Journal of Non-Newtonian Fluid Mechanics 322 (2023) 105133

Contents lists available at ScienceDirect

Journal of Non-Newtonian Fluid Mechanics

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

An eigenvalue-free implementation of the log-conformation formulation

Florian Becker [a,*], Katharina Rauthmann [a], Lutz Pauli [b], Philipp Knechtges [a]

[a] German Aerospace Center (DLR), Institute for Software Technology, High-Performance Computing, Cologne, Germany
[b] MAGMA Gießereitechnologie GmbH, Aachen, Germany

ARTICLE INFO

Keywords:
Log-conformation
Oldroyd-B model
Giesekus model
Finite Volume Method

ABSTRACT

The log-conformation formulation, although highly successful, was from the beginning formulated as a partial differential equation that contains an, for PDEs unusual, eigenvalue decomposition of the unknown field. To this day, most numerical implementations have been based on this or a similar eigenvalue decomposition, with Knechtges et al. (2014) being the only notable exception for two-dimensional flows.

In this paper, we present an eigenvalue-free algorithm to compute the constitutive equation of the log-conformation formulation that works for two- and three-dimensional flows. Therefore, we first prove that the challenging terms in the constitutive equations are representable as a matrix function of a slightly modified matrix of the log-conformation field. We give a proof of equivalence of this term to the more common log-conformation formulations. Based on this formulation, we develop an eigenvalue-free algorithm to evaluate this matrix function. The resulting full formulation is first discretized using a finite volume method, and then tested on the confined cylinder and sedimenting sphere benchmarks.

1. Introduction

Since its inception [1], the log-conformation formulation undoubtedly has been a huge success. It had a considerable impact on attacking the High Weissenberg Number Problem (HWNP) that had riddled simulation results the decades before.

The general idea of the log-conformation formulation is simple: The conformation tensor $\mathbf{C}(x,t) \in \mathbb{R}^{d\times d}$, which, for a given instant of space $x \in \mathbb{R}^d$ and time t, essentially encodes a macroscopically averaged covariance of the microscopic configuration, is replaced by its matrix logarithm $\mathbf{\Psi}$ such that the conformation tensor can be recovered by the matrix exponential $\mathbf{C} = \exp \mathbf{\Psi}$. The initial motivation was to better resolve exponential stress profiles. However, another important fact is that the matrix exponential function ensures that $\mathbf{C}$ stays a symmetric positive definite matrix; a property all non-degenerate covariance matrices share. In fact, it was already known before [2] that a substantial class of macroscopic models respect this microscopic property also in the macroscopic equations, and the divergence of numerical simulations quite often coincided with the loss of this property.

This introduction, so far, suggests that the log-conformation formulation is a rather technical trick to enforce positivity, but in order to shed more light on the failure mechanism of numerical simulations, we want to also highlight the fact that $\mathbf{\Psi}$ naturally appears in the free energy density. E.g., in the Oldroyd-B model or Giesekus model with polymeric viscosity μ_P and relaxation time λ, it has been known for quite some time [3–6], that the free energy density of the polymeric part $\mathcal{F}_P$ is given by $\mathcal{F}_P = \mu_P/(2\lambda)\,(\operatorname{tr}(\mathbf{C}) - \log\det\mathbf{C} - d)$. Acknowledging that $\log\det\mathbf{C} = \operatorname{tr}\log\mathbf{C}$ this can be rewritten in $\mathbf{\Psi}$

$$\mathcal{F}_P = \frac{\mu_P}{2\lambda}\operatorname{tr}\left(e^{\mathbf{\Psi}} - \mathbf{\Psi} - \mathbf{1}\right). \tag{1}$$

The implications of this statement are quite remarkable, since the second law of thermodynamics states that the free energy in total and in absence of external forces has to be non-increasing, which thus puts severe bounds on $\mathbf{\Psi}$. At best, any reasonable numerical simulation should respect this dissipative nature of the free energy, and in the light of this insight it does not seem too unexpected that it is of course easier to construct such a dissipative scheme in $\mathbf{\Psi}$ than in $\mathbf{C}$.

However, even potentially violating this physical principle does not directly explain the failure of numerical simulations. That the free energy relates to the stability of the numerical schemes is mostly an indication from the known mathematical existence results in the discretized setting [7–10]. They all use the free energy to prove existence, and it is thus not unreasonable to conclude that the free-energy-dissipative nature of a numerical scheme and the existence of a numerical solution essentially appear as two sides of the same medal. It is this insight that brings us to the conclusion that the log-conformation formulation has, as far as the fully nonlinear numerical schemes are concerned, solved the HWNP.

* Corresponding author.
E-mail addresses: f.becker@dlr.de (F. Becker), katharina.rauthmann@dlr.de (K. Rauthmann), l.pauli@magmasoft.de (L. Pauli), philipp.knechtges@dlr.de (P. Knechtges).

https://doi.org/10.1016/j.jnnfm.2023.105133
Received 18 August 2023; Received in revised form 9 October 2023; Accepted 10 October 2023
Available online 12 October 2023

Nonetheless, all these advantages have a drawback: the resulting constitutive equation as formulated in Ψ becomes much more complex. Beginning from the first log-conformation formulation, almost all new constitutive equations in Ψ made use of an eigenvalue decomposition of Ψ. The latter is highly unusual for a partial differential equation in the sense that the new equation contains an eigenvalue decomposition of the unknown degrees of freedom. Two notable exceptions to this were [11], which introduced an eigenvalue-free formulation in two-dimensions, and [12], which substituted the eigenvalue-based terms by a Cauchy-type integral in the three-dimensional setting. However, [12] still relied on eigenvalues for the actual numerical computation, since Cauchy integrals are known to be prone to numerical cancellation issues. With this paper we will bridge the gap, and provide an eigenvalue-free implementation also for the three-dimensional setting.

In order to derive this new algorithm, we will use a formulation of the constitutive equation that was introduced in [10]. Since we do not want to derive a constitutive equation from first principles, as it was done in [10], and for the sake of brevity, we rather make the connection to the more popular log-conformation formulations in Section 2. There it will be shown that all these log-conformation formulations are equal in perfect arithmetic.

Given this new formulation, we will, in Section 3, derive an algorithm that allows for the eigenvalue-free numerical evaluation of this term. This algorithm is in principle not bound to a particular discretization scheme, and thus suitable for either finite element or finite volume discretizations.

In Section 4, we then subsequently introduce shortly the finite volume aspects of the numerical scheme we chose to conduct our experiments in. Our implementation is based on the RheoTool software [13], and since our reformulation is independent of the actual discretization of differential operators, we keep the changes minimal. Therefore, we will also not discuss matters of stable discretization of the incompressible Navier–Stokes equations using the finite volume method, and rather refer to [14]. Furthermore, we also want to point the interested reader to the review paper [15] and the references therein for a broader picture on the simulation of viscoelastic fluid flows.

In Section 5, we present two benchmarks: the confined cylinder and the sedimenting sphere. Both benchmarks consider fluid flow around an obstacle, a cylinder and a sphere, respectively. Furthermore, drag coefficient values are computed and compared to results from selected publications.

At last, we also want, for the sake of completeness, mention that other schemes than the log-conformation formulation have been proposed and successfully employed to enforce the positive-definiteness of $\mathbf{C}$. Most notably are the square-root-based approach in [16] or the Cholesky-type decomposition in [17], as well as the more recently introduced contravariant deformation tensor approach [18].

2. Theory of log-conformation formulations

Over the course of the years there have been many different log-conformation formulations, which in perfect arithmetic all yield the same result. Starting point is a constitutive equation of the symmetric conformation tensor $\mathbf{C}$

$$\partial_t \mathbf{C} + (\mathbf{u}\cdot\nabla)\mathbf{C} - \nabla\mathbf{u}\,\mathbf{C} - \mathbf{C}\,\nabla\mathbf{u}^T = -P(\mathbf{C})\,. \tag{2}$$

Here we have chosen the convention that $[\nabla\mathbf{u}]_{ij} = \partial_i u_j$ is the Jacobian of the velocity field $\mathbf{u}$, such that the left-hand side of the equation corresponds to the upper-convected derivative of the conformation tensor. P is in full generality a function of $\mathbf{C}$ that maps $\mathbf{C}$ to another symmetric matrix that commutes with $\mathbf{C}$, i.e., $P(\mathbf{C})\,\mathbf{C} = \mathbf{C}\,P(\mathbf{C})$. Common choices, that are relevant for later sections of this paper, are the Oldroyd-B model $P(\mathbf{C}) = \frac{1}{\lambda}(\mathbf{C}-\mathbf{1})$ with a relaxation time λ, as well as the Giesekus model $P(\mathbf{C}) = \frac{1}{\lambda}(\mathbf{1}+\alpha(\mathbf{C}-\mathbf{1}))(\mathbf{C}-\mathbf{1})$ with an additional mobility parameter α.

The log-conformation formulation now replaces $\mathbf{C}$ by an auxiliary symmetric tensor Ψ such that the two relate via the matrix exponential function $\mathbf{C} = \exp(\Psi)$. As stated in the introduction, the replacement has the advantage that $\mathbf{C}$ stays positive definite. However, this necessitates a new constitutive equation for Ψ that replaces Eq. (2). In the formulation that will be used throughout this paper, this equation is stated as

$$\begin{aligned} 0 = {} & \partial_t\Psi + (\mathbf{u}\cdot\nabla)\Psi + \Psi\omega(\mathbf{u}) - \omega(\mathbf{u})\Psi \\ & -2\,f(\operatorname{ad}\Psi)\,\epsilon(\mathbf{u}) + P(e^{\Psi})e^{-\Psi}\,, \end{aligned} \tag{3}$$

where $\omega(\mathbf{u}) := (\nabla\mathbf{u} - \nabla\mathbf{u}^T)/2$ is the vorticity tensor and $\epsilon(\mathbf{u}) := (\nabla\mathbf{u} + \nabla\mathbf{u}^T)/2$ is the strain tensor. The most important part, however, is $f(\operatorname{ad}\Psi)\,\epsilon(\mathbf{u})$, for which different formulations and numerical algorithms exist. Note that this term distinguishes the different log-conformation formulations, which in perfect arithmetic all yield the same numerical results.

The formulation chosen here is in full generality proven to be equal to the original conformation equation (2) in [10, Theorem A.42]. We will refrain here from an exposition that shows this equivalence from first principles and in full generality. Instead, we explain our formulation first by defining $f(\operatorname{ad}\Psi)\,\epsilon(\mathbf{u})$ properly, and then show the equivalence of the different log-conformation formulation to this formulation in a second step. Those already familiar with one of the other log-conformation formulations should thus more easily grasp the formulation in Eq. (3).

For the definition, we first introduce some terminology: Given two square matrices $\mathbf{A},\mathbf{B} \in \mathbb{R}^{d\times d}$, we define the commutator $[\mathbf{A},\mathbf{B}] = \mathbf{AB} - \mathbf{BA}$. Then, the adjoint operator $\operatorname{ad}\mathbf{A} : \mathbb{R}^{d\times d} \to \mathbb{R}^{d\times d}$ is defined as the linear operator that maps any matrix $\mathbf{B}$ to $[\mathbf{A},\mathbf{B}]$, i.e.,

$$\operatorname{ad}\mathbf{A}\,(\mathbf{B}) := [\mathbf{A},\mathbf{B}]\,.$$

The important point to note here, which will become crucial for our algorithm, is that $\operatorname{ad}\mathbf{A}$ is a linear operator, i.e., a homomorphism from a vector space $\mathbb{R}^{d\times d}$ to the same vector space $\mathbb{R}^{d\times d}$. As such, it is in linear algebra terms representable as a matrix: There exists a matrix $\mathbf{M}$ in $\mathbb{R}^{d^2\times d^2}$ such that

$$\operatorname{ad}\mathbf{A}\,(\mathbf{B}) = \mathbf{M}\tilde{\mathbf{b}}\,,$$

where $\tilde{\mathbf{b}}$ is just a reshaping of the matrix $\mathbf{B}$ to a vector in $\mathbb{R}^{d^2}$, and the product between $\mathbf{M}$ and $\tilde{\mathbf{b}}$ is the usual matrix vector product. To make this more explicit and less abstract, e.g., in the $d = 2$ case we could write $\mathbf{D} = \operatorname{ad}\mathbf{A}\,(\mathbf{B}) = \mathbf{AB} - \mathbf{BA}$ as

$$\begin{pmatrix} D_{11}\\ D_{12}\\ D_{21}\\ D_{22}\end{pmatrix} = \begin{pmatrix} 0 & -A_{21} & A_{12} & 0\\ -A_{12} & A_{11}-A_{22} & 0 & A_{12}\\ A_{21} & 0 & A_{22}-A_{11} & -A_{21}\\ 0 & A_{21} & -A_{12} & 0\end{pmatrix}\begin{pmatrix} B_{11}\\ B_{12}\\ B_{21}\\ B_{22}\end{pmatrix}.$$

For the sake of brevity, and since it will not be used for the actual algorithm, we skip the related formula for $d = 3$. For the rest of the paper d will be fixed to $d = 3$.

