for the situation with a lot of near colliding particles. Even in the latter case 100 to 150 iterations where enough.

3.6.2 Barnes–Hut Algorithm

The evaluation of the particle–particle interactions \( \mathbf{F}_\Pi \) in Eq. (3.70) and its Jacobian is a very crucial task. If it is implemented in a naive way, the computation costs are proportional to the number of particles squared. For a huge amount of particles (more than 1000) this is the bottleneck of the whole application. To avoid this, we use the Barnes–Hut algorithm which reduces the complexity to \( \mathcal{O}(N \log(N)) \) with an acceptable loss of accuracy, see [3]. The algorithm requires that the force decays fast enough and that the cause of the force is summable, e.g. mass for gravitational forces or charge for Coulomb forces. These two points guarantee that a collective of particles which is far away can be treated as a single large particle. These properties hold e.g. for Coulomb forces between charged particles. The force exerted by a group of \( N \) distant particles on a single independent particle is described by
\[
F = \sum_{i=1}^{N} C \frac{q_i}{D_i^2}, \tag{3.77}
\]
where \( D_i \) is the distance to the \( i \)--th particle, \( q_i \) its charge and \( C \) a constant. The force decays very fast and its cause, the charge \( q_i \), can be summed up. The idea of the algorithm is to merge the forces created by a group of distant particles into a
single force on one pseudo-particle. The charge of the pseudo-particle is the sum of
the charges of the particle group and its position is the mean position (weighted by
charges) of the group of particles. Finally the force can be computed as

\[ F \approx C \frac{q_{\text{sum}}}{D_{\text{mean}}} \]  

(3.78)

The costs of calculating \( q_{\text{sum}} \) and \( D_{\text{mean}} \) grow only logarithmic with \( N \) in the context
of an octree. For a more detailed description of the Barnes–Hut algorithm see Figure
3.8 and its caption.

In addition to the long range Coulomb forces, we also add short term repulsive
forces in order to avoid particle overlap. The octree in the Barnes–Hut algorithm
can also be used to effectively calculate forces between neighboring particles. With
the tree given, particle neighbors are easily found by determining the neighboring
nodes of the tree which is a fast operation.

**Figure 3.8:** Idea of the Barnes–Hut algorithm. The first step is to construct an octree
(which is displayed as a quadtree in this figure). The leafs of the octree are either
particles (white circles) or empty. In the construction process, a pseudo-particle
(gray) is created in each cell. This pseudo-particle encodes the properties of the
(pseudo-)particles of its child nodes. In this example, the black particle interacts
with its direct neighbor, with a pseudo-particle on level 1 (lightgray) and a pseudo-
particle on level 0 (darkgray). The criterion which determines which pseudo-particle
to interact with is called the multipole acceptance criterion (MAC). The pseudo-
particle is related due to the octree to a boxsize \( d \) and has a distance \( l \) to the particle
to evaluate. If the ratio \( d/l \) is smaller than a given number \( \Theta \), then an evaluation
with this pseudo-particle is allowed, else go to the children of the pseudo-particle and
check again. Usually \( \Theta \) is set to about 0.7.
3.6.3 Short Term Repulsive Forces

The particle–particle and particle–wall collisions are modeled by a local force term. This force acts in a very small range $d$ of about 10% of the particle radius (surface–surface distance). Beyond this range, the force is negligible and thus set to zero. The magnitude of the force is assumed to be proportional to $1/D^2$ where $D$ is the distance between the surface of two particles or the surface–wall distance. For Newton’s method, it is necessary to use a differentiable function as forcing term. A polynomial of second degree is constructed to obtain a $C^1$ extension of this function. With the aforementioned restrictions, the magnitude of the force is described as

$$f(D) = A \begin{cases} \frac{1}{D^2} & \text{for } 0 < D \leq \frac{d}{2} \\ \frac{16D^2}{d^4} - \frac{32D}{d^3} + \frac{16}{d^2} & \text{for } \frac{d}{2} < D \leq d \\ 0 & \text{for } d < D \end{cases},$$

(3.79)

where $A$ is the proportionality constant which has to be fitted to the situation. For a plot of this function and its derivative see Fig. 3.9.

![Figure 3.9: Plot of the function $f(D)$ (a) and its derivative (b) for $d = A = 1$.](image)

**Remark 10.** For the construction, a polynomial of third degree is used to control the derivatives and function values at the end $d$ and the transition point $D_T$, $0 < D_T < d$. A good idea would be to set the transition point such that the jump of the second derivative is minimized. This is the case for $D_T$ near $d/2$. For setting $D_T$ exactly to $d/2$ the leading coefficient of the polynomial is zero, resulting in a second degree polynomial which gives a compact formula.

Consider the two spherical particles $i, j$ with coordinates $X_i, X_j$ and radii $R_i, R_j$. The oriented midpoint distance vector is then $D_X := X_j - X_i$. For the evaluation
of the force the surface to surface distance $D$ is needed which can be calculated by

$$D = \left\| \frac{D_X}{\|D_X\|} \right\| \left( \frac{\|D_X\|}{\|D_X\|} - R_i - R_j \right).$$

Finally the force on particle $i$ is given by

$$f(D) = - \frac{D}{\|D\|} f(\|D\|) = - \frac{D_X}{\|D_X\|} f(\|D\|).$$

(3.80)

For Newton’s method we must calculate the gradient of this function with respect to $X_i$ and $X_j$. From the definition of $D_X$ and $f$, we receive

$$\nabla_{X_i} f(D) = - \nabla_{X_j} f(D), \quad (3.81)$$

$$\nabla_{X_i} f(D) = \left( \nabla_{X_i} f(D) \right)^T, \quad (3.82)$$

which is helpful in the implementation. The $\ell$–th derivative of the $k$–th component of $f$ with respect to $X_i$ is

$$\partial_{\ell} f_k(D) =
\partial_{\ell} \left( - \frac{D_X}{\|D_X\|} f(\|D\|) \right) = - \frac{D_X}{\|D_X\|} \partial_{\ell} f(\|D\|) \frac{\partial D_X}{\|D_X\|} =
- \frac{D_X}{\|D_X\|} f'(\|D\|) \partial_{\ell} \|D\| - f(\|D\|) \partial_{\ell} \frac{D_X}{\|D_X\|}. \quad (3.83)$$

The expressions in detail are

$$f'(D) = A \begin{cases} 
- \frac{2}{D^3} & \text{for } 0 < D \leq \frac{d}{2} \\
\frac{2D}{D^2} - \frac{2d}{D^2} & \text{for } \frac{d}{2} < D \leq d \\
0 & \text{for } d < D 
\end{cases}, \quad (3.84a)$$

$$\frac{D_X}{\|D_X\|} = \frac{1}{\|D_X\|^3} \begin{cases} 
(X_{j,l} - X_{i,l})^2 - \|D_X\| & \text{for } l = k \\
(X_{j,k} - X_{i,k}) (X_{j,l} - X_{i,l}) & \text{for } l \neq k 
\end{cases}, \quad (3.84b)$$

$$\partial_{\ell} \frac{D_X}{\|D_X\|} = \frac{D_X, \ell D_X, k - \delta_{\ell,k} \|D_X\|^2}{\|D_X\|^3}, \quad (3.84c)$$

$$\partial_{\ell} \|D\| = - \frac{D_X, \ell}{\|D_X\|}. \quad (3.84d)$$

With this knowledge we are able to simplify the previous formula a little bit

$$\partial_{\ell} f_k(D) = \frac{D_X, \ell D_X, \ell}{\|D_X\|^2} \left( f'(\|D\|) - \frac{f(\|D\|)}{\|D_X\|} \right) + \delta_{\ell,k} \frac{f(\|D\|)}{\|D_X\|}. \quad (3.85)$$

For the particle–wall interaction we set $R_j = 0$ and $X_j$ to the nearest point to the wall.
3.7 Validation

In this section numerical simulations are presented with the intention of a quantitative validation of the method outlined above. To this end, two different validation experiments are considered. In the first one, a single sedimenting particle is computed. We calculate its drag force and compare it to values from literature. The second experiment, viscometer, focuses on the flow of many particles. Its goal is to approve Einstein’s relation for the effective viscosity of a suspension with low particle volume fraction. Besides these two important test cases, there are some other experiments for very basic validation simulations, like constant translational or rotational flow.