Hence, for a given instant of space x and time t, the operation $\operatorname{ad}\Psi\,(\epsilon(\mathbf{u}))$ can be thought of as a matrix–vector multiplication of a matrix $\mathbb{R}^{9\times 9}$ and a vector $\mathbb{R}^9$ for the three-dimensional case. In the following, as is customary for linear operators and especially matrix–vector multiplications, we will omit the parentheses around the argument and just write $\operatorname{ad}\Psi\,\epsilon(\mathbf{u})$.

Lastly, we define $f(\operatorname{ad}\Psi)$ as the application of the function

$$f(x) = \frac{x/2}{\tanh(x/2)} \tag{4}$$

to the 9×9-dimensional matrix that represents $\operatorname{ad}\Psi$.

To summarize: For each instant of space and time, we think of $f(\operatorname{ad}\Psi)\,\epsilon(\mathbf{u})$ as the function f applied to a 9×9-matrix representation of $\operatorname{ad}\Psi$, and the result being multiplied with a 9-vector representation of $\epsilon(\mathbf{u})$.

This is already, modulo several optimizations for symmetric matrices, the gist of the Algorithm 1 in the following section: We will evaluate this function f of a matrix that represents $\operatorname{ad}\boldsymbol{\Psi}$ without the need to do an eigenvalue decomposition of $\boldsymbol{\Psi}$.

This brings us to the second part of this section: The question how previous log-conformation formulations have evaluated this term.

A straightforward way is using the Taylor expansion of f, which is given by

$$f(x) = \sum_{n=0}^{\infty} \frac{B_{2n}}{(2n)!} x^{2n}\,, \tag{5}$$

where B_{2n} are the even Bernoulli numbers. Substituting x by $\operatorname{ad}\boldsymbol{\Psi}$ yields

$$\begin{aligned} f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u}) &= \sum_{n=0}^{\infty} \frac{B_{2n}}{(2n)!} \operatorname{ad}^{2n}\boldsymbol{\Psi}\,\epsilon(\mathbf{u}) \\ &= \sum_{n=0}^{\infty} \frac{B_{2n}}{(2n)!} \underbrace{[\boldsymbol{\Psi},[\boldsymbol{\Psi},[\,\ldots,[\boldsymbol{\Psi}}_{2n \text{ commutators}},\epsilon(\mathbf{u})]\,\ldots\,]\,. \end{aligned} \tag{6}$$

This formulation was first proven in [11, Theorem 1]. However, as it was noted in [11], this formulation alone is for practical numerical simulations not directly usable, since $f(x)$ has singularities at $\pm 2\pi i$, which limits the convergence radius of the Taylor expansion.

To make a connection with the eigenvalue-based formulations, we introduce the eigenvalue decomposition of $\boldsymbol{\Psi}$

$$\boldsymbol{\Psi} = \mathbf{O}\begin{pmatrix}\lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3\end{pmatrix}\mathbf{O}^T\,, \tag{7}$$

with $\mathbf{O} = \left(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\right)$ being an orthogonal matrix. λ_i are the eigenvalues and $\mathbf{e}_i$ the corresponding eigenvectors. For the following, it is also customary to introduce the projection operators $\mathbf{P}_i = \mathbf{e}_i\mathbf{e}_i^T$, which allows us to state the decomposition in the form $\boldsymbol{\Psi} = \sum_i \lambda_i \mathbf{P}_i$. Furthermore, the fact $\mathbf{O}\mathbf{O}^T = \mathbf{1}$ yields $\mathbf{1} = \sum_i \mathbf{P}_i$.

In combination, we can thus state

$$\begin{aligned} \operatorname{ad}\boldsymbol{\Psi}\,\epsilon(\mathbf{u}) &= \boldsymbol{\Psi}\epsilon(\mathbf{u}) - \epsilon(\mathbf{u})\boldsymbol{\Psi} \\ &= \sum_{i,j}(\lambda_i - \lambda_j)\mathbf{P}_i\epsilon(\mathbf{u})\mathbf{P}_j\,. \end{aligned}$$

Furthermore, it is not difficult to see by algebraic manipulations that this can be generalized to any polynomial p

$$p(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u}) = \sum_{i,j} p(\lambda_i - \lambda_j)\mathbf{P}_i\epsilon(\mathbf{u})\mathbf{P}_j\,.$$

It is now mostly an application of the Stone–Weierstrass theorem that this not only holds for polynomials, but also for the continuous function f

$$f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u}) = \sum_{i,j} f(\lambda_i - \lambda_j)\mathbf{P}_i\epsilon(\mathbf{u})\mathbf{P}_j\,. \tag{8}$$

Eq. (8) is the formulation as it was used for numerical evaluation in [10,12,19], and is in some sense closest to what was used in [20].

To see that the more popular eigenvalue-based formulations are just variations of this formulation, we also need to incorporate the rotational term

$$\begin{aligned} &\boldsymbol{\Psi}\omega(\mathbf{u}) - \omega(\mathbf{u})\boldsymbol{\Psi} \\ &\quad= \sum_{i,j}(\lambda_i - \lambda_j)\frac{e^{\lambda_i} - e^{\lambda_j}}{e^{\lambda_i} - e^{\lambda_j}}\mathbf{P}_i\frac{\nabla\mathbf{u} - \nabla\mathbf{u}^T}{2}\mathbf{P}_j\,. \end{aligned}$$

Using $\tanh((\lambda_i - \lambda_j)/2) = (e^{\lambda_i} - e^{\lambda_j})/(e^{\lambda_i} + e^{\lambda_j})$ we can combine this with Eq. (8) to get

$$\begin{aligned} &\boldsymbol{\Psi}\omega(\mathbf{u}) - \omega(\mathbf{u})\boldsymbol{\Psi} - 2\,f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u}) \\ &\quad= -\sum_{i,j}\frac{\lambda_i - \lambda_j}{e^{\lambda_i} - e^{\lambda_j}}\mathbf{P}_i\left(e^{\lambda_j}\nabla\mathbf{u} + e^{\lambda_i}\nabla\mathbf{u}^T\right)\mathbf{P}_j\,. \end{aligned} \tag{9}$$

Furthermore, note that $\lim_{\lambda_j\to\lambda_i}\frac{\lambda_i-\lambda_j}{e^{\lambda_i}-e^{\lambda_j}} = e^{-\lambda_i}$, which allows us to split off the $i = j$ part

$$\begin{aligned} &\boldsymbol{\Psi}\omega(\mathbf{u}) - \omega(\mathbf{u})\boldsymbol{\Psi} - 2\,f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u}) \\ &\quad= -2\mathbf{B} - \sum_{i\neq j}\frac{\lambda_i - \lambda_j}{e^{\lambda_i} - e^{\lambda_j}}\mathbf{P}_i\left(e^{\lambda_j}\nabla\mathbf{u} + e^{\lambda_i}\nabla\mathbf{u}^T\right)\mathbf{P}_j\,, \end{aligned} \tag{10}$$

with

$$\mathbf{B} = \sum_i \mathbf{P}_i\,\epsilon(\mathbf{u})\,\mathbf{P}_i = \sum_i \mathbf{P}_i\,\nabla\mathbf{u}\,\mathbf{P}_i\,. \tag{11}$$

Except notation, Eq. (10) is the same formulation as given in [21, Eq. (44)].

To prove the equivalence to the most widespread log-conformation formulation, we introduce

$$\tilde{\mathbf{M}} = \begin{pmatrix}\tilde{m}_{11} & \tilde{m}_{12} & \tilde{m}_{13} \\ \tilde{m}_{21} & \tilde{m}_{22} & \tilde{m}_{23} \\ \tilde{m}_{31} & \tilde{m}_{32} & \tilde{m}_{33}\end{pmatrix} := \mathbf{O}^T\nabla\mathbf{u}\mathbf{O}\,.$$

We can thus express $\mathbf{B}$ as

$$\mathbf{B} = \mathbf{O}\begin{pmatrix}\tilde{m}_{11} & 0 & 0 \\ 0 & \tilde{m}_{22} & 0 \\ 0 & 0 & \tilde{m}_{33}\end{pmatrix}\mathbf{O}^T\,. \tag{12}$$

Moreover, considering the case of distinct eigenvalues, we introduce

$$\boldsymbol{\Omega} = -\sum_{i\neq j}\frac{1}{e^{\lambda_i} - e^{\lambda_j}}\mathbf{P}_i\left(e^{\lambda_j}\nabla\mathbf{u} + e^{\lambda_i}\nabla\mathbf{u}^T\right)\mathbf{P}_j\,. \tag{13}$$

With the projection operators $\mathbf{P}_i$ being orthogonal and idempotent, i.e., $\mathbf{P}_i\mathbf{P}_j = \delta_{ij}\mathbf{P}_i$ and δ_{ij} being the Kronecker Delta, this yields the equivalent formulation

$$\begin{aligned} &\boldsymbol{\Psi}\omega(\mathbf{u}) - \omega(\mathbf{u})\boldsymbol{\Psi} - 2\,f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u}) \\ &\quad= -2\mathbf{B} + \boldsymbol{\Psi}\boldsymbol{\Omega} - \boldsymbol{\Omega}\boldsymbol{\Psi}\,. \end{aligned} \tag{14}$$

Similarly to the formulation of $\mathbf{B}$ we can also reformulate $\boldsymbol{\Omega}$ using $\tilde{\mathbf{M}}$ as

$$\boldsymbol{\Omega} = \mathbf{O}\begin{pmatrix}0 & \omega_{12} & \omega_{13} \\ \omega_{21} & 0 & \omega_{23} \\ \omega_{31} & \omega_{32} & 0\end{pmatrix}\mathbf{O}^T\,, \tag{15}$$

where the ω_{ij} are given by

$$\omega_{ij} := -\frac{e^{\lambda_j}\tilde{m}_{ij} + e^{\lambda_i}\tilde{m}_{ji}}{e^{\lambda_i} - e^{\lambda_j}}\,. \tag{16}$$

This is mostly the original formulation, as it was first used by Fattal and Kupferman [1] and has been used in many numerical implementations.

For the sake of completeness, and without proof, we also mention the formulation using a Dunford-type/Cauchy-type integral

$$\begin{aligned} &f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u}) \\ &\quad= \frac{1}{(2\pi i)^2}\int_\Gamma\int_\Gamma f(z - z')\,(z\mathbf{1} - \boldsymbol{\Psi})^{-1}\,\epsilon(\mathbf{u})\left(z'\mathbf{1} - \boldsymbol{\Psi}\right)^{-1}\,dz\,dz'\,, \end{aligned} \tag{17}$$

where Γ is a suitably chosen integration contour in the complex plane that encompasses the eigenvalues λ_i, but avoids the singularities of f. This formulation, which was proven in [10,12], facilitates analytical insights into the log-conformation formulation, but is less suited for the direct numerical implementation, due to the expected cancellation effects in the Cauchy-type integral.

3. Eigenvalue-free algorithm design

In the last section, we discussed several of the different existing formulations for the $f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u})$ term in the logarithmic constitutive equation. We also mentioned the connection to the eigenvalue-based algorithms.

In this section, we will come to an eigenvalue-free algorithm that represents $\operatorname{ad}\boldsymbol{\Psi}$ as a matrix on a suitably chosen vector space, which allows us to evaluate $f(\operatorname{ad}\boldsymbol{\Psi})$ as a matrix function.