3.7.1 Sedimentation

The setting of this first example is rather simple but serves as a good validation experiment though. Consider a single particle placed in the center of a cylinder of diameter $D$ filled with a Newtonian fluid whose density is lower than that of the particle. Gravity $g$ accelerates the particle until it reaches a terminal velocity $U$. The drag coefficient $c_D$ of the particle is given in terms of the terminal velocity by

$$c_D := \frac{2\tilde{F}_D}{\rho AU^2} = \frac{2\tilde{M}g}{\rho AU^2} = \frac{8}{3} \left( \frac{\rho \Pi}{\rho} - 1 \right) \frac{Rg}{U^2}$$

(3.86)

with $F_D$ the drag force which is by definition the force component in the direction of the particle’s velocity, $R$ the particle’s radius, $\tilde{M} = (\rho \Pi - \rho)V$ its reduced mass, $V = \frac{4}{3}\pi R^3$ its volume and $A = \pi R^2$ the projected area.

The simulations were carried out for a variety of different Reynolds numbers and different particle–cylinder ratios $\lambda = 2R/D$. In the simulations, it turned out that one cannot neglect the influence of the walls, especially for low Reynolds numbers. Thus the analytic results (which are for infinite large domains) are extended by a model for wall correction from [17] to take this effect into account. To this end the formula for the drag coefficient is

$$c_D = \frac{24}{Re} \left( 1 + 0.1315Re^{0.82 - 0.05 \log(Re)} \right) + \frac{24(K + 1)}{Re},$$

(3.87)

$$K = \frac{1}{1 - 2.104\lambda + 2.09\lambda^3 - 0.38\lambda^5},$$

(3.88)

where the first term is the drag value for $\lambda = 0$ and the second term accounts for the wall effect. Several different values for time step sizes and grid sizes were tested to assess appropriate discretization parameters. For (too) large grid sizes, an oscillating
behavior of the particle could be observed. Also in this case the terminal velocity was overestimated. On the other side, if the time step size was not small enough, the settling velocity was underestimated. The four simulations in the right bottom corner of Fig. 3.10 gave sufficiently accurate results. The discretized particles with number of nodes from 117 to 2469 in Fig. 3.12 correspond to the four different mesh widths in Fig. 3.10. A further refinement did not improve the results significantly. The particle discretized with only 21 nodes did not deliver a reasonable result.

![Figure 3.10: Velocity versus time. From left to right: finer mesh width. From top to bottom: smaller time step size. The analytic asymptotic value for the velocity is indicated by a dotted line. Time and length scales are dimensionless quantities, nondimensionalized by the particle’s radius and its terminal velocity.](image)

The simulations in Fig. 3.11 show a good agreement with the data from literature. The overall error including the wall effect is about 3-5%, which we regard as rather satisfying.

### 3.7.2 Viscometer

In [22] Einstein derived the effective viscosity $\eta_{eff}$ of a suspension consisting of a Newtonian fluid loaded with spherical particles in the case of small particle volume
Figure 3.11: Drag curves for two different particle–cylinder diameter ratios $\lambda$. The numerical results are compared with results from Clift et al.[17].

The effective viscosity is given by:

$$\frac{\eta_{\text{eff}}}{\eta} = 1 + 2.5\Phi,$$

(3.89)

where $\Phi$ is the particle volume fraction in the fluid. This result holds for noninteracting particles and is independent of their radii. This equation is only valid when $\Phi < 0.01$. For larger volume fractions hydrodynamic interactions between the particles must be considered. This was done by [41] and [5] where they derived the following extensions for the relative viscosity:

$$\frac{\eta_{\text{eff}}}{\eta} = \left(1 - \frac{\Phi}{\Phi_m}\right)^{-2.5\Phi_m},$$

(Krieger) (3.90)

$$\frac{\eta_{\text{eff}}}{\eta} = 1 + 2.5\Phi + 6.2\Phi^2,$$

(Batchelor) (3.91)

where $\Phi_m = 0.74$ is the maximum packing fraction. To compare results from simulations with our code with this analytical solutions, we performed simulations for different volume fractions and particle radii. The effective viscosity was obtained by simulating a viscometer.
More precisely, we simulated a Couette flow and then measured the force of the fluid acting on the boundary. For a sketch see Fig. 3.13. To this end, consider a shear flow with shear rate \( c = \frac{u_0}{H} \) given by

\[
\mathbf{u} = \begin{bmatrix} cx_2 \\ 0 \\ 0 \end{bmatrix}; \quad D[\mathbf{u}] = \begin{bmatrix} 0 & c & 0 \\ c & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \quad p = 0. \tag{3.92}
\]

The force \( F = \int_S \sigma \cdot \mathbf{n}_S \, ds \) acting on the bottom plane \( S \) is evaluated. In this case
the constant stress \( \tau = \sigma \cdot n_S \) can be computed via

\[
\sigma \cdot n_S = \tau = \frac{F}{|S|},
\]

(3.93)

where \( |S| \) is the area of \( S \). With the definition of the stress tensor and restricting the result to the first component, the formula for the viscosity is then

\[
\eta = \frac{F_1}{c|S|}.
\]

(3.94)

Thus, the effective viscosity \( \eta_{eff} \) can be computed by the above formula.

Simulations were carried out on a hexahedral domain with periodic boundary conditions in \( x_1 \) and \( x_3 \) direction. Prescribing appropriate boundary values for \( u \) at the bottom and top, a Couette flow was obtained. For suspensions with different particle volume fractions and particle radii the forces on the \( x_1-x_3 \)-planes are evaluated, yielding a value for the effective viscosity via Eq. (3.94). For small particle volume fractions \( \Phi \) the results displayed in Fig. 3.14 are identical with Einstein’s prediction (3.89). For larger volume fractions we obtain greater effective viscosities which is covered by Krieger’s and Batchelor’s prediction, (3.90) and (3.91) respectively.

![Figure 3.14: Results of the viscometer calculations.](image-url)
3.7.3 Drafting Kissing Tumbling

This qualitative experiment considers the sedimentation of two spherical particles in a cylindrical domain. The particles are aligned on the center line, separated by a small distance of a few particle diameters in the starting configuration. When the gravity starts acting, one can observe the following situations.

- Both particles start accelerating. There is no interaction between them.
- Drafting: After a while the slipstream of the first particle causes the second one to accelerate a little more.
- Kissing: A near impact is inevitable as the second particle has a higher velocity than the first one. The slower particle is pushed by the faster one (The force is transferred by the viscous fluid).
• Tumbling: The above situation is unstable. To solve this conflict the slower particle moves aside so that the faster particle can overtake it. This can be interpreted as tumbling when observed in relative coordinates.

The particle Reynolds number in the numerical experiment was about 200. The four phases described in the itemization are displayed in Fig. 3.16.
Chapter 4

Nernst–Planck–Poisson Model
(NPP)

The Nernst–Planck equations are a continuum model for the description of the motion of small charged particles (e.g., ions) in a fluid. The flux of the ions is influenced by the ionic concentration gradient, an electric field and the fluid velocity. \( \Omega \subset \mathbb{R}^d, d \in \{2, 3\} \) is a domain.

\[
\frac{\partial n_s}{\partial t} - k_B T \mu_s \Delta n_s - e z_s \mu_s \nabla \cdot (n_s \nabla \varphi) + \mathbf{u} \cdot \nabla n_s = 0 \quad \text{in} \ \Omega, \quad (4.1a)
\]

\[
n_s = n_{D,s} \quad \text{on} \ \Gamma_{in}, \quad (4.1b)
\]

\[-(e z_s \mu_s n_s \nabla \varphi + k_B T \mu_s \nabla n_s) \cdot \mathbf{n} = 0 \quad \text{on} \ \partial \Omega \setminus \Gamma_{in}, \quad (4.1c)
\]

with \( s = 1, \ldots, n_{\text{spec}} \) where \( n_{\text{spec}} \) is the number of species, \( n_s \) is the concentration of the \( s \)–th species, \( \varphi \) is an electric potential, \( \mathbf{u} \) is a given fluid velocity, \( z_s \) is the charge number (valence), \( e \) is the unit charge, \( T \) is the absolute fluid temperature and \( k_B \) is the Boltzmann constant. The mobility \( \mu \) is related to the diffusion coefficient \( D \) by the Einstein relation \( D = \mu k_B T \). Note that the mobility used here is defined by the asymptotic velocity \( v_a \) resulting from an applied force \( F \)

\[
\mu := \frac{v_a}{F}. \quad (4.2)
\]

An alternative definition is often found in literature, using the electric field \( E \), which is \( v_a = \mu E \). The conversion is done by \( \mu = \mu_E / q \) where \( q \) is the charge of one ion. At the inflow boundary \( \Gamma_{in} = \{ \mathbf{x} \in \partial \Omega \mid \mathbf{u}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) < 0 \} \) Dirichlet boundary conditions \( n_{D,s} \) (4.1b) are applied. The so-called nonreacting boundary condition (4.1c) is imposed on the remaining border. Like usual, \( \mathbf{n} \) is the outer normal of \( \Omega \). By virtue of Gauss’ law, the charged ions generate an electric potential which is...
described in differential form by the Poisson equation

\[-\nabla \cdot (\varepsilon_0 \varepsilon_r \nabla \varphi) = \sum_{s=1}^{n_{\text{spec}}} \varepsilon_s n_s \quad \text{in } \Omega, \quad (4.3a)\]

\[\varphi = \varphi_D \quad \text{on } \partial \Omega. \quad (4.3b)\]

Here, \(\varepsilon_0\) is the vacuum permittivity and \(\varepsilon_r\) is the relative permittivity. The Nernst–Planck equation together with the Poisson equation are called Nernst–Planck–Poisson equations (NPP) and are a standard model in electrochemistry [36, 56, 57]. These equations are nonlinearly coupled. However, being a standard tool does not mean that its numerical treatment is straight forward or well documented. In the next sections, we will develop a numerical scheme for the NPP system which is stable and reliable.