Since it is instructive for what comes, and since it is also necessary for the numerical implementation, we will first concern ourselves with the eigenvalue-free evaluation of the matrix function $\exp(\boldsymbol{\Psi})$. We will use the Scaling&Squaring algorithm, which has been extensively studied. For an in-depth review article, we refer to [22].

The basic idea of the Scaling&Squaring algorithm consists of two ingredients: One ingredient is a simple approximation of the function. This can, e.g., be a truncated Taylor series, or a rationale approximation. For the exponential function, the Padé approximation $R_{m,m}$ as a rationale approximation is a common choice. Usually, such an approximation is only reasonable in a small region close to some pivot point, which, for our approximation of the exponential function, is the origin of the coordinate system.

At this point, the second ingredient comes into action: a functional relation that helps to map the argument to the region where the aforementioned simple approximation is valid, and thus allows us to construct a more universal approximation. For the exponential function this relation is

$$\exp(\boldsymbol{\Psi}) = \exp(\boldsymbol{\Psi}/2)^2 . \tag{18}$$

Given a general $\boldsymbol{\Psi}$ and iterating this functional equation, one can choose a $j \in \mathbb{N}$ such that $2^{-j}\|\boldsymbol{\Psi}\|$ is small enough for the Padé approximant to be sufficiently good. Then evaluating

$$\exp(\boldsymbol{\Psi}) \approx \left(R_{m,m}(\boldsymbol{\Psi}/2^j)\right)^{2^j} \tag{19}$$

should give a reasonable approximation even for large $\boldsymbol{\Psi}$. As is apparent, we first scale the argument with 2^{-j} and after the evaluation of the Padé approximant, we employ j successive squarings. Hence, the name of the algorithm: Scaling&Squaring.

For the actual implementation, we use the software library Eigen [23], which uses variations of the Scaling&Squaring algorithm, as described in [24, Algorithm 2.3] and [25, Algorithm 3.1].

In principle, as already noted in the previous section, we want to employ a similar algorithm for $f(\operatorname{ad}\boldsymbol{\Psi})$. However, we want to reduce the computational complexity first, i.e., we do not want to represent $\operatorname{ad}\boldsymbol{\Psi}$ as a 9×9 matrix.

Notice that we will apply $f(\operatorname{ad}\boldsymbol{\Psi})$ to the symmetric matrix $\epsilon(\mathbf{u})$ and will get as a result a symmetric matrix again. In fact, the application of a symmetric matrix to $f(\operatorname{ad}\boldsymbol{\Psi})$ will always give a symmetric matrix. This can, e.g., be seen from Eq. (8) by simply transposing the equation, but also from the Taylor series expansion in Eq. (6) and the fact that $\operatorname{ad}^2\boldsymbol{\Psi}$ also has this feature: $\operatorname{ad}^2\boldsymbol{\Psi}$ maps symmetric matrices to symmetric matrices, and antisymmetric matrices to antisymmetric matrices.

The fact that $\operatorname{ad}^2\boldsymbol{\Psi}$ decomposes into two parts, of course nurtures the idea of just using the part operating on symmetric matrices. Since the vector space of symmetric 3×3 matrices is only 6-dimensional, this would already reduce the computational complexity. We could represent $\operatorname{ad}^2\boldsymbol{\Psi}$ as a 6×6-dimensional matrix, and then apply the function

$$g(z) = \begin{cases} \dfrac{\sqrt{z}}{\tanh\sqrt{z}} & \text{for } \operatorname{Re} z \geq 0 \\ \dfrac{\sqrt{-z}}{\tan\sqrt{-z}} & \text{for } \operatorname{Re} z < 0 \end{cases} , \tag{20}$$

such that

$$f(\operatorname{ad}\boldsymbol{\Psi}) = g\left(\frac{1}{4}\operatorname{ad}^2\boldsymbol{\Psi}\right) . \tag{21}$$

This shifts the problem from evaluating a matrix function f to a matrix function g. Note that we have added the negative real part in Eq. (20) to illustrate that g can be continued analytically in the negative half-plane to a meromorphic function. It thus becomes evident that g has a pole at $z = -\pi^2$. In fact, the Taylor expansion follows from Eq. (5)

$$g(z) = \sum_{n=0}^{\infty} \frac{B_{2n}}{(2n)!} 4^n z^n , \tag{22}$$

which, due to the pole, only converges absolutely for $|z| < \pi^2$.

However, we can go one step further: First, we notice that $\operatorname{ad}\boldsymbol{\Psi}$ maps symmetric matrices, like $\epsilon(\mathbf{u})$, to an antisymmetric 3×3-matrix. More importantly, the vector space of antisymmetric 3×3-matrices is 3-dimensional, hence any $\operatorname{ad}^{2n}\boldsymbol{\Psi}$ is at most of rank-3 as a linear operator or matrix for $n > 0$. In other words, in the Taylor series of f or g applied to $\operatorname{ad}\boldsymbol{\Psi}$, only the $n = 0$ term, which is the identity operator/matrix $\mathbf{1}$, is of full rank, while all other terms are at most of rank-3.

This clearly motivates to split off the identity matrix $\mathbf{1}$ and only compute the remaining part on a 3×3-matrix instead of an 6×6-matrix or 9×9-matrix. Thus, we define

$$h(x) = \frac{1}{x}\left(\frac{\sqrt{x}}{\tanh\sqrt{x}} - 1\right) , \tag{23}$$

with its Taylor series for small x given as

$$h(x) = \sum_{n=1} \frac{B_{2n}}{(2n)!} 4^n x^{n-1} . \tag{24}$$

Using the equations above, we can write

$$\begin{aligned} &f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u}) \\ &\quad = \epsilon(\mathbf{u}) + \frac{1}{4}\operatorname{ad}\boldsymbol{\Psi}\; h\left(\frac{1}{4}\operatorname{ad}^2\boldsymbol{\Psi}\right)\; \operatorname{ad}\boldsymbol{\Psi}\,\epsilon(\mathbf{u}) , \end{aligned} \tag{25}$$

which contains already all components of the final algorithm that will compute $f(\operatorname{ad}\boldsymbol{\Psi})\epsilon(\mathbf{u})$.

In the actual computation, we will need different representations of $\operatorname{ad}\boldsymbol{\Psi}$. Going through the different instances of $\operatorname{ad}\boldsymbol{\Psi}$ in Eq. (25) from right to left:

- $\operatorname{ad}\boldsymbol{\Psi}\epsilon(\mathbf{u})$ as noted earlier is an antisymmetric 3×3-matrix, and thus can be represented in some basis as a 3-dimensional vector. We will denote this vector as $\mathbf{v} \in \mathbb{R}^3$.
- On the 3-dimensional space of antisymmetric 3×3-matrices, the operator $\operatorname{ad}^2\boldsymbol{\Psi}$ will be represented as a 3×3-matrix, which we will denote by $\mathbf{X} \in \mathbb{R}^{3\times3}$. Dividing by four and applying h gives another 3×3-matrix $h\left(\frac{1}{4}\operatorname{ad}^2\boldsymbol{\Psi}\right)$, which is multiplied with the 3-vector $\mathbf{v}$ that represents $\operatorname{ad}\boldsymbol{\Psi}\epsilon(\mathbf{u})$. The final result of $h\left(\frac{1}{4}\operatorname{ad}^2\boldsymbol{\Psi}\right)\operatorname{ad}\boldsymbol{\Psi}\epsilon(\mathbf{u})$ is once again then represented by a 3-vector $h(\mathbf{X}/4)\,\mathbf{v}$.
- The last invocation of $\operatorname{ad}\boldsymbol{\Psi}$ linearly maps an antisymmetric 3×3-matrix to a symmetric 3×3-matrix. Therefore, it can be represented as a 6×3-matrix, which we will denote by $\mathbf{Y} \in \mathbb{R}^{6\times3}$. It is multiplied by the 3-vector from the previous step, resulting in a 6-vector $\mathbf{Y}\,h(\mathbf{X}/4)\,\mathbf{v}$.

In order to concretize the computational steps, we will need to choose specific bases. We start with the basis for the symmetric 3×3-matrices: the matrix $\boldsymbol{\Psi}$ is already stored in most codes as a 6-vector $(\Psi_{11}, \Psi_{12}, \Psi_{13}, \Psi_{22}, \Psi_{23}, \Psi_{33})^T$. The same holds for $\epsilon(\mathbf{u})$ with $(\epsilon_{11}, \epsilon_{12}, \epsilon_{13}, \epsilon_{22}, \epsilon_{23}, \epsilon_{33})^T$.

For the antisymmetric 3×3-matrices to be represented as a 3-vector, we want to have further properties for the representation of $\operatorname{ad}^2\boldsymbol{\Psi}$ as a matrix on that vector space. Most notably, we want $\operatorname{ad}^2\boldsymbol{\Psi}$ to be represented as a symmetric matrix $\mathbf{X} \in \mathbb{R}^{3\times3}_{sym}$.

For that, we first define a scaled Frobenius product of two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{3\times3}$, i.e.,

$$(\mathbf{A}, \mathbf{B})_{sF} := \frac{1}{2}\operatorname{tr}\mathbf{A}^T\mathbf{B} . \tag{26}$$

The factor $1/2$ is introduced to avoid several $\sqrt{2}$ factors in the following formulas. The more important aspect here is that $\operatorname{ad}^2\boldsymbol{\Psi}$ is selfadjoint with respect to this scalar product

$$\begin{aligned} (\mathbf{A}, \operatorname{ad}^2\boldsymbol{\Psi}\,\mathbf{B})_{sF} &= \frac{1}{2}\operatorname{tr}\left(\mathbf{A}^T\left(\boldsymbol{\Psi}^2\mathbf{B} - 2\boldsymbol{\Psi}\mathbf{B}\boldsymbol{\Psi} + \mathbf{B}\boldsymbol{\Psi}^2\right)\right) \\ &= \frac{1}{2}\operatorname{tr}\left(\left(\boldsymbol{\Psi}^2\mathbf{A} - 2\boldsymbol{\Psi}\mathbf{A}\boldsymbol{\Psi} + \mathbf{A}\boldsymbol{\Psi}^2\right)^T\mathbf{B}\right) \\ &= (\operatorname{ad}^2\boldsymbol{\Psi}\,\mathbf{A}, \mathbf{B})_{sF} . \end{aligned}$$

Algorithm 1 Computing $f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u})$

Require: $\boldsymbol{\Psi}$ given as $(\Psi_{11},\Psi_{12},\Psi_{13},\Psi_{22},\Psi_{23},\Psi_{33})^T$
Require: $\epsilon(\mathbf{u})$ given as $(\epsilon_{11},\epsilon_{12},\epsilon_{13},\epsilon_{22},\epsilon_{23},\epsilon_{33})^T$
compute $\mathbf{v}\in\mathbb{R}^3$ according to Eq. (30)–(32)
compute $\mathbf{X}\in\mathbb{R}^{3\times3}$ according to Eq. (33)–(38)
compute $\mathbf{Y}\in\mathbb{R}^{6\times3}$ according to Eq. (39)
use Algorithm 2 to compute $\mathbf{Z}\leftarrow h(\mathbf{X}/4)$
return $\epsilon(\mathbf{u})+\frac{1}{4}\mathbf{Y}\,\mathbf{Z}\,\mathbf{v}$

One ramification of the selfadjointness is that the matrix representation of $\operatorname{ad}^2\boldsymbol{\Psi}$ in a basis will yield a symmetric matrix $\mathbf{X}$, if we choose that basis to be orthonormal with respect to the same scalar product.

This motivates our choice of an orthonormal basis $\{\mathbf{E}_i\}$ of the antisymmetric 3×3-matrices

$$\mathbf{E}_1=\begin{pmatrix}0&1&0\\-1&0&0\\0&0&0\end{pmatrix} \tag{27}$$

$$\mathbf{E}_2=\begin{pmatrix}0&0&1\\0&0&0\\-1&0&0\end{pmatrix} \tag{28}$$

$$\mathbf{E}_3=\begin{pmatrix}0&0&0\\0&0&1\\0&-1&0\end{pmatrix}. \tag{29}$$

Going through the different needed representations of $\operatorname{ad}\boldsymbol{\Psi}$, we will start with $\operatorname{ad}\boldsymbol{\Psi}\,\epsilon(\mathbf{u})$, which we represent as a vector $\mathbf{v}\in\mathbb{R}^3$, whose components are given by $v_i=(\mathbf{E}_i,\operatorname{ad}\boldsymbol{\Psi}\,\epsilon(\mathbf{u}))_{sF}$. The latter yields

$$\begin{aligned}v_1=\;&-\epsilon_{11}\Psi_{12}+\epsilon_{12}\Psi_{11}-\epsilon_{12}\Psi_{22}\\&-\epsilon_{13}\Psi_{23}+\epsilon_{22}\Psi_{12}+\epsilon_{23}\Psi_{13}\end{aligned} \tag{30}$$

$$\begin{aligned}v_2=\;&-\epsilon_{11}\Psi_{13}-\epsilon_{12}\Psi_{23}+\epsilon_{13}\Psi_{11}\\&-\epsilon_{13}\Psi_{33}+\epsilon_{23}\Psi_{12}+\epsilon_{33}\Psi_{13}\end{aligned} \tag{31}$$

$$\begin{aligned}v_3=\;&-\epsilon_{12}\Psi_{13}+\epsilon_{13}\Psi_{12}-\epsilon_{22}\Psi_{23}\\&+\epsilon_{23}\Psi_{22}-\epsilon_{23}\Psi_{33}+\epsilon_{33}\Psi_{23}\,.\end{aligned} \tag{32}$$

To represent $\operatorname{ad}^2\boldsymbol{\Psi}$ on the space of antisymmetric 3×3-matrices, we introduce $\mathbf{X}\in\mathbb{R}^{3\times3}$, whose entries are given by $X_{ij}=(\mathbf{E}_i,\operatorname{ad}^2\boldsymbol{\Psi}\,\mathbf{E}_j)_{sF}$. As noted, the resulting matrix $\mathbf{X}$ is symmetric. With our chosen basis, the coefficients are given by

$$X_{11}=\Psi_{11}^2-2\Psi_{11}\Psi_{22}+4\Psi_{12}^2+\Psi_{13}^2+\Psi_{22}^2+\Psi_{23}^2 \tag{33}$$

$$X_{12}=-2\Psi_{11}\Psi_{23}+3\Psi_{12}\Psi_{13}+\Psi_{22}\Psi_{23}+\Psi_{23}\Psi_{33} \tag{34}$$

$$X_{13}=-\Psi_{11}\Psi_{13}-3\Psi_{12}\Psi_{23}+2\Psi_{13}\Psi_{22}-\Psi_{13}\Psi_{33} \tag{35}$$

$$X_{22}=\Psi_{11}^2-2\Psi_{11}\Psi_{33}+\Psi_{12}^2+4\Psi_{13}^2+\Psi_{23}^2+\Psi_{33}^2 \tag{36}$$

$$X_{23}=\Psi_{11}\Psi_{12}+\Psi_{12}\Psi_{22}-2\Psi_{12}\Psi_{33}+3\Psi_{13}\Psi_{23} \tag{37}$$

$$X_{33}=\Psi_{12}^2+\Psi_{13}^2+\Psi_{22}^2-2\Psi_{22}\Psi_{33}+4\Psi_{23}^2+\Psi_{33}^2\,. \tag{38}$$

For the last representation of $\operatorname{ad}\boldsymbol{\Psi}$, from the space of antisymmetric matrices to the space of symmetric 3×3-matrices, we compute $\operatorname{ad}\boldsymbol{\Psi}\,\mathbf{E}_i$ and extract the coefficients. We denote the representation by $\mathbf{Y}\in\mathbb{R}^{6\times3}$ and its coefficients are given by

$$\mathbf{Y}=\begin{pmatrix}-2\Psi_{12}&-2\Psi_{13}&0\\\Psi_{11}-\Psi_{22}&-\Psi_{23}&-\Psi_{13}\\-\Psi_{23}&\Psi_{11}-\Psi_{33}&\Psi_{12}\\2\Psi_{12}&0&-2\Psi_{23}\\\Psi_{13}&\Psi_{12}&\Psi_{22}-\Psi_{33}\\0&2\Psi_{13}&2\Psi_{23}\end{pmatrix}. \tag{39}$$

Taking for the moment the algorithm to compute $h(\mathbf{X}/4)$ as given, we can then use Eq. (25) to compute $f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u})$ as a series of matrix operations. The actual algorithm to compute $f(\operatorname{ad}\boldsymbol{\Psi})\epsilon(\mathbf{u})$ is illustrated in Algorithm 1.

Algorithm 2 Computing $h(\mathbf{X})$

Require: $\mathbf{X}$
$j\leftarrow\max\left(0,j_0+\texttt{std::ilogb}(\|\mathbf{X}\|_F^2)/4\right)$
$\mathbf{L}\leftarrow\mathbf{X}/4^j$
$\mathbf{H}\leftarrow\left(\left(\left(\frac{5}{66}\frac{4^5}{3628800}\mathbf{L}-\frac{1}{30}\frac{4^4}{40320}\mathbf{1}\right)\mathbf{L}+\frac{1}{42}\frac{4^3}{720}\mathbf{1}\right)\mathbf{L}-\frac{1}{30}\frac{4^2}{24}\mathbf{1}\right)\mathbf{L}+\frac{1}{6}\frac{4}{2}\mathbf{1}$
$\mathbf{G}\leftarrow\mathbf{1}+\mathbf{H}\,\mathbf{L}$
for $i=1$ **to** j **do**
$\mathbf{H}\leftarrow\frac{1}{4}\left(\mathbf{H}+\mathbf{G}^{-1}\right)$
$\mathbf{L}\leftarrow4\,\mathbf{L}$
$\mathbf{G}\leftarrow\mathbf{1}+\mathbf{H}\,\mathbf{L}$
end for
return $\mathbf{H}$

Before we come to the general case of computing $h(\mathbf{X}/4)$, we want to first mention a case in which evaluating h becomes as easy as a simple function evaluation: the two-dimensional case.

To see this, note that in the two-dimensional case X_{12} and X_{13} are both zero. As such $\mathbf{X}$ is the direct sum of two submatrices, of which the first one consists of just a single entry X_{11}. Furthermore, since v_1 is the only non-zero entry of $\mathbf{v}$ in this case, it is also just $h(X_{11})$ that needs to be calculated. In fact, acknowledging that

$$h(x)=\frac{1}{x}\left(\sqrt{x}+\frac{2\sqrt{x}}{e^{2\sqrt{x}}-1}-1\right), \tag{40}$$

this yields exactly the representation of $f(\operatorname{ad}\boldsymbol{\Psi})\,\epsilon(\mathbf{u})$ that was given in [11, Theorem 2].

Coming to the general case, we so far only have a Taylor series of h, Eq. (24), which only works for small $\mathbf{X}$. Taking the Scaling&Squaring algorithm for the matrix exponential function as an instructive example, we seek a functional equation that allows us to reduce the computation to arguments that are amenable to the Taylor series.

In fact, using the formula for doubling the argument of $\tanh x$

$$\tanh x=\frac{2\tanh\frac{x}{2}}{1+\tanh^2\frac{x}{2}}, \tag{41}$$

we obtain

$$h(x)=\frac{1}{4}\left(h(x/4)+(g(x/4))^{-1}\right) \tag{42}$$

with

$$g(x/4)=1+x/4\,h(x/4)\,. \tag{43}$$

Considering a general $\mathbf{X}\in\mathbb{R}^{3\times3}_{sym}$, we can readily use this to seek an appropriate $j\in\mathbb{N}$ such that $\mathbf{X}/4^j$ is small enough to be approximated by a truncated Taylor series. Then iterating Eqs. (42) and (43) j-times we get the final result. The full algorithm is displayed in Algorithm 2.

For the actual algorithm, we needed to decide on when $\mathbf{X}/4^j$ is small enough. Evaluating Algorithm 2 for scalar instead of matrix arguments and comparing it with a high-precision calculation of h gives an indication on the accuracy of the algorithm. This analysis yields that $j_0=4$ is sufficient for an absolute accuracy of 10^{-16} in the scalar argument case. Therefore, this is also the value that was used for all our numerical evaluations. We also compared the matrix argument case with an eigenvalue-based evaluation for random $\mathbf{X}$ and could not observe any severe issues.

However, this should not be taken without a word of caution. The functions $g(x)$ and $h(x)$ asymptotically behave like $\sqrt{x}$ and $1/\sqrt{x}$, respectively. In fact, $\sqrt{x}$ as a function is known as a prime example, where an algorithm works for scalar arguments, but may fail for matrix arguments of even moderate condition number, cf. [26,27]. Hence, although our numerical experiments do already give a strong indication for a stable algorithm, a thorough mathematical error analysis of the algorithm is still outstanding and subject of future research.

4. Finite volume implementation

In the following, we are going to embed the new eigenvalue-free constitutive formulation in a numerical implementation. We will, therefore, augment the constitutive equation (3) with a system of partial differential equations consisting of the continuity equation and the momentum balance, as well as Kramers' expression to relate the polymeric stress and the log-conformation field. These equations will then be solved using a finite volume method (FVM), where the polymeric stress is computed with the log-conformation approach according to Eq. (3), and where the $f(\mathrm{ad}\,\boldsymbol{\Psi})\,\epsilon(\mathbf{u})$-term on the right-hand side is computed without an eigenvalue decomposition of $\boldsymbol{\Psi}$ according to Eq. (25) and Algorithm 1.

As noted earlier, the eigenvalue-free log-conformation formulation is quite universal and not necessarily tied to a specific discretization scheme. Like the eigenvalue-based formulation, it needs a point-based evaluation of $\boldsymbol{\Psi}$, and the discretization scheme needs to provide a good approximation of $\nabla\mathbf{u}$ at the same point, such that Algorithm 1 can compute the $f(\mathrm{ad}\,\boldsymbol{\Psi})\,\epsilon(\mathbf{u})$ term also at this point in space and time.

To illustrate how easily this different evaluation of the $f(\mathrm{ad}\,\boldsymbol{\Psi})\,\epsilon(\mathbf{u})$ term can be dropped into an existing code, we chose to base our numerical implementation on one of the existing and established open source computational rheology packages: RheoTool [13]. It is based on OpenFOAM® [28] and has many constitutive models for viscoelastic fluid simulations implemented already. The eigenvalue-free formulations for the log-conf variants of the Oldroyd-B and Giesekus models, which are subject of this work, are implemented among those models and can be used and configured analogously in the overall OpenFOAM® framework. More specifically, we use RheoTool in version 6 and OpenFOAM® in version 9.

A detailed description of the system of partial differential equations and algebraic equations that we will use, and of the corresponding finite volume discretization and linearization follows next. Afterwards, in Section 5, our implementation is applied to the study of two well-known tests for viscoelastic fluid flow: the confined cylinder and the sedimenting sphere benchmarks.

4.1. Statement of the full set of partial differential equations

To state the full system of partial differential equations, which we are going to discretize and solve, we start with the incompressible isothermal Navier–Stokes equations

$$\nabla\cdot\mathbf{u}=0 \tag{44}$$

$$\rho(\partial_t\mathbf{u}+(\mathbf{u}\cdot\nabla)\mathbf{u})=-\nabla p+\nabla\cdot(\eta_s\nabla\mathbf{u})+\nabla\cdot\boldsymbol{\tau}\,, \tag{45}$$

where $\mathbf{u}$ is the velocity vector, p the pressure, $\boldsymbol{\tau}$ the polymeric extra stress tensor, η_s the solvent viscosity, and ρ the density of the fluid. These equations are coupled with an additional partial differential equation for the log-conf tensor $\boldsymbol{\Psi}$, which was already stated in its $f(\mathrm{ad}\,\boldsymbol{\Psi})$ form in Eq. (3). Rearranging some terms, the constitutive equation can be written as

$$\begin{aligned}\partial_t\boldsymbol{\Psi}+(\mathbf{u}\cdot\nabla)\boldsymbol{\Psi}&=-\boldsymbol{\Psi}\omega(\mathbf{u})+\omega(\mathbf{u})\boldsymbol{\Psi}\\&\quad+2\,f(\mathrm{ad}\,\boldsymbol{\Psi})\,\epsilon(\mathbf{u})-P(e^{\boldsymbol{\Psi}})e^{-\boldsymbol{\Psi}}\,.\end{aligned} \tag{46}$$

In the following benchmarks, we only consider the Oldroyd-B and Giesekus constitutive models, thus setting $P(\exp(\boldsymbol{\Psi}))=\frac{1}{\lambda}(\exp(\boldsymbol{\Psi})-\mathbf{1})$ and $P(\exp(\boldsymbol{\Psi}))=\frac{1}{\lambda}(\mathbf{1}+\alpha(\exp(\boldsymbol{\Psi})-\mathbf{1}))(\exp(\boldsymbol{\Psi})-\mathbf{1})$, respectively. The conformation tensor and the log-conformation field are related to the polymeric stress $\boldsymbol{\tau}$ by means of Kramers' expression

$$\boldsymbol{\tau}=\frac{\eta_p}{\lambda}(e^{\boldsymbol{\Psi}}-\mathbf{1})\,, \tag{47}$$

where η_p is the polymeric viscosity and λ the relaxation time of the fluid.