### 4.1 Dimensionless Formulation

The dimensionless form of the NPP system is given by

\[-\tilde{\nabla} \cdot \tilde{\varepsilon}_r \tilde{\nabla} \tilde{\varphi} - R^c \sum_{s=1}^{n_{\text{spec}}} z_s \tilde{n}_s = 0 \quad \text{in } \tilde{\Omega}, \quad (4.4a)\]

\[\tilde{\varphi} = \tilde{\varphi}_D \quad \text{on } \partial \tilde{\Omega}, \quad (4.4b)\]

\[\partial_t \tilde{n}_s - S^c_s \delta \tilde{n}_s - K^c_s z_s \tilde{\varphi} \cdot \tilde{u} + \tilde{u} \cdot \tilde{\nabla} \tilde{n}_s = 0 \quad \text{in } \tilde{\Omega}, \quad (4.4c)\]

\[\tilde{n}_s = \tilde{n}_{D,s} \quad \text{on } \tilde{\Gamma}_{\text{in}}, \quad (4.4d)\]

\[-(K^c_s z_s \tilde{n}_s \tilde{\nabla} \tilde{\varphi} + S^c_s \tilde{\nabla} \tilde{n}_s) \cdot \tilde{n} = 0 \quad \text{on } \partial \tilde{\Omega}\setminus \tilde{\Gamma}_{\text{in}}, \quad (4.4e)\]

where we nondimensionalized using the characteristic values

\[x = x_c \tilde{x}, \quad n = n_c \tilde{n}, \quad \varphi = U_c \tilde{\varphi}, \quad t = t_c \tilde{t}. \quad (4.5)\]

This gives the characteristic numbers

\[R^c = \frac{x_c^2 c n_c}{\varepsilon_0 U_c}, \quad S^c = \frac{t_c D_s}{x_c^2}, \quad K^c = \frac{t_c D_s e U_c}{k_B T x_c^2}. \quad (4.6)\]

This specific nondimensionalization with a set of characteristic numbers for each species allows us to use different mobilities and charge numbers without having different time scales for each ion species. In the standard approach \(t_c\) is set independently for each species, which then results in different time scales for the nondimensional Nernst–Planck equations. The choice

\[x_c = \left(\frac{\varepsilon_0 k_B T}{e^2 n_c}\right)^{\frac{1}{2}}, \quad t_c = \frac{x_c^2}{D_1}, \quad U_c = \frac{k_B T}{e}, \quad (4.7)\]

would lead to the usual dimensionless formulation found in literature with \(R^c = 1\), \(S^c_1 = 1\) and \(K^c_1 = 1\). For the readers convenience we skip the~ in the remainder.
4.2 Time Discretization

The system (4.4a)–(4.4c) constitutes a differential algebraic equation (DAE) of index 1. This means that the first time derivative of the Poisson equation can be transformed into an explicit expression for $\partial_t \varphi$. The sometimes ignored fact about DAE is that order reduction may occur when higher order time stepping methods are used for time discretization of this nonlinear system, see [46, 47, 49, 50]. A time discretization which does not suffer from order reduction in the case of a DAE of index 1 is called algebraically stable.

Since the above system is also stiff, an A– or L–stable (implicit) scheme [21] is required. In order to avoid unnecessarily small time steps because of possible oscillations, a L–stable scheme is preferable.

Splitting the system into two decoupled linear ones seems appealing at first sight. However, while this idea works very well for small ion concentration, where the coupling between the Poisson and Nernst–Planck equation is weak, for higher concentration a very small time step is required to get a stable solution. Therefore, we solve the fully coupled problem.

For sufficient accuracy, the time discretization scheme should be of second order. There are only two feasible schemes known which fulfill the requirements stated above. The first one is BDF2 [20, 9], a multistep method. The second one, called DIRK(2,2) [1] is a diagonally implicit Runge–Kutta method. Other schemes that comply with the requirements are fully implicit Runge–Kutta methods which results in a two times larger nonlinear system and are therefore not practical. The BDF2 method requires only one nonlinear solve in each time step, whereas the specific Runge–Kutta method requires two nonlinear solves. For this reason, we selected BDF2 as our algorithm of choice. Note that BDF2 needs two previous time steps. To start the simulation one can calculate the first time step with a one step method, or assume to start the simulation at rest. In this case the values at time $t^{-1} = -\tau$ are the same at time $t^0 = 0$. $\tau$ is the constant time step size, $\gamma = \frac{2}{3} \tau$ and $k$ the number of time step. For the derivation we formally add the time derivative $\epsilon \partial_t \varphi$ to the Poisson equation, discretize the system with BDF2 and send $\epsilon \to 0$. Then we have
Time discretization of NPP system

For given \( n_k^s, n_s^{k-1} \) for \( s = 1, \ldots, \text{n}_{\text{spec}} \) find

\[ (\varphi^{k+1}, n_1^{k+1}, \ldots, n_{\text{n}_{\text{spec}}}^{k+1}) \in H^1_\varphi(\Omega) \times H^1_1(\Omega) \times \cdots \times H^1_{\text{n}_{\text{spec}}}(\Omega) \]

such that

\[ -\nabla \cdot \varepsilon_r \nabla \varphi^{k+1} - R^e \sum_{s=1}^{\text{n}_{\text{spec}}} z_s n_s^{k+1} = 0, \quad (4.8a) \]

\[ n_s^{k+1} - \gamma S^c_s \Delta n_s^{k+1} - \gamma K^c_s z_s \nabla \cdot (n_s^{k+1} \nabla \varphi^{k+1}) + \gamma u^{k+1} \cdot \nabla n_s^{k+1} = \frac{4}{3} n_s^k - \frac{1}{3} n_s^{k-1}, \quad (4.8b) \]

4.3 Spatial Discretization

Linear finite elements are used for space discretization. Compared to finite differences, finite elements allow to incorporate the nonreacting boundary condition (4.4e) in a more natural manner and the possibility to use local grid adaptivity in an easy way. In our case, adaptivity is mandatory to resolve the boundary layers. Compared to existing computational work on NPP system, see [19, 11, 42, 45, 61], this approach differs in the way that finite elements are combined with a higher order time discretization scheme that is free from order reduction. The first step is to derive the weak formulation of the time discretized NPP system (4.8a)-(4.8b).

To simplify the notation, we assume \( \Gamma_{in} \) to be constant in time. For given \( u^{k+1} \in (H^1(\Omega))^d \) and \( n_k^s, n_s^{k-1} \in H^1(\Omega) \) we derive the weak formulation of Eqs. (4.8a)-(4.8b) by multiplying all equations with a testfunction from \( H^1_\varphi(\Omega) \times H^1_1(\Omega) \times \cdots \times H^1_{\text{n}_{\text{spec}},0}(\Omega) \) and summing them up. Partial integration is applied on the terms with the characteristic numbers \( S^c_s \) and \( K^c_s \). The boundary term is identical to the nonreacting boundary condition and vanishes.

Find \((\varphi^{k+1}, n_1^{k+1}, \ldots, n_{\text{n}_{\text{spec}}}^{k+1}) \in H^1_\varphi(\Omega) \times H^1_1(\Omega) \times \cdots \times H^1_{\text{n}_{\text{spec}}}(\Omega) \) such that

for all \((\varphi, v_1, \ldots, v_{\text{n}_{\text{spec}}}) \in H^1_\varphi(\Omega) \times H^1_{1,0}(\Omega) \times \cdots \times H^1_{\text{n}_{\text{spec}},0}(\Omega) \)

\[ (\nabla \varphi^{k+1}, \nabla \varphi) - R^e \sum_{s=1}^{\text{n}_{\text{spec}}} z_s (n_s^{k+1}, \varphi) + 
\sum_{s=1}^{\text{n}_{\text{spec}}} (n_s^{k+1} - \frac{4}{3} n_s^k + \frac{1}{3} n_s^{k-1}, v_s) + \gamma \sum_{s=1}^{\text{n}_{\text{spec}}} S^c_s (\nabla n_s^{k+1}, \nabla v_s) 
+ \gamma \sum_{s=1}^{\text{n}_{\text{spec}}} K^c_s z_s (n_s^{k+1} \nabla \varphi^{k+1}, \nabla v_s) + \gamma \sum_{s=1}^{\text{n}_{\text{spec}}} (u^{k+1} \cdot \nabla n_s^{k+1}, v_s) = 0 \]  (4.9)
where $H_1^+(\Omega)$ and $H_1^-(\Omega)$ are subsets of $H^1(\Omega)$ whose elements respect the Dirichlet boundary conditions (4.4b) and (4.4d). They are defined as

$$H_1^+(\Omega) = \{ v \in H^1(\Omega) | \text{tr}(v) = \varphi_D \}$$

(4.10)

and similar for $H_1^-(\Omega)$, where $\text{tr} : H^1(\Omega) \to L^2(\partial\Omega)$ is the standard trace operator. The spaces denoted by $H_{\varphi,0}^1(\Omega)$ and $H_{s,0}^1(\Omega)$ are the corresponding spaces with homogeneous boundary data. Uniqueness and existence of solutions of this system with $n_{\text{spec}} = 2$ are proven in [28]. Stability results for stationary solutions can be found in [30, 16].

Problem (4.9) is discretized using finite dimensional approximations of the function spaces $H_1^+(\Omega)$ and $H_1^-(\Omega)$. Let $T$ be a triangulation of $\Omega$ with mesh width $h$ and $X_h$ be the linear finite element space defined by

$$X_h := \{ v \in C^0(\bar{\Omega}) | v|_T \in K^1(T) \forall T \in T \} = \text{span}\{\phi_1, \ldots, \phi_{d_h}\}.$$  

(4.11)

where $\phi_i$ is the nodal Lagrange basis. This space does not contain any Dirichlet boundary data. The subsets with the proper boundary values are $X_\varphi$ and $X_s$, analogously to $H_\varphi(\Omega)$ and $H_s(\Omega)$, respectively. Denote by $N$ the set of nodes of the triangulation $T$ and $D_\varphi$ the set of boundary nodes. With this notation the spaces $X_{\varphi,0}$ and $X_{\varphi}$ can be expressed as

$$X_{\varphi,0} := \left\{ \sum_{i \in N \setminus D_\varphi} v_i \phi_i \Big| v_i \in \mathbb{R} \right\},$$

(4.12a)

$$X_{\varphi} := \left\{ \sum_{i \in N \setminus D_\varphi} v_i \phi_i + \sum_{j \in D_\varphi} v_j \phi_j \Big| v_i \in \mathbb{R} \text{ and } v_j = \varphi_D(x_j) \right\},$$

(4.12b)

where $x_j$ is the coordinate of node $j$. The sets $X_s$ and $X_{s,0}$ can be expressed in the same manner, except the set $D_\varphi$ is replaced by a set $D_s$ which contains the boundary nodes lying in $\Gamma_{in}$ and of course the values of the coefficients $v_j$ are computed from $n_{D,s}(x_j)$. In fact we have $D_{s1} = D_{s2}$ and consequently $X_{s1,0} = X_{s2,0}$ for all $s1, s2 \in \{1, \ldots, n_{\text{spec}}\}$.

For given $u^{k+1} \in (H^1(\Omega))^d$ and $n_{s}^k, n_{s}^{k-1} \in X_s$
Find \((\varphi^{k+1}, n_1^{k+1}, \ldots, n_{n_{\text{spec}}}^{k+1}) \in X_{\varphi}(\Omega) \times X_1(\Omega) \times \cdots \times X_{n_{\text{spec}}}(\Omega)\) such that for all \((v_\varphi, v_1, \ldots, v_{n_{\text{spec}}}) \in X_{\varphi,0}(\Omega) \times X_{1,0}(\Omega) \times \cdots \times X_{n_{\text{spec}},0}(\Omega)\)

\[
(\nabla \varphi^{k+1}, \nabla v_\varphi) - R^c \sum_{s=1}^{n_{\text{spec}}} z_s (n_s^{k+1}, v_\varphi) + \sum_{s=1}^{n_{\text{spec}}} (n_s^{k+1} - \frac{4}{3} n_s^k + \frac{1}{3} n_s^{k-1}, v_s) + \gamma \sum_{s=1}^{n_{\text{spec}}} S_s^c (\nabla n_s^{k+1}, \nabla v_s) + \gamma \sum_{s=1}^{n_{\text{spec}}} K_s^c z_s (n_s^{k+1}, \nabla \varphi^{k+1}, \nabla v_s) + \gamma (u^{k+1} \cdot \nabla n_s^{k+1}, v_s) = 0.
\]

(4.13)

Since \(X_h\) is a finite dimensional space with the basis \(\phi_i\), problem (4.13) is equivalent to

Find \((\vec{\varphi}^{k+1}, \vec{n}_1^{k+1}, \ldots, \vec{n}_{n_{\text{spec}}}^{k+1}) \in \mathbb{R}^{(n_{\text{spec}}+1)d_h}\) such that

\[
(\nabla \varphi^{k+1}, \nabla \phi_i) - R^c \sum_{s=1}^{n_{\text{spec}}} z_s (n_s^{k+1}, \phi_i) = 0 \quad \text{for } i \in \mathcal{N}\setminus\mathcal{D}_\varphi,
\]

\[
\varphi_i - \varphi_D(x_i) = 0 \quad \text{for } i \in \mathcal{D}_\varphi,
\]

\[
(n_s^{k+1} - \frac{4}{3} n_s^k + \frac{1}{3} n_s^{k-1}, \phi_i) + \gamma S_s^c (\nabla n_s^{k+1}, \nabla \phi_i) + \gamma K_s^c z_s (n_s^{k+1}, \nabla \phi^{k+1}, \nabla \phi_i) + \gamma (u^{k+1} \cdot \nabla n_s^{k+1}, \phi_i) = 0 \quad \text{for } i \in \mathcal{N}\setminus\mathcal{D}_s,
\]

\[
n_{s_i} - n_{D,s}(x_i) = 0 \quad \text{for } i \in \mathcal{D}_s,
\]

\[
s = 1, \ldots, n_{\text{spec}},
\]

(4.14)

where \(\vec{\varphi} = [\varphi_1, \ldots, \varphi_{d_h}]^T\) and \(\varphi = \sum_{j=1}^{n} \varphi_j \phi_j\) and analogously for \(n_s^{k+1}\). This discrete problem is equivalent to: find the zero of the nonlinear vector valued function

\[
g(\vec{\varphi}, \vec{n}_1, \ldots, \vec{n}_{n_{\text{spec}}}) : \mathbb{R}^{(n_{\text{spec}}+1)d_h} \to \mathbb{R}^{(n_{\text{spec}}+1)d_h},
\]

(4.15)

defined by Eq. (4.14). We apply Newton’s method to this function

\[
J_g(x^\ell)(x^{\ell+1} - x^\ell) = -g(x^\ell) \Leftrightarrow J_g(x^\ell)x^{\ell+1} = -g(x^\ell) + J_g(x^\ell)x^\ell =: G(x^\ell).
\]

(4.16)

In practice, three to four iterations are sufficient in every time step. The Jacobian of the function \(g\) is (example for two ion species)
\[ J_g(\vec{\varphi}, \vec{n}_1, \vec{n}_2) = \begin{bmatrix} S & -R^c z_1 M & -R^c z_2 M \\ \gamma K' z_1 S[n_1] & M + \gamma(S'[z + K' z_1 \{\nabla \varphi\}] + K'[\vec{u}]) & 0 \\ \gamma K' z_2 S[n_2] & 0 & M + \gamma(S'[z + K' z_2 \{\nabla \varphi\}] + K'[\vec{u}]) \end{bmatrix}. \] 

(4.17)

The five different submatrices resulting from the finite element space discretization are

\[
M_{i,j} := (\phi_j, \phi_i), \quad S_{i,j} := (\nabla \phi_j, \nabla \phi_i), \quad K_{i,j} := (\nabla \phi_j, \nabla \phi_i), \quad S[n]_{i,j} := (n \nabla \phi_j, \nabla \phi_i), \quad K'[\vec{u}]_{i,j} := (\vec{u} \cdot \nabla \phi_j, \phi_i),
\]

(4.18a-c)

where \( M \) is the standard mass matrix and \( S \) the standard stiffness matrix. The respectively three other matrices are a weighted stiffness matrix and two convection matrices. The matrices are defined on the space \( X_h \) i.e. for all \( i, j \in \{1, \ldots, d_h\} \).