In total, the set of partial differential equations and one algebraic equation (44)–(47) composes, when augmented with appropriate initial and boundary conditions, the mathematical problem we try to solve. In the following, we will lay out our chosen discretization scheme.

4.2. Temporal discretization, linearization and SIMPLEC

Starting off with the set of equations in Eqs. (44)–(47), we at first discretize in time using the backwards Euler scheme. This leads to

$$\nabla\cdot\mathbf{u}_t=0 \tag{48}$$

$$\begin{aligned}\frac{\rho}{\Delta t}\mathbf{u}_t+\rho\left(\mathbf{u}_t\cdot\nabla\right)\mathbf{u}_t&=-\nabla p_t+\nabla\cdot(\eta_s\nabla\mathbf{u}_t)\\&\quad+\nabla\cdot\boldsymbol{\tau}_t+\frac{\rho}{\Delta t}\mathbf{u}_{t-\Delta t}\end{aligned} \tag{49}$$

$$\begin{aligned}\frac{1}{\Delta t}\boldsymbol{\Psi}_t+(\mathbf{u}_t\cdot\nabla)\boldsymbol{\Psi}_t&=-\boldsymbol{\Psi}_t\omega(\mathbf{u}_t)+\omega(\mathbf{u}_t)\boldsymbol{\Psi}_t\\&\quad+2\,f(\mathrm{ad}\,\boldsymbol{\Psi}_t)\,\epsilon(\mathbf{u}_t)\\&\quad-P(e^{\boldsymbol{\Psi}_t})e^{-\boldsymbol{\Psi}_t}+\frac{1}{\Delta t}\boldsymbol{\Psi}_{t-\Delta t}\end{aligned} \tag{50}$$

$$\boldsymbol{\tau}_t=\frac{\eta_p}{\lambda}(e^{\boldsymbol{\Psi}_t}-\mathbf{1})\,. \tag{51}$$

In order to not overload the notation, we drop the t indices from the current time-step and only keep $\mathbf{u}_{t-\Delta t}$ and $\boldsymbol{\Psi}_{t-\Delta t}$.

As a next step, we approach the non-linearity. Therefore, we choose a Picard-type fixed-point iteration. We indicate the current iteration with a suffix i and start our iteration with $\mathbf{u}_0=\mathbf{u}_{t-\Delta t}$ and $\boldsymbol{\Psi}_0=\boldsymbol{\Psi}_{t-\Delta t}$. We linearize our equations in such a way that $\boldsymbol{\Psi}_i$ is solved for after $\mathbf{u}_i$ and p_i have been computed. In the constitutive equation, all non-linear occurrences of $\boldsymbol{\Psi}$ are replaced by $\boldsymbol{\Psi}_{i-1}$. In the momentum equation, we choose to linearize the convective derivative as usual, by computing the flux based on the previous iteration. We thus obtain

$$\nabla\cdot\mathbf{u}_i=0 \tag{52}$$

$$\begin{aligned}\frac{\rho}{\Delta t}\mathbf{u}_i+\rho\left(\mathbf{u}_{i-1}\cdot\nabla\right)\mathbf{u}_i&=-\nabla p_i+\nabla\cdot(\eta_s\nabla\mathbf{u}_i)\\&\quad+\nabla\cdot\boldsymbol{\tau}_{i-1}+\frac{\rho}{\Delta t}\mathbf{u}_{t-\Delta t}\end{aligned} \tag{53}$$

$$\begin{aligned}\frac{1}{\Delta t}\boldsymbol{\Psi}_i+(\mathbf{u}_i\cdot\nabla)\boldsymbol{\Psi}_i&=-\boldsymbol{\Psi}_{i-1}\omega(\mathbf{u}_i)+\omega(\mathbf{u}_i)\boldsymbol{\Psi}_{i-1}\\&\quad+2\,f(\mathrm{ad}\,\boldsymbol{\Psi}_{i-1})\,\epsilon(\mathbf{u}_i)\\&\quad-P(e^{\boldsymbol{\Psi}_{i-1}})e^{-\boldsymbol{\Psi}_{i-1}}+\frac{1}{\Delta t}\boldsymbol{\Psi}_{t-\Delta t}\end{aligned} \tag{54}$$

$$\boldsymbol{\tau}_i=\frac{\eta_p}{\lambda}(e^{\boldsymbol{\Psi}_i}-\mathbf{1})\,. \tag{55}$$

Note that the way we have linearized the system, first $\mathbf{u}_i$ and p_i should be solved in a coupled way, then $\boldsymbol{\Psi}_i$ can be computed based on $\mathbf{u}_i$, which, at last, results in $\boldsymbol{\tau}_i$. It is also noteworthy that our chosen scheme does not use any type of both-sides diffusion (BSD), which was introduced in [29] and applied in a finite volume context in [14].

To further reduce the coupling between $\mathbf{u}_i$ and p_i we employ the SIMPLEC method [30]. For that, consider the following form of the momentum equation (53)

$$\begin{aligned}&\overbrace{\left(\frac{\rho}{\Delta t}+\rho(\mathbf{u}_{i-1}\cdot\nabla)-\nabla\cdot(\eta_s\nabla)\right)}^{=:A-H}\mathbf{u}^*\\&\qquad=-\nabla p^*+\underbrace{\nabla\cdot\boldsymbol{\tau}_{i-1}+\frac{\rho}{\Delta t}\mathbf{u}_{t-\Delta t}}_{=:\mathbf{b}}\,,\end{aligned} \tag{56}$$

where $A-H$ encodes the linear operator that operates on $\mathbf{u}$ in the momentum equation.[1] After the spatial discretization, which follows in Section 4.3, A will be the diagonal part of the matrix and $-H$ the off-diagonal part. In particular A will be easy to invert.

Now, assuming $\mathbf{u}^*$ solves Eq. (56) given the pressure $p^*:=p_{i-1}$ from the previous iteration, we seek an update $\mathbf{u}'$ such that $\mathbf{u}_i=\mathbf{u}^*+\mathbf{u}'$ solves the continuity equation (52). Introducing the pressure update $p'=p_i-p^*$, the velocity update $\mathbf{u}'$ needs to solve

$$(A-H)\,\mathbf{u}'=-\nabla p'\,. \tag{57}$$

[1] Our notation deviates a bit from the actual implementation in OpenFOAM®, where H is used to denote what is here given as $H\mathbf{u}^*+\mathbf{b}$.

SIMPLEC now approximates H by another operator H_1, which like A is easy to invert. In the actual implementation, i.e., after the spatial discretization, H_1 will be realized as a matrix lumping of the off-diagonal entries onto the diagonal. For the details consult [14]. Thus, we can solve

$$\mathbf{u}' = -\left(A - H_1\right)^{-1} \nabla p' . \tag{58}$$

Therefore, the continuity equation $\nabla \cdot (\mathbf{u}' + \mathbf{u}^*) = 0$ amounts to

$$\begin{aligned} 0 = \nabla \cdot \Big(& -\left(A - H_1\right)^{-1} \nabla (p_i - p^*) \\ & + A^{-1} \left(H\mathbf{u}^* - \nabla p^* + \mathbf{b}\right) \Big) , \end{aligned} \tag{59}$$

which can be rearranged to the pressure correction equation

$$\begin{aligned} & \nabla \cdot \left(\left(A - H_1\right)^{-1} \nabla p_i \right) \\ & \quad = \nabla \cdot \left(A^{-1}(H\mathbf{u}^* + \mathbf{b}) + \left((A - H_1)^{-1} - A^{-1}\right) \nabla p^* \right) . \end{aligned} \tag{60}$$

The corrected velocity $\mathbf{u}_i$ is then given by

$$\begin{aligned} \mathbf{u}_i = & A^{-1}(H\mathbf{u}^* + \mathbf{b}) + \left((A - H_1)^{-1} - A^{-1}\right) \nabla p^* \\ & - \left(A - H_1\right)^{-1} \nabla p_i . \end{aligned} \tag{61}$$

In principle, we now have arrived at a set of decoupled partial differential equations (56), (54) and (60) and two algebraic evaluations (55) and (61) that can be composed into an algorithm as illustrated in Fig. 1. However, Fig. 1 contains another interior fixed-point loop around the pressure correction equation (60). The rationale here is that the spatial discretization of the surface gradient ∇p_i, which will be described in the following section, is defective for non-orthogonal meshes. To correct for this, some computations in the scheme are deferred in a non-linear fashion, which then necessitate another fixed-point loop around the discretized version of Eq. (60). The latter happens even though Eq. (60) looks linear on the current level of abstraction. For the details, we refer the reader to [31, Sec. 9.8].

In all simulations that are presented in Section 5, a total of two inner iteration loops and two non-orthogonal correction steps per time-step are used.

4.3. Spatial discretization

After temporal discretization, linearization and decoupling of velocity and pressure with the SIMPLEC method, we arrive at three decoupled, linear partial differential equations (56), (60) and (54). In order to solve those, we need to choose a method for spatial discretization. As noted earlier, we have chosen the Finite Volume Method (FVM), and in particular base our implementation on RheoTool [13] and OpenFOAM® [28].

In the FVM, the computational domain is subdivided into a set of appropriate interconnected control volumes (the mesh) and the integral form of these PDEs is then evaluated on every single control volume [31]. The variables of interest ($\mathbf{u}^*$, p_i and $\mathbf{\Psi}_i$) are, in our choice of a cell-centered FVM, considered as discrete fields (vector-, scalar- and tensorfields, respectively) which attain their respective value at the cell center. The appearing spatial differential operators are then approximated using different schemes that solely depend on those cell-centered quantities. With the initial PDEs being linear, this approach results in sparse linear equation systems.

Next, we list the configuration of the spatial discretization schemes, which will be used throughout all simulations that follow in Section 5.

- The divergence terms are discretized according to the divergence theorem via the Gauss scheme. For that, the argument of the divergence operator needs to be evaluated on the faces of the cell. For $\nabla \cdot \boldsymbol{\tau}_{i-1}$ or $\nabla \cdot \left(A^{-1}(H\mathbf{u}^* + \mathbf{b})\right)$ this means that the cell-centered value is interpolated linearly from cell to face. In Eq. (60) the term $(A - H_1)^{-1} - A^{-1}$ is also linearly interpolated from cell to face.
- The Laplacian terms, such as $\nabla \cdot (\eta_s \nabla \mathbf{u}_i)$ and $\nabla \cdot \left(\left(A - H_1\right)^{-1} \nabla p_i \right)$, are also discretized using Gaussian integration, with the difference that only the inner factors are linearly interpolated. The gradients $\nabla \mathbf{u}_i$ and ∇p_i, but also ∇p^* in Eq. (60), are directly evaluated on the face using a surface normal scheme. In all our computations we have employed a surface normal gradient scheme with an explicit deferred non-orthogonal correction.
- Cell-centered gradients, as ∇p^* and $\nabla \mathbf{u}_i$ in Eqs. (54), (56), (61), are computed using the Gauss scheme with linear interpolation. Interpolation in general is linear per default, whenever needed.
- For the convective term in the constitutive equation, $(\mathbf{u}_i \cdot \nabla)\mathbf{\Psi}_i$, the corrected, component-wise CUBISTA scheme is used, which is described in [14]. The convective term $(\mathbf{u}_{i-1} \cdot \nabla)\mathbf{u}_i$ in the momentum balance is removed from Eq. (56) in the later benchmarks (to enforce Re = 0) and, therefore, no discretization scheme is needed.