According to (4.14) the Dirichlet boundary conditions are included in \( J_g \) by replacing the rows corresponding to the indices \( D_{\varphi} \) and \( D_s \) through unit vectors and setting the right hand side vector to the related Dirichlet value for this indices.

With the solutions given at time \( t^k \) and \( t^{k-1} \) the solution at time \( t^{k+1} \) is calculated by the following procedure.

**Newton's method for the NPP–System**

- Set \( \varphi^{k,0} = \varphi^k, \quad \vec{n}_{1,0} = \vec{n}_1^k, \quad \vec{n}_{2,0} = \vec{n}_2^k \).
- By using GMRes, solve

\[
J_g(\varphi^{k,\ell}, \vec{n}_1^{k,\ell}, \vec{n}_2^{k,\ell}) \begin{bmatrix} \varphi^{k,\ell+1} \\ n_1^{k,\ell+1} \\ n_2^{k,\ell+1} \end{bmatrix} = -G(\varphi^{k,\ell}, \vec{n}_1^{k,\ell}, \vec{n}_2^{k,\ell}),
\]

(4.19)

until

\[
\text{max} \left( \left\| \begin{bmatrix} \varphi^{k,\ell+1} - \varphi^{k,\ell} \\ n_1^{k,\ell+1} - n_1^{k,\ell} \\ n_2^{k,\ell+1} - n_2^{k,\ell} \end{bmatrix} \right\|_2, \| g(\varphi^{k,\ell}, \vec{n}_1^{k,\ell}, \vec{n}_2^{k,\ell}) \|_2 \right) \leq \text{tol}.
\]

(4.20)
Set $\vec{\varphi}^{k+1} = \vec{\varphi}^{k,\ell+1}$, $\vec{n}_1^{k+1} = \vec{n}_1^{k,\ell+1}$ and $\vec{n}_2^{k+1} = \vec{n}_2^{k,\ell+1}$.

The function $G$ is defined by

$$G(\vec{\varphi}, \vec{n}_1, \vec{n}_2) := \begin{bmatrix} 0 \\ M \left( \frac{1}{3}n_1^{k-1} - \frac{4}{3}n_1^{k} \right) - \gamma K_1 c_1 z_1 S[n_1] \varphi \\ M \left( \frac{1}{3}n_2^{k-1} - \frac{4}{3}n_2^{k} \right) - \gamma K_2 c_2 z_2 S[n_2] \varphi \end{bmatrix}. \quad (4.21)$$

As proposed in the common literature a mixed stopping criterion is used. This procedure has been implemented in NAVIER, see Section 3.4.

### 4.4 Stabilization

In real applications, the coefficient $S^c$ is much smaller than $K^c$ which in turn is much smaller than 1, $S^c \ll K^c \ll 1$. That means the Nernst–Planck equation is convection dominated. This could cause huge unphysical oscillations in the numerical solution obtained with standard finite element methods. For the standard convection–diffusion equation

$$-S^c \Delta \varphi + \vec{u} \cdot \nabla \varphi = f \quad (4.22)$$

only a weak error bound of the form

$$\|\nabla (\varphi - \varphi_h)\| \leq \frac{\|\vec{u}\|}{S^c} Ch \quad (4.23)$$

can be obtained. The fact that the constant $\|u\|/S^c$ is very large suggests only a poor approximation for the standard Galerkin discretization. This loss of control with respect to the gradients results in oscillatory behavior of the solution, see [23]. Many methods have been proposed to circumvent this defect. The most popular techniques are streamline diffusion combined with shock capturing [10, 6], subgrid scale stabilization [38, 32] and Galerkin least square methods [27, 18]. These methods modify the discrete equation in a certain way such that oscillations are reduced. Error estimates in special norms can be derived for all these methods. A common drawback of this methods is the need of a user–chosen stabilization parameter. The Flux Corrected Transport (FCT) [7, 60] method, however, acts on an algebraic level and is directly applied to the linear system of equations. Instability phenomena often occur with the loss of M–matrix properties in the linear system. This property is enforced by the FCT algorithms where it is needed. For a detailed overview, see [43].
We use the linear version of the FCT algorithm and apply it to the Nernst–Planck part of the Jacobian $J_g$, i.e.
\[ M + \tau \frac{2}{3} \left( S_c^c S + K_c^c z_a K [\nabla \varphi] + K'[u] \right). \] (4.24)

The stabilization process is applied independently for each species. Therefore we skip the index $s$ and to fit the notation in [43] we abbreviate

\[ K^\ell := -\frac{2}{3} \left( S_c^c S + K_c^c z_a K [\nabla \varphi^\ell] + K'[u] \right) \] (4.25)

for an arbitrary $s$. In this application the results changed only marginally when the diffusion operator $S_c^c S$ was part of $K^\ell$ or not. The version presented here requires less storage. We apply the following procedure per time step:

**Linear FCT Algorithm**

1. Compute artificial diffusion operator $D^\ell$, the lumped mass matrix $M^L$ and the low order approximation $L^\ell$ in every step $\ell$ of Newton’s method (4.19)–(4.21) with $M$ replaced by $M^L$ and $K^\ell$ replaced by $L^\ell$ in the Jacobian and additionally $M$ replaced by $M^L$ on the right hand side.

\[ D^\ell_{ij} = D^\ell_{ji} = \max\{-K^\ell_{ij}, 0, -K^\ell_{ji}\} \] (4.26a)
\[ D^\ell_{ii} = -\sum_{j \neq i} D^\ell_{ij} \] (4.26b)
\[ M^L_{ii} = \sum_{j} M_{ij} \] (4.26c)
\[ L^\ell = K^\ell + D^\ell \] (4.26d)

At the end we obtain the solution $\varphi^{k+1}$ and the low order solution $n^L$.

2. Compute approximate time derivatives $\dot{n}$ for the prediction of the fluxes.

\[ \dot{n}^L = \frac{n^L - n^k}{\tau}. \] (4.27)

3. Determine antidiffusive flux matrix $f$

\[ f_{ij} = M_{ij}(\dot{n}^L_i - \dot{n}^L_j) + D^L_{ij} (n^L_i - n^L_j) \] (4.28)

4. Apply the multidimensional Zalesak [60] limiter to the antidiffusive flux matrix $f$. 

• Obtain the sums of positive and negative antidiffusive fluxes into node $i$.

$$P_i^+ = \sum_{j \neq i} \max\{0, f_{ij}\}, \quad P_i^- = \sum_{j \neq i} \min\{0, f_{ij}\}. \quad (4.29)$$

• Compute the distance to a local extrema of the low order solution.

$$Q_i^+ = \max\{0, \max_{j \neq i}(n_j^L - n_i^L)\}, \quad Q_i^- = \min\{0, \min_{j \neq i}(n_j^L - n_i^L)\}. \quad (4.30)$$

• Determine the nodal correction factors for the net increment to node $i$.

$$R_i^+ = \min\left\{1, \frac{M_i^L Q_i^+}{\tau P_i^+}\right\}, \quad R_i^- = \min\left\{1, \frac{M_i^L Q_i^-}{\tau P_i^-}\right\}. \quad (4.31)$$

To avoid unnecessary limiting on $\partial \Omega$ set $R^+ = 1$ and $R^- = 1$ at boundary nodes.

• Limit the raw antidiffusive fluxes $f_{ij}$ and $f_{ji}$ in a symmetric fashion.

$$\alpha_{ij} = \begin{cases} \min\{R_i^+, R_j^-\}, & \text{if } f_{ij} > 0, \\
\min\{R_i^-, R_j^+\}, & \text{otherwise.} \end{cases} \quad (4.32)$$

• Limit the antidiffusive fluxes.

$$\bar{f}_i = \sum_{j \neq i} \alpha_{ij} f_{ij} \quad (4.33)$$

5. Correct the low order solution.

$$M_i^L n_i^{k+1} = M_i^L n_i^L + \tau \bar{f} \quad (4.34)$$

### 4.5 Validation

In this section, we present numerical results for two test problems.