Overall, all used spatial discretization schemes are under ideal conditions, i.e., on orthogonal meshes, second order accurate. However, as for example shown in [32], the gradient computation may lose its second order accuracy on meshes of poor quality, e.g., high non-orthogonalities or skewnesses. As a consequence, particular attention was paid to the selection and design of the hexahedral meshes in Section 5.

A crucial aspect when simulating incompressible Navier–Stokes equations, regardless of the actually employed spatial discretization scheme, are the issues with checkerboard patterns and in general the saddle-point structure of the linearized problem. Here, this issue has been approached with the Rhie–Chow method [33], where $\nabla p_i, \nabla p^*$ are differently discretized in Eq. (60) than they are in Eqs. (56) and (61). We do not want to go into the details here, since they have already been laid out in [14], but solely mention two points: Firstly, there is a connection to the—in the finite element world important—inf-sup condition, and we refer the interested reader to [34] for a recent account into that direction. Secondly, on top of what has just been described, OpenFOAM® employs a correction of the flux in Eq. (60) that shall remedy unphysical dependencies of steady-state solutions on the actually chosen time-step size. The reader is once again referred to [14] for the details.

Of course, boundary conditions do also constitute an important aspect of numerical methods for partial differential equations. The specific choice of boundary conditions for the later benchmarks will follow in the corresponding Sections 5.1 and 5.2. Nonetheless, it should be noted that boundary conditions are handled according to the technique that is implemented in OpenFOAM®, where specific boundary structures, called patches, are used to store boundary information. Hence, whenever needed by a certain discretization scheme for elements at the edge of the computational domain, the required values, that cannot be provided by interior neighbors, are fetched form these boundary patches.

Finally, we will mention that the choice of the viscoelastic model (e.g., Oldroyd-B or Giesekus) and in particular the implementation of the eigenvalue-free $f(\mathrm{ad}\,\mathbf{\Psi})$ term does not affect the overall procedure depicted in Fig. 1, but rather the assembling of the right-hand side of Eq. (54). It can therefore be implemented quite straightforward as described in Section 3 by computing the $f(\mathrm{ad}\,\mathbf{\Psi})$ term according to Algorithm 1.

4.4. Choice of linear solvers

Through the spatial discretization in the last section, we now have effectively derived three systems of sparse linear equation system that correspond to Eqs. (56), (60), (54) and which are solved for the cell-centered values of $\mathbf{u}^*, p_i$ and $\mathbf{\Psi}_i$. For the rest of this section, we will refer to these systems as the $\mathbf{u}^*, p_i$ and $\mathbf{\Psi}_i$ equation respectively. One immediate computational optimization, which is employed in

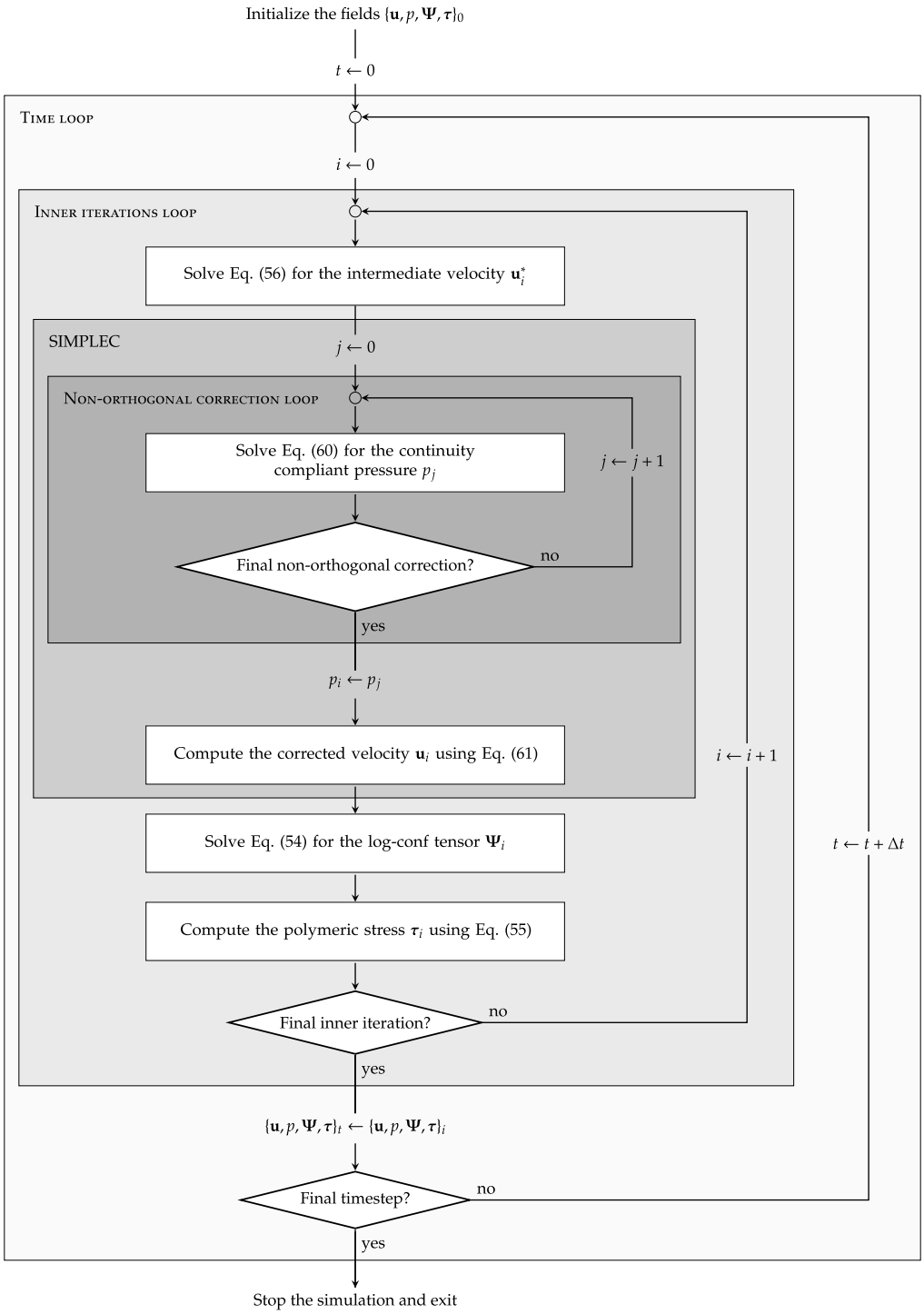

Fig. 1. Solver flowchart.

OpenFOAM®, is that the left-hand sides of Eqs. (54) and (60) can be decoupled and solved individually for the components of $\mathbf{u}^*, \boldsymbol{\Psi}_i$.

After this optimization, the individual linear systems are solved using the following solvers: For the $\mathbf{u}^*$ and p_i equations, the Preconditioned Conjugate Gradient Method (PCG) is applied with an Diagonal-Based Incomplete Cholesky preconditioner (DIC). An absolute tolerance of 10^{-10}, relative tolerance of 10^{-4} and a maximum number of 1000 iterations are chosen as the possible termination criteria for these solvers. The $\boldsymbol{\Psi}_i$ equation uses a Preconditioned Bi-Conjugate Gradient method (PBiCG) with an Diagonal-Based Incomplete LU preconditioner (DILU). The same termination configuration is chosen as for the $\mathbf{u}^*$ and p_i equations.

In our numerical algorithm, the currently available field data is used as the initial guess for the corresponding iterative solver. In our benchmarks, a dimensionless timescale $T = t/\lambda$ is used and each simulation is run until $T = 30$ with a Courant number of 0.5. We, therefore, ensure that the viscoelastic stresses in the fluid have converged at the end of a simulation, i.e., that the fluid has reached a steady-state. Within this steady-state, the initial guesses for the iterative solvers will already be close to the actual solutions, such that the number of iterations is

expected to decrease as the simulation progresses in time. However, in a non-steady-state, i.e., at the beginning of a simulation, the initial guesses may be quite far from the actual solution of the system, such that more iterations are needed in general.

Typically, the p_i equation is the most expensive to solve. At the beginning of a simulation, the p_i equation requires several hundred iterations for convergence or even reaches the maximum number of iterations on our finest meshes. Overall the number of iterations needed for convergence decreases as the fluid approaches a steady-state. In a steady-state, there is often no need for a single iteration of the $\mathbf{u}^*$ and $\mathbf{\Psi}_i$ equations, since the initial guess already solves the system well enough.

5. Benchmarks

In this section, our implementation of the newly derived eigenvalue-free constitutive formulation is applied to a study of two benchmarks: the confined cylinder and the sedimenting sphere. These benchmarks represent similar flow problems, i.e., flow around an obstacle, in a two-dimensional and a three-dimensional case, respectively. Both benchmarks have been examined in the literature before, in order to validate new numerical schemes or models, see for example [11,21,35–39] for the confined cylinder and [12,40–43] for the sedimenting sphere. For comparability, we specifically follow the setups, i.e., the geometries and fluid parameters, that were used in [11] for the confined cylinder and [12] for the sedimenting sphere. A detailed description will follow in the corresponding Sections 5.1 and 5.2, where results for the eigenvalue-free logarithmic Oldroyd-B and Giesekus models are shown and discussed.

The main quantity of interest in both benchmarks is the drag coefficient C_d, which describes the non-dimensionalized force the fluid exerts on the obstacle in x-direction. C_d is given by

$$C_d = \frac{1}{(\eta_s + \eta_p)\bar{u}} \int_\Gamma \mathbf{e}_x \cdot (\boldsymbol{\sigma}\mathbf{n}), \tag{62}$$

where Γ is the surface of the obstacle, $\mathbf{n}$ the corresponding unit normal, $\mathbf{e}_x$ the unit vector in x-direction and $\boldsymbol{\sigma}$ the Cauchy stress tensor

$$\boldsymbol{\sigma} = -p\mathbf{1} + \eta_s(\nabla\mathbf{u} + \nabla\mathbf{u}^T) + \boldsymbol{\tau}. \tag{63}$$

It is known that the drag coefficient varies with the Reynolds number Re of the simulation. This has for example been investigated by [35]. However, for comparability, we follow the literature and consider creeping flow conditions (Re = 0) in both benchmarks by removing the convective term from the momentum equation (45).

Overall, a variety of flow simulations for different Weissenberg numbers will be presented and the corresponding drag coefficient values will be compared to the literature. The dimensionless Weissenberg number is given by

$$\mathrm{Wi} = \frac{\lambda\bar{u}}{R}, \tag{64}$$

where λ is the relaxation time of the fluid, R is the radius of the cylinder or the sphere and $\bar{u}$ the mean inflow velocity.

5.1. Confined cylinder

In the confined cylinder case, a two-dimensional channel with a cylindrical obstacle of radius R in its center is considered as the computational domain. The channel has a height of $4R$, such that the ratio of the channel height to the cylinder diameter is 2. Our setup mimics the setup of Knechtges et al. [11] and Hulsen et al. [21], where the channel has a total length of $30R$ in order to reduce effects of the inflow and outflow and where the cylinder center is at $(15R, 2R)$. An illustration of the geometry can be seen in Fig. 2.

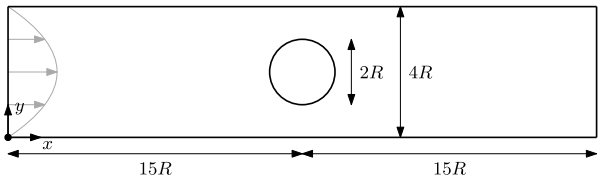

Fig. 2. Illustration (not to scale) of the confined cylinder. Fluid flows from the inlet at the left side to the outlet at the right side. The upper and lower boundaries of the channel and the cylinder surface are considered as solid walls.