#### 4.5.1 Weakly Coupled NPP Equations

One ion species is considered in this example. However, the characteristic concentration $n_c$ is so small that the Poisson equation is barely influenced by it ($R_c \ll 1$). The
external velocity $u$ is set to zero. We consider an unit cube with periodic boundary conditions in two directions, resulting in a quasi one dimensional problem. For $R^e = 0$ a constant potential difference is applied as boundary condition for the Poisson equation, resulting in a linear potential. Then the exact stationary solution of the Nernst–Planck equation is

$$n = \frac{a}{1 - \exp(-a)} \exp(-ax),$$

(4.35)

with $a = \frac{eU_i \nabla \phi}{k_B T}$. This stationary solution is an exponential function which only depends on the potential difference. The results for $R^e = 0$ are compared qualitatively, in particular the stability properties of the solution with and without using FCT, see Figs. 4.1–4.4. The Galerkin solution deteriorates completely if the boundary layer becomes too small. This results in an oscillatory solution. In the case of a strongly coupled system, these oscillations would have an impact on the electric potential $\phi$ which in return has an effect on the concentration. This coupling would lead to completely unphysical solutions in a few time steps. In contrast, the solution with FCT remains stable in any cases which guarantees a stable solution of the strongly coupled system. As a consequence of mass conservation, the maximum value of the FCT solution must be smaller than the actual maximum value if the boundary layer is not resolved anymore. The philosophy here is then to resolve the boundary layers where possible. If this is not possible, then we would rather have a stable solution with a smaller maximum value than a completely unstable solution yielding unphysical solution behavior and which also diverges after some time steps.

### 4.5.2 Strongly Coupled NPP Equations

If the assumption $R^e \ll 1$ does not hold anymore, both equations are strongly coupled, and it is much more difficult to find explicit nontrivial solutions of the NPP equations. In the stationary case and $u = 0$ solutions of the Nernst–Planck equations have the form $n = \exp(b \phi)$ for $b = \frac{e}{k_B T}$. By applying this formula to the Poisson equation, we receive the Boltzmann–Poisson equation

$$-\Delta \phi = -a \exp(b \phi),$$

(4.36)

for a single negative loaded ion species, constant $\varepsilon_r$ and $a = \frac{e}{\varepsilon_0 \varepsilon_r}$. If the size of the domain is set to $1/b$ and the initial ion concentration is $n(t = 0) = \frac{2b \tan(1)}{a}$, then
Figure 4.1: Ion concentration over capacitor length with mesh width $h = 1/16$ and $a = 1$. All solutions agree very well. The version with FCT has a slightly smaller gradient but the mass is conserved.

This simple expression is a stationary solution of the NPP equations

$$\varphi = -\frac{1}{b} \ln \left( \frac{a}{2b} \cos(bx)^2 \right),$$

(4.37)

$$n = \frac{2b}{a \cos(bx)^2}.$$  

(4.38)

Table 4.1 shows the errors produced by the method. For half the mesh width, the error is reduced by a factor of four, which indicates $h^2$ convergence in the $L^2$-norm.

| $h$  | $||\varphi_h - \varphi||_2$ | EOC | $||n_h - n||_2$ | EOC |
|------|-----------------|-----|---------------|-----|
| 1/10 | 0.064710        | -   | 0.1419545     | -   |
| 1/20 | 0.015207        | 2.089 | 0.0374622 | 1.922 |
| 1/40 | 0.003584        | 2.085 | 0.0091357 | 2.036 |

Table 4.1: Errors of the stationary solution of the strongly coupled NPP equations test case.
Figure 4.2: Ion concentration over capacitor length with mesh width \( h = 1/16 \) and \( a = 6 \). In this case the numerical solution without FCT is still identical with the expected solution. The version with FCT has a slightly lesser maximum value.
Figure 4.3: Ion concentration over capacitor length with mesh width $h = 1/16$ and $a = 36$. In this case the standard Galerkin solution still has the same maximum value as the analytic solution. Unfortunately, a large undershoot occurs near the left boundary. The maximum value of the FCT solution is quite off whereas the solution shows neither undershoots nor wiggles.
Figure 4.4: Ion concentration over capacitor length with mesh width $h = 1/16$ and $a = 216$. The analytic solution develops a sharp boundary layer whose size is much smaller than the mesh width. The classical solution shows huge oscillations. The FCT solution remains stable.
Chapter 5

Electrodynamical Particulate Flows (PFNPP)

In numerous applications for particulate flows, especially for small scales (nano–particles), electrodynamics have to be considered. The reasons are charged particles, ion laden fluids and external electric fields which may have desired or undesired effects on a particulate flow. The work presented here was developed as a part of the Bayerische Forschungsstiftung project “Digitaldruck: Dünnere Farbschichten”. The aim of the project was to develop a new digital printing system which uses very thin color layers to reduce the printing costs. This new printing system is based on liquid toner, where toner particles are immersed in a carrier fluid. This fluid contains additionally charge carrier agents (CCA), charged ions which influences the behavior of the charged toner particles. Simulations are important for a physical insight into the processes involved and serve as a reliable tool in a rapid development.

5.1 Model

A model for the description of such a situation is the Particulate–Flows–Nernst–Planck–Poisson model (PFNPP), [40]. The toner particles in the fluid are represented by the PF model, while the CCA is described by the NPP model. With the same notation as in the previous chapters the dimensionless model reads

\[
\begin{align*}
\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} - \nabla \cdot \left( \frac{1}{\text{Re}} \nabla \phi - p^\parallel \right) &= -M^e \nabla \phi \sum_{s=1}^{n_{spec}} z_s n_s \quad \text{in } \Omega(t), \\
\nabla \cdot \mathbf{u} &= 0 \quad \text{in } \Omega(t),
\end{align*}
\]

(5.1a)  
(5.1b)
\[ \partial_t n_s - S^c_s \Delta n_s - K^c_s z_s \nabla \cdot (n_s \nabla \varphi) + \mathbf{u} \cdot \nabla n_s = 0 \quad s = 1, \ldots, n_{\text{spec}} \quad \text{in } \Omega(t), \]  
\[ -\nabla \cdot (\varepsilon_r \nabla \varphi) - R^c \sum_{s=1}^{n_{\text{spec}}} z_s n_s = 0 \quad \text{in } \Omega_c, \]
\[ \frac{d}{dt}(M\mathbf{U}) = \mathbf{F} + \int_{\partial \Pi(t)} \sigma n ds - Z Q_c^u \int_{\Omega(t)} \nabla \varphi \rho_V dx, \]
\[ \frac{d}{dt}(I\omega) = \int_{\partial \Pi(t)} \mathbf{r} \times (\sigma n) ds - Z Q_c^u \int_{\Omega(t)} \mathbf{r} \times \nabla \varphi \rho_V dx, \]
\[ \frac{d}{dt} \Theta = R[\omega] \Theta. \]

The two new dimensionless numbers are defined as \( M^c := \frac{e n c U_c}{\mu s} \) and \( Q^c := \frac{e U_c}{\mu s^2 \xi_s^2} \). \( Z \) is the valence of the particle and the unit distribution function \( \rho_V \) is positive with \( \int_{\Pi} \rho_V dx = 1 \). This function models different kinds of charge distributions like constant charges, surface charges or more sophisticated ones. The boundary conditions are described by

\[ \mathbf{u} = \mathbf{u}_D \quad \text{on } \Gamma_D, \]  
\[ \mathbf{u} = \mathbf{U} + \omega \times \mathbf{r} \quad \text{on } \partial \Pi(t), \]  
\[ n_s = n_{D,s} \quad \text{on } \Gamma_m \subset \Gamma_D, \]  
\[ -(K^c_s z_s n_s \nabla \varphi + S^c_s \nabla n_s) \cdot \mathbf{n} = 0 \quad \text{on } \partial \Omega(t) \setminus \Gamma_m. \]

The system is closed by appropriate initial conditions.