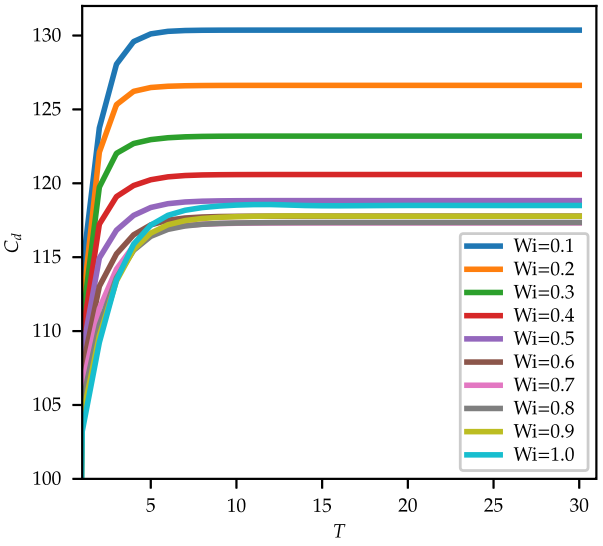

Fig. 3. Convergence of the C_d values for the confined cylinder case on mesh M3 at different Weissenberg numbers over time from $T = 1$ to $T = 30$ using the eigenvalue-free logarithmic Oldroyd-B formulation.

Table 1
Mesh statistics for the confined cylinder geometry.

	M1	M2	M3
Number of elements in the mesh	99 576	398 304	1 593 216
Number of elements on the cylinder surface	756	1512	3024
Average element non-orthogonality	12.6	12.6	12.7
Maximum element non-orthogonality	44.7	44.9	45.0
Maximum element skewness	1.5	1.5	1.5

5.1.1. Setup

Boundary and initial conditions are chosen according to literature. At the inlet, a fully developed Poiseuille solution for an Oldroyd-B fluid is imposed for the velocity $\mathbf{u}$ (with mean inflow $\bar{u}$) and the polymeric extra stresses $\boldsymbol{\tau}$ and $\mathbf{\Psi}$, similar to [11]. A zero-gradient condition is considered for the pressure p. The exact values for the Poiseuille flow are given in the Appendix. Furthermore, at the channel and cylinder walls, zero-gradient conditions are considered for the pressure and zero velocities ($\mathbf{u} = \mathbf{0}$). The polymeric extra stress components are linearly extrapolated. At the outlet, zero-gradient conditions are imposed for all variables, except for the pressure, which is set to zero. Initially ($t = 0$) the fluid is at rest ($\mathbf{u} = \mathbf{0}$) and the extra-stresses are null ($\boldsymbol{\tau} = \mathbf{\Psi} = \mathbf{0}$). The pressure is set to zero as well.

In all of the following tests, $R = 1\,\mathrm{m}$ and $\bar{u} = 1\,\mathrm{m/s}$ were fixed, such that the Weissenberg number equals the numerical value of the relaxation time in seconds and could therefore easily be controlled by a change of λ. Finally, as in the corresponding literature, a viscosity

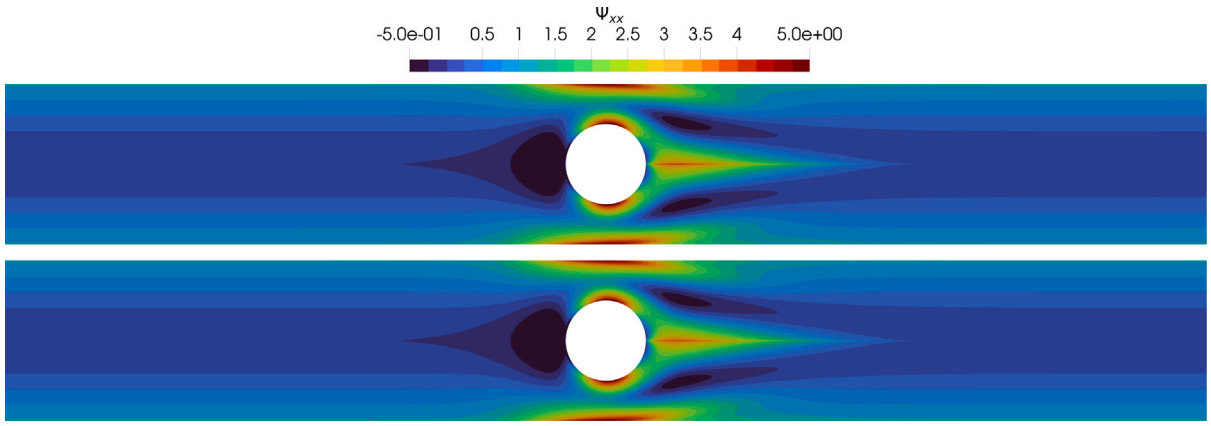

Fig. 4. Comparison of Ψ_{xx} at the final time-step $T = 30$. Top: computed with the eigenvalue-free logarithmic Oldroyd-B formulation; bottom: computed with the standard logarithmic Oldroyd-B formulation that relies on an eigenvalue decomposition. Looking at the entrance of both simulations, it can additionally be seen that developed Poiseuille inflow conditions have been used, since the Ψ_{xx} components are already developed at the inlet.

ratio of $\beta = \eta_s/(\eta_s + \eta_p) = 0.59$ and a density of $\rho = 1\,\mathrm{kg/m^3}$ have been used.[2]

Three quadrilateral meshes M1, M2 and M3 of different refinement levels have been considered. Their main properties are shown in Table 1. In each refinement step the total number of elements is quadrupled from mesh to mesh and the number of elements at the cylinder surface is doubled. An important property of these meshes and their refinement is that characteristics, such as the element non-orthogonality and skewness, are sufficiently small. Element non-orthogonality refers to the angle between the vector of two neighbored cell centers and their corresponding face normal. Element skewness refers to the deviation of the intersection point of this cell-center-connecting vector from the actual face center. For example, in a pure square mesh, element non-orthogonality and skewness would both be zero. In the FVM, the gradient computation can be negatively affected by such mesh irregularities, as is described and investigated by Syrakos et al. [32]. Furthermore, the importance of good quality meshes and strategic mesh refinement is particularly emphasized in [31]. Therefore, only mesh configurations were considered where these characteristic values were sufficiently small on all refinement levels. RheoTool does already provide a confined cylinder case with an appropriate mesh [13]. The latter has been used as the basis for our benchmarks and adjusted, e.g., by adding several different refinement levels for the mesh. It should also be mentioned, that in order to achieve reasonable C_d values, boundary layers around the obstacle surface were used. This use of thin boundary layers has increased the resolution of the solution close to the obstacle surface and did also reduce the extrapolation error, resulting in C_d values that are in good agreement with the literature.

As already mentioned in Section 4.4, adaptive time-stepping kept a Courant number of 0.5 in all simulations. Typical time-step sizes were then ranging from 1.8×10^{-3} s on M1, to 9.0×10^{-4} s on M2, and 4.5×10^{-4} s on M3. For all simulations, a dimensionless timescale $T = t/\lambda$ was used with end time $T = 30$ in order to ensure convergence of the fluid to a steady-state. Therefore, the C_d values also converge eventually, as can be seen in Fig. 3.

5.1.2. Results

Table 2 shows the final C_d values for the eigenvalue-free logarithmic Oldroyd-B formulation. Overall, the results on the finest mesh M3 show good agreement with the literature at all considered Weissenberg numbers. At smaller Weissenberg numbers (Wi ≤ 0.7) the values in the compared publications [11,21,35,36] deviate at a magnitude of 10^{-3} and our results on M3 (which we consider as our most accurate ones) do also fit into this range. At higher Weissenberg numbers the values tend to deviate more from each other among all publications, roughly at a magnitude of 10^{-2}; a property that has already been observed and described for example in [11]. Furthermore, all publications agree that the minimum drag coefficient is obtained at Wi $= 0.7$. The highest C_d values of around 130.36 are reached at the lowest Weissenberg number of 0.1.

Table 2
Final values for the drag coefficient C_d at $T = 30$ for the confined cylinder case, using the eigenvalue-free Oldroyd-B formulation at different Weissenberg numbers.

Wi	C_d						
	M1	M2	M3	[11]	[21]	[35]	[36]
0.1	130.31898	130.36049	130.36653	130.3626	130.363	130.364	130.36
0.2	126.58894	126.62264	126.62875	126.6252	126.626	126.626	126.62
0.3	123.16959	123.18940	123.19475	123.1912	123.193	123.192	123.19
0.4	120.59084	120.59124	120.59500	120.5912	120.596	120.593	120.59
0.5	118.85227	118.82872	118.83021	118.8260	118.836	118.826	118.83
0.6	117.83174	117.78125	117.77988	117.7752	117.775	117.776	117.78
0.7	117.40242	117.32483	117.32079	117.3157	117.315	117.316	117.32
0.8	117.45188	117.35293	117.35114	117.3454	117.373	117.368	117.36
0.9	117.87883	117.76574	117.77477	117.7678	117.787	117.812	117.80
1.0	118.60224	118.47727	118.49927		118.471		118.49

Fig. 4 shows solutions of the confined cylinder case at $T = 30$ and a Weissenberg number Wi $= 0.7$. Presented are the Ψ_{xx} components for the eigenvalue-free logarithmic Oldroyd-B formulation in comparison with a eigenvalue-based formulation, that is described by Pimenta [14, Eq. (7)] and previously implemented in RheoTool [13]. The contours of the tensor components, and in particular those close to the cylinder, look almost identical. To emphasize and quantify the similarity of these solutions, it can additionally be stated that their final C_d difference is only of magnitude 10^{-6}.

Table 3 shows C_d results for computations with the eigenvalue-free logarithmic Giesekus model. The Giesekus model has an additional parameter, the mobility factor $\alpha \in [0, 1]$. Again, good agreement with the literature can be observed. Additionally, our results show the significant influence of α on the drag coefficient. We do not want to go into detail here, as the effect of α on C_d has already been investigated by others, see for example [35]. As α increases (for fixed Wi), the drag decreases, which is explained by the shear-thinning property of the Giesekus model. When α tends to zero, the Giesekus model transitions to the Oldroyd-B model and thus, the C_d values converge to the corresponding values in Table 2.

All computations were run in parallel on the Caro HPC cluster of the German Aerospace Center. M1 simulations were run on 32 cores, M2

[2] The parameters for our tests are chosen according to the literature for comparability and do not represent real fluids.

Table 3
Final values for the drag coefficient C_d at $T = 30$ for the confined cylinder case, using the eigenvalue-free Giesekus formulation at different Weissenberg numbers. Three different mobility factors $\alpha \in \{0.1, 0.01, 0.001\}$ were considered.

Wi	C_d											
	$\alpha = 0.1$				$\alpha = 0.01$				$\alpha = 0.001$			
	M1	M2	M3	[35]	M1	M2	M3	[35]	M1	M2	M3	[35]
0.1	125.542	125.585	125.591	125.587	129.626	129.667	129.674	129.671	130.246	130.287	130.293	130.291
0.2	117.068	117.109	117.116	117.113	124.629	124.666	124.672	124.670	126.358	126.392	126.398	126.396
0.3	111.055	111.095	111.102	111.098	120.050	120.081	120.087	120.085	122.753	122.775	122.780	122.778
0.4	106.814	106.852	106.859	106.855	116.487	116.513	116.519	116.517	119.974	119.979	119.984	119.981
0.5	103.694	103.731	103.737	103.733	113.842	113.863	113.869	113.867	118.020	118.005	118.008	118.005
0.6	101.304	101.340	101.345	101.341	111.884	111.900	111.906	111.906	116.756	116.721	116.722	116.719
0.7	99.413	99.447	99.452	99.448	110.401	110.415	110.421	110.422	116.040	115.986	115.985	115.982
0.8	97.875	97.908	97.913	97.909	109.238	109.249	109.255	109.258	115.736	115.666	115.665	115.679
0.9	96.599	96.631	96.636	96.631	108.287	108.297	108.302	108.307	115.724	115.641	115.642	115.664
1.0	95.520	95.552	95.556	95.552	107.483	107.491	107.496	107.505	115.907	115.811	115.813	115.868

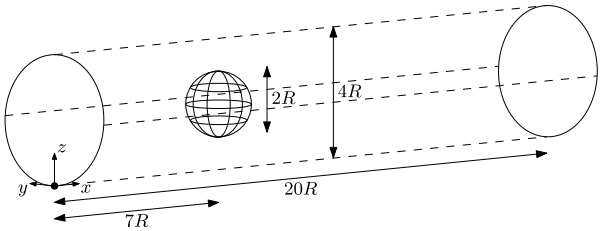

Fig. 5. Illustration (not to scale) of the sedimenting sphere. Fluid flows from the inlet at the left side to the outlet at the right side. A fixed non-zero velocity is considered at the channel wall. A sphere with solid surface (zero velocity) is placed inside the channel.

simulations on 64 cores and M3 simulations on 128 cores. In its current state, we observe that our implementation of the eigenvalue-free variant is slightly slower than the standard eigenvalue-based implementation. In particular, we measure a runtime increase of around 7% per time-step in the log-conf equation. However, solving the constitutive equation for a single relaxation mode has only a minor impact on the overall runtime of the algorithm, since the momentum equation and the SIMPLEC algorithm are more computationally heavy. This is corroborated by the comparison of the total runtimes for our test case on M3 using the eigenvalue-free formulation with those of the standard formulation, which differ by less than 1%.