**5.2 Numerical Method**

For the numerical solution of this model, we use an explicit decoupling in time. That means, for time step \( t^{k+1} \) we solve the Nernst–Planck–Poisson equations with the domain \( \Omega \) and the velocity \( \mathbf{u} \) given at time step \( t^k \). Then the Particle–Navier–Stokes equations are solved for time step \( t^{k+1} \), using the data obtained from the solution of the Nernst–Planck–Poisson equations. This approach enables us to use the solution methods presented in Chapters 3 & 4. To find an approximate solution of Eqs. (5.1a)–(5.1h), proceed as follows
Solution of the PFNPP model

**Step 1** (Nernst–Planck–Poisson)

Given \(n^k_s, n^{k-1}_s\) and \(u^k\) use Newton’s method as described in Chapter 4 to find a solution of the nonlinear equation

\[
\begin{align*}
(\nabla \phi^{k+1}, \nabla v_\phi) - Rc \sum_{s=1}^{n_{\text{spec}}} z_s (n^{k+1}_s, v_\phi) + & 
\sum_{s=1}^{n_{\text{spec}}} (n^{k+1}_s - \frac{4}{3} n^k_s + \frac{1}{3} n^{k-1}_s, v_s) + \gamma \sum_{s=1}^{n_{\text{spec}}} S^c_s (\nabla n^{k+1}_s, \nabla v_s) + \\
\gamma \sum_{s=1}^{n_{\text{spec}}} K^c_s z_s (n^{k+1}_s \nabla \phi^{k+1}, \nabla v_s) + \gamma \sum_{s=1}^{n_{\text{spec}}} (u^{k+1} \cdot \nabla n^{k+1}_s, v_s) = 0.
\end{align*}
\]

(5.3)

**Step 2** (Predictor)

Given \(F, X^k\) and \(U^k\) use Newton’s method as outlined in Subsection 3.6.1 to solve

\[
\begin{align*}
\tilde{X} &= X^k + \frac{\tau}{2} \left( U^k + \tilde{U} \right), \\
\tilde{U} &= U^k + \frac{\tau}{M} F \left( \frac{X^k + \tilde{X}}{2} \right).
\end{align*}
\]

(5.4, 5.5)

**Step 3** (Burgers problem)

Given \(u^k, u^{k-1}, p^k, \chi^k, \chi^{k-1}, \tilde{U}, \omega^k\).

Find \(\tilde{u}^{k+1} \in X\) such that for all \(\tilde{v} \in X\)

\[
a(P\tilde{u}^{k+1}, P\tilde{v}) = (f, P\tilde{v}),
\]

(5.6)

and set \((u^{k+1}, U^{k+1}, \omega^{k+1}) = u^{k+1} = P\tilde{u}^{k+1}\).

**Step 4** (Computation of the pressure correction)

Find \(\chi^{k+1} \in Y\) such that for all \(\Psi \in Y\)

\[
m(\nabla \chi^{k+1}, \nabla \Psi) = \frac{1}{\gamma} b(\Psi, u^{k+1}).
\]

(5.7)
Step 5 (Pressure update in rotational form)

Find \( p^{k+1} \in Y \) such that for all \( q \in Y \)

\[
m(p^{k+1}, q) = m(p^{k} + \chi^{k+1}, q) - b(q, \frac{2}{\text{Re}} u^{k+1}).
\]  

(5.8)

Step 6 (Corrector)

Given \( \Theta^k, X^k, \omega^k, \omega^{k+1}, U^k \) and \( U^{k+1} \).

\[
\Theta^{k+1} = \left( I - \frac{\tau}{2} \mathcal{R}[\omega^{k+1}] \right)^{-1} \left( I + \frac{\tau}{2} \mathcal{R}[\omega^{k}] \right) \Theta^k, \]  

(5.9a)

\[
X^{k+1} = X^k + \frac{\tau}{2} \left( U^k + U^{k+1} \right).
\]  

(5.9b)

5.3 Validation

5.3.1 A Single Particle in a Capacitor

The motion of a single particle in a plate capacitor filled with a fluid can be described by the model presented in the first section of this chapter. For a sketch of the situation, see Fig. 5.1. Hence, we denote by \( d_C \) the plate distance and \( R \) the radius of the spherical particle. Periodic boundary conditions are set in the \( x_1 \) and \( x_3 \) directions. We apply homogeneous Dirichlet boundary conditions (D.b.c.) for the potential \( \varphi \) at the bottom plane and a nonhomogeneous, time-dependent D.b.c. \( V(t) \) at the upper plane. As usual, the fluid viscosity is denoted by \( \eta \), \( e \) is the unit charge and \( Z \) is the charge number. We are mainly interested in the particle motion in \( x_2 \)-direction. Alternatively, the same problem can also be described approximately
by the following system of ODE’s

\[ \dot{U} = \left( \frac{eZV(t)}{dC} - 6\pi\eta RU \right)/M, \quad (5.10) \]
\[ \dot{X} = U. \quad (5.11) \]

The first term on the right hand side of Eq. (5.10) is the force applied by the external electric field. Assuming that the solution of the Poisson equation is linear, the electric field is constant and the electric force on a homogeneous charged particle is calculated with the given formula. The second term is Stokes law for the frictional force of a spherical particle in an infinite viscous fluid. This law is valid for small Reynolds numbers and replaces the PF model. Additionally \( M \) is the mass of the particle.

We set the applied voltage \( V(t) \) to be of the form \( V(t) = V_{\text{max}} \cos(2\pi f) \) where \( f \) is the frequency. Then for the trivial initial conditions \( U(0) = 0 \) and \( X(0) = 0 \) the analytical solution is given by

\[ X(t) = \frac{ce^{-bt}}{a^2 + b^2} + \frac{b^2 c \sin(at) - abc \cos(at)}{ab(a^2 + b^2)}, \quad (5.12) \]

with \( a = 2\pi f, \ b = 6\pi\eta R/M \) and \( c = \frac{eZV_{\text{max}}}{d_c M}. \) We compare this solution with the computations obtained from the implementation of the solution procedure described in the previous section, see Fig. 5.2 and Fig. 5.3. In this experiment the frequency is 30000Hz and the applied potential difference is 290V. The plate distance is 0.02mm, the particle diameter is 0.002mm and its density \( \rho = 1200 \text{kg/m}^3 \). With the viscosity \( \eta = 2.8 \cdot 10^{-3} \text{Ns/m}^2 \) and the fluid density \( \rho = 784 \text{kg/m}^3 \) the Reynolds number is \( \text{Re} = 1.342 \cdot 10^{-4} \). The characteristic ion concentration is \( n_c = 0 \) as no ions are present in this simulation. The numerical results are in good agreement with the simplified analytical model.
Figure 5.2: Magnitude of the electric force acting on the particle over time. The numerical solution obtained from the term $-eZ \int_\Pi \nabla \varphi \rho V \, dx$ is identical with the analytic term $\frac{eZV_{\text{max}}}{d_C} \cos(2\pi ft)$.

Figure 5.3: Particle displacement over time. The frequency as well as the phase are identical for both solutions. The amplitudes differ a little bit, as in the full model wall effects are included, resulting in a smaller displacement.
Chapter 6

Conclusions & Outlook

6.1 Conclusions

A novel finite element method for the simulation of particulate flows is presented. Its key ingredients are the one domain approach, splitting in time, subspace projection method to account for the rigid body motion within the particles and time dependent adaptively refined meshes. The advantages of the method are its easy implementation and its efficiency. Only few modifications are needed to extend an existing Navier–Stokes code to simulate particulate flows by this method. An error bound is given quantifying the error introduced by not exactly resolving the particle boundary. To overcome this limitation, adaptive finite elements are used. The code is validated using two different test cases. The results agree well with experimental and theoretical findings.

With the BDF2 method for time discretization and finite elements combined with flux corrected transport for the spatial discretization it is possible to derive an efficient and stable scheme for the solution of the Nernst–Planck–Poisson equations. Its implementation is verified with stationary test cases.

Combined with the NPP model, the PF model can be used to efficiently simulate particulate electrodynamic flows.

6.2 Outlook

The error estimate in Section 3.3 shows a $O(h^{\frac{1}{2}})$ energy error for the discrete Burgers problem. Numerical experiments suggest that this bound can be improved to $O(h)$ or even $O(h^2)$ if the mesh is aligned to the particle boundary. The first bound can be achieved if the simplices are aligned to the boundary using a linear transformation,
the second bound uses isoparametric deformed simplices for the alignment. A fast and reliable grid deformation algorithm which is able to align simplices exactly to the boundary is therefore needed. A promising approach can be found in [4]. Local refinement still proves to be important to resolve boundary layers near the particle surface.

The splitting in the time domain presented in Section 3.2.1 is of accuracy $O(\tau)$. In cases where the external force $\mathbf{F}$ is very large compared to the fluid force $\int_{\partial \Pi(t)} \sigma \mathbf{n} \, ds$, very small time steps are required to resolve the correct solution. On the other hand, a monolithic approach would involve either Picard’s iteration or Newton’s method to solve the nonlinear problems arising from a stable and therefore implicit time discretization of the full system (3.19a)–(3.19d). For Picard’s iteration it is not clear how fast it would converge, resulting in possibly huge computational costs. Newton’s method needs information on the gradient of the implicit problem which involves the mesh change if the particle position changes. This information is not at hand, so approximations to the gradient have to be used which can cause Newton’s method to converge only linearly which in turn results in huge computational costs. A different approach is to use straightforward higher order operator splitting methods for the decoupling of the principal unknowns. With the auxiliary variable $u = [\mathbf{u}, p, \mathbf{U}, \omega, \mathbf{X}, \Theta]^T$ Eqs. (3.19a)–(3.19d) can be rewritten as

$$M(t) \frac{d}{dt} u = A(u, t),$$

with $A$ a nonlinear operator. Higher order operator splitting methods require $M(t)$ to be constant and invertible. None of this is the case here which discards this option. ROW–methods [35] are a convenient way to obtain a stable higher order time discretization with only linear subproblems. These linear subproblems are defined by the Jacobian of the origin problem which are unfortunately not available as discussed before. W–methods can circumvent the explicit calculation of the full Jacobian and are able to use an approximation instead. This preserves the order of the method but can cause problems with the stability. For example if the Jacobian is approximated by the identity, the method will convert into an explicit Runge–Kutta time stepping with poor stability properties. In summary it is not clear how to receive a higher order approximation in time without very huge computational costs.
Appendix A

Appendix

In this section a detailed step by step calculation is performed for equation (3.16). First we observe

\[ \int_{\Pi(t)} r \, dx = \int_{\Pi(t)} x - X \, dx = \int_{\Pi(t)} x \, dx - X \int_{\Pi(t)} 1 \, dx \]

\[ = \int_{\Pi(t)} x \, dx - \frac{||\Pi(t)||}{||\Pi(t)||} \int_{\Pi(t)} x \, dx = 0. \]  

(A.1)

The total time derivative of the velocity is

\[ \frac{d}{dt} u = \partial_t u + (u \cdot \nabla) u. \]  

(A.2)

In addition we have \( u = U + \omega \times r \) inside the particle which results in

\[ \frac{d}{dt} u = \dot{U} + \dot{\omega} \times r + \omega \times (\omega \times r). \]  

(A.3)

Together it follows that

\[ \partial_t u + (u \cdot \nabla) u = \dot{U} + \dot{\omega} \times r + \omega \times (\omega \times r) \quad \text{in } \Pi(t). \]  

(A.4)
Multiplying Eq. (3.15) by $v$ and $v = V + \xi \times r$ and integrating on $\Pi(t)$ leads to

$$
\int_{\Pi(t)} (\partial_t u + (u \cdot \nabla) u) \cdot v \, dx =
\int_{\Pi(t)} \dot{U} \cdot v \, dx + \int_{\Pi(t)} (\dot{\omega} \times r) \cdot v \, dx +
\int_{\Pi(t)} (\omega \times (\omega \times r)) \cdot V \, dx =
\int_{\Pi(t)} \dot{U} \cdot V \, dx + \int_{\Pi(t)} \dot{U} \cdot (\xi \times r) \, dx +
\int_{\Pi(t)} (\dot{\omega} \times r) \cdot V \, dx + \int_{\Pi(t)} (\omega \times (\omega \times r)) \cdot (\xi \times r) \, dx
$$

(A.5)

With Eq. (A.1) and the identities

$$a \cdot (b \times c) = b \cdot (c \times a) = c \cdot (a \times b), \quad (A.6)$$

$$(a \times b) \cdot (c \times d) = (a \cdot c) (b \cdot d) + (b \cdot c) (a \cdot d), \quad (A.7)$$

$$(a \otimes a) \cdot b = (a \cdot b) a. \quad (A.8)$$

for $a, b, c, d \in \mathbb{R}^3$ we calculate the result for each of the six terms. Recall the definition of the dimensionless mass and inertia

$$M = \frac{1}{\alpha} \int_{\Pi(t)} 1 \, dx, \quad (A.9)$$

$$I = \frac{1}{\alpha} \int_{\Pi(t)} (r \cdot r^\parallel - r \otimes r) \, dx. \quad (A.10)$$

$$I : \int_{\Pi(t)} \dot{U} \cdot V \, dx = \frac{\alpha}{\alpha} \dot{U} \cdot V \int_{\Pi(t)} 1 \, dx = \alpha M \dot{U} \cdot V \quad (A.11)$$

$$II : \int_{\Pi(t)} \dot{U} \cdot (\xi \times r) \, dx = (\dot{U} \times \xi) \cdot \int_{\Pi(t)} r \, dx = 0 \quad (A.12)$$

$$III : \int_{\Pi(t)} (\dot{\omega} \times r) \cdot V \, dx = (V \times \dot{\omega}) \cdot \int_{\Pi(t)} r \, dx = 0 \quad (A.13)$$
\[ IV: \int_{\Pi(t)} (\dot{\omega} \times \mathbf{r}) \cdot (\xi \times \mathbf{r}) \, d\mathbf{x} = \int_{\Pi(t)} (\dot{\omega} \cdot \xi) (\mathbf{r} \cdot \mathbf{r}) + (\mathbf{r} \cdot \xi) (\dot{\omega} \cdot \mathbf{r}) \, d\mathbf{x} = \]
\[ \int_{\Pi(t)} (\mathbf{r} \cdot \mathbf{r}) \dot{\omega} + (\dot{\omega} \cdot \mathbf{r}) \mathbf{r} \, d\mathbf{x} \cdot \xi = \int_{\Pi(t)} (\mathbf{r} \cdot \mathbf{r}) \dot{\omega} + (\mathbf{r} \otimes \mathbf{r}) \dot{\omega} \, d\mathbf{x} \cdot \xi = \]
\[ \left( \int_{\Pi(t)} (\mathbf{r} \cdot \mathbf{r}I + \mathbf{r} \otimes \mathbf{r}) \, d\mathbf{x} \right) \cdot \xi = \alpha I \dot{\omega} \cdot \xi \quad (A.14) \]

\[ V: \int_{\Pi(t)} (\omega \times (\omega \times \mathbf{r})) \cdot \mathbf{V} \, d\mathbf{x} = \int_{\Pi(t)} (\omega \times \mathbf{r}) \cdot (\mathbf{V} \times \omega) \, d\mathbf{x} = \]
\[ \int_{\Pi(t)} (\omega \cdot \mathbf{V}) (\mathbf{r} \cdot \omega) + (\mathbf{r} \cdot \mathbf{V}) (\omega \cdot \omega) \, d\mathbf{x} = ((\omega \cdot \mathbf{V}) \omega + (\omega \cdot \omega) \mathbf{V}) \cdot \int_{\Pi(t)} \mathbf{r} \, d\mathbf{x} = 0 \]
\[ (A.15) \]

\[ VI: \int_{\Pi(t)} (\omega \times (\omega \times \mathbf{r})) \cdot (\xi \times \mathbf{r}) \, d\mathbf{x} = \]
\[ \int_{\Pi(t)} (\omega \cdot \xi) \left( (\omega \times \mathbf{r}) \cdot \mathbf{r} \right) + \left( (\omega \times \mathbf{r}) \cdot \xi \right) (\omega \cdot \mathbf{r}) \, d\mathbf{x} = \]
\[ \int_{\Pi(t)} (\omega \cdot \mathbf{r}) \omega + (\omega \cdot \mathbf{r}) (\omega \cdot \mathbf{r}) \, d\mathbf{x} \cdot \xi = \int_{\Pi(t)} 0 + \omega \times [(\omega \cdot \mathbf{r}) \mathbf{r}] \, d\mathbf{x} \cdot \xi = \]
\[ \int_{\Pi(t)} \omega \times [(\mathbf{r} \cdot \mathbf{r}) \omega] + \omega \times [(\omega \cdot \mathbf{r}) \mathbf{r}] \, d\mathbf{x} \cdot \xi = \omega \times \left( \int_{\Pi(t)} [(\mathbf{r} \cdot \mathbf{r}) \omega] + [(\mathbf{r} \otimes \mathbf{r}) \omega] \, d\mathbf{x} \right) \cdot \xi = \]
\[ \omega \times \left( \int_{\Pi(t)} (\mathbf{r} \cdot \mathbf{r}I + \mathbf{r} \otimes \mathbf{r}) \, d\mathbf{x} \omega \right) \cdot \xi = \alpha \omega \times (I \omega) \cdot \xi \quad (A.16) \]

Finally we arrive at
\[ \int_{\Pi(t)} (\partial_t \mathbf{U} + (\mathbf{u} \cdot \nabla) \mathbf{u}) \cdot \mathbf{v} \, d\mathbf{x} = \alpha M \dot{\mathbf{U}} \cdot \mathbf{V} + \alpha (I \dot{\omega} + \omega \times (I \omega)) \cdot \xi. \quad (A.17) \]
Bibliography