Further performance optimizations of the Algorithms 1 and 2 are possible, but at the moment not considered in our prototypical implementation. At the moment, e.g., our implementation does not exploit the matrix symmetry of $\mathbf{X}$ in Algorithm 2, which could easily save some floating point operations. Another optimization opportunity that is currently unexploited, and which is for the eigenvalue-based implementations much more difficult to pursue, is to bring the actual computations onto a GPU.

We also applied the eigenvalue-free approach to other simulation cases at higher Weissenberg numbers and did not observe any significant differences regarding its stability compared to the standard approach.

5.2. Sedimenting sphere

To demonstrate the eigenvalue-free approach on a three-dimensional problem, a simulation similar to the confined cylinder, the sedimenting sphere, is considered. In this benchmark, fluid flow around a spherical obstacle inside a three-dimensional channel is considered. The sphere has a radius of R and the channel a height (or diameter) of $4R$. Based on the setup in [12], we impose a channel length of $20R$ and keep the sphere centered at $(7R, 0, 2R)$. An excerpt of the computational domain is shown in Fig. 5.

Table 4
Mesh statistics for the sedimenting sphere geometry.

	M1	M2	M3
Number of elements in the mesh	139 392	1 115 136	8 921 088
Number of elements on the sphere surface	1152	4608	18 432
Average element non-orthogonality	11.1	11.7	12.0
Maximum element non-orthogonality	41.1	52.4	64.4
Maximum element skewness	1.7	1.8	1.8

5.2.1. Setup

Boundary and initial conditions are chosen according to the literature in order to increase comparability. A uniform inlet condition is considered, with a fixed non-zero velocity $\bar{u}$ in x-direction, zero polymeric extra stress components, and a zero-gradient condition for the pressure. At the channel wall, the boundary conditions are chosen equal to the inlet conditions. Thus, in particular, the velocity is uniformly fixed with non-zero component in x-direction as well. At the sphere, a no-slip condition for the velocity is considered ($\mathbf{u} = \mathbf{0}$). The polymeric extra stress components are linearly extrapolated onto the surface and the pressure uses a zero-gradient condition. At the outlet, zero-gradient conditions are imposed for all variables except for the pressure, which uses a fixed value condition $p = 0$.

In the following tests, $R = 1\,\mathrm{m}$ and $\bar{u} = 1\,\mathrm{m/s}$ were used, such that the Weissenberg number equals the numerical value of the relaxation time in seconds and can again be controlled by a change of λ. As in the corresponding literature, a viscosity ratio of $\beta = \eta_s/(\eta_s + \eta_p) = 0.5$ and a density of $\rho = 1\,\mathrm{kg/m^3}$ have been used.

Three purely hexahedral meshes M1, M2 and M3 of different refinement levels have been considered. Their main properties are shown in Table 4. During refinement, the total number of elements is multiplied by eight from mesh to mesh, while the number of elements at the sphere surface is quadrupled. It was observed that the C_d computation in this case was very sensitive to the overall mesh quality. Configuring the mesh for the sedimenting sphere simulations, with the goal to minimize non-orthogonalities and skewnesses on all refinement levels, did therefore play an important role during our research. Additionally, boundary layers around the sphere surface were introduced for smaller numerical errors close to the surface and, therefore, a better C_d accuracy. An excerpt of mesh M1 is presented in Fig. 6. Again, an end time $T = 30$ was used and a Courant number of 0.5 was fixed, leading to typical time-step sizes ranging from $1.6 \times 10^{-2}\,\mathrm{s}$ on M1, to $8.0 \times 10^{-3}\,\mathrm{s}$ on M2 and $4.0 \times 10^{-3}\,\mathrm{s}$ on M3. At this point, it should be mentioned that the quantity of interest in the following tests is the drag correction factor K, which is typically used in sedimenting sphere benchmarks [12,40–43]. K is given by

$$K = \frac{C_d}{6\pi}, \tag{65}$$

where C_d is the drag coefficient value from Eq. (62) with Γ being the sphere surface. The definition of K is motivated by Stokes' law [44,45].

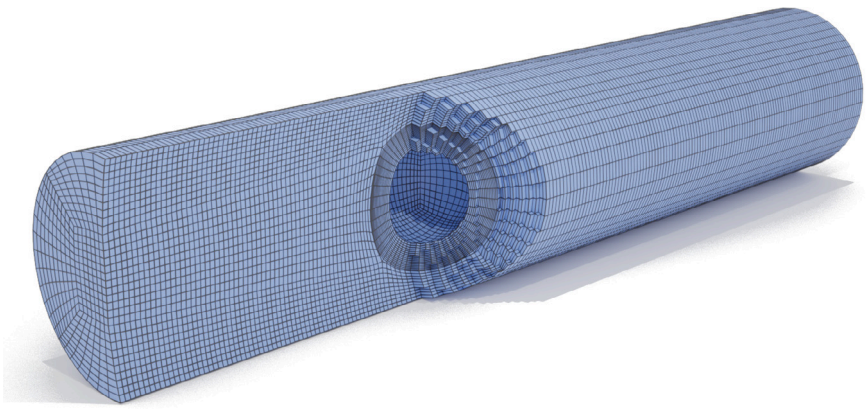

Fig. 6. Rendering of the hexahedral mesh M1 for the sedimenting sphere benchmark.

Table 5
Final values for the drag coefficient C_d at $T = 30$ for the sedimenting sphere case, using the eigenvalue-free Oldroyd-B formulation at different Weissenberg numbers. RE corresponds to the Richardson extrapolation value.

Wi	K								
	M1	M2	M3	RE	[12]	[40]	[41]	[42]	[43]
0.1	5.73784	5.82977	5.86723	5.90469	5.90576				
0.2	5.64635	5.73532	5.77120	5.80708	5.80763				
0.3	5.53994	5.62529	5.65930	5.69331	5.69356	5.69368	5.6963		
0.4	5.43888	5.52076	5.55300	5.58524	5.58527				
0.5	5.35026	5.42977	5.46043	5.49109	5.49093			5.4852	
0.6	5.27577	5.35396	5.38330	5.41264	5.41227	5.41225	5.4117	5.4009	
0.7	5.21468	5.29244	5.32071	5.34898	5.34838			5.3411	
0.8	5.16544	5.24335	5.27092	5.29849	5.29747			5.2945	
0.9	5.12649	5.20481	5.23202	5.25923	5.25761	5.25717		5.2518	
1.0	5.09638	5.17511	5.20219	5.22927	5.22700			5.2240	
1.1	5.07430	5.15274	5.17989	5.20704	5.20402			5.2029	
1.2	5.05872	5.13653	5.16379	5.19105	5.18733	5.18648		5.1842	5.1877
1.3	5.04914	5.12552	5.15281	5.18010	5.17581				5.1763
1.4	5.04439	5.11890	5.14608	5.17326	5.16851				
1.5	5.04361	5.11609	5.14291	5.16973		5.15293			

5.2.2. Results

Table 5 shows K values for the eigenvalue-free logarithmic Oldroyd-B formulation and varying Weissenberg numbers between 0.1 and 1.5. In most cases, the compared publications show similar values up to a magnitude of 10^{-3}. In comparison, our results differ slightly more, at a magnitude of 10^{-2}. However, the mesh convergence of our results suggests that better values could possibly be reached when considering even finer meshes M4, M5 etc. To emphasize this point, we apply a Richardson extrapolation with the discretization length h as a parameter. In our setting, we expect the error in C_d to scale linearly with h, since we are using a piecewise linear approximation of the sphere surface when computing the integral in Eq. (62). Furthermore, the discretization length h is divided by two in each refinement step. In this case, the Richardson extrapolation value K_{RE} of the drag correction factor using the obtained values for M2 and M3 yields $K_{\mathrm{RE}} = 2K_{\mathrm{M3}} - K_{\mathrm{M2}}$. The resulting values are shown in the RE column of Table 5 and they show a very good agreement with the compared publications, now deviating at a magnitude of 10^{-3} as well. However, as already observed by Knechtges [12], the results start to deviate more from each other with increasing Weissenberg numbers, especially for $Wi \geq 1.4$. Finally, it can be noted that all data in Table 5 agrees on the overall trend of decreasing K values for increasing Weissenberg numbers.

Table 6 shows K values for the eigenvalue-free logarithmic Giesekus model, evaluated for the same variety of Weissenberg numbers as before and mobility factors $\alpha \in \{0.1, 0.01, 0.001\}$. Our data agrees with the literature. We observe a noticeable mesh convergence towards the compared values. Furthermore, increasing Weissenberg numbers result in decreasing K values, which also agrees with the literature. For decreasing α values, an expected convergence of K towards the corresponding values in Table 5 is observed.

6. Conclusion and outlook

In this paper, we have shown how the $f(\mathrm{ad}\,\boldsymbol{\Psi})$-based formulation that was first introduced in [10] can be used to engineer an eigenvalue-free numerical algorithm for the log-conformation formulation.

In the course of our analysis, we first have proven the equivalence of this formulation to many different log-conformation formulations, including the original formulation by Fattal and Kupferman [1].

The new algorithm is in principle not tied to a specific discretization scheme of the resulting constitutive equation. However, in order to verify our algorithm, we have shown a working implementation in the RheoTool [13,14] framework, which is based on OpenFOAM® [28]. The resulting implementation was successfully validated on the confined cylinder and sedimenting sphere benchmarks.

For the future, we plan to bridge the performance gap of our prototypical implementation in comparison to the standard implementation by exploiting the matrix symmetries even more. Furthermore, we mostly see the application of this eigenvalue-free algorithm in areas that have so far been hindered by the eigenvalue decomposition. One is certainly bringing more of these heavy computations per finite volume cell onto the GPU. Another is that the algorithm facilitates the development of semi-implicit or fully implicit discretization schemes, in the same vein as [11] facilitated adoption of automatic differentiation methods in [46]. The latter may allow us to perform more efficient simulations with larger time-steps in the future.

Table 6
Final C_d values at $T = 30$ for the sedimenting sphere case, using the eigenvalue-free Giesekus formulation at different Weissenberg numbers. Again, three different mobility factors $\alpha \in \{0.1, 0.01, 0.001\}$ were considered.

Wi	K											
	$\alpha = 0.1$				$\alpha = 0.01$				$\alpha = 0.001$			
	M1	M2	M3	[12]	M1	M2	M3	[12]	M1	M2	M3	[12]
0.1	5.65893	5.74798	5.78425	5.82166	5.72846	5.82003	5.85734	5.89573	5.73688	5.82878	5.86622	5.90473
0.2	5.42343	5.50506	5.53791	5.57160	5.61367	5.70137	5.73675	5.77275	5.64289	5.73172	5.76754	5.80393
0.3	5.18896	5.26348	5.29312	5.32349	5.47733	5.56024	5.59341	5.62694	5.53296	5.61800	5.65191	5.68610
0.4	4.98481	5.05332	5.08021	5.10785	5.34386	5.42207	5.45317	5.48451	5.42763	5.50898	5.54108	5.57324
0.5	4.81155	4.87494	4.89950	4.92489	5.22228	5.29652	5.32581	5.35531	5.33417	5.41278	5.44327	5.47366
0.6	4.66432	4.72327	4.74586	4.76938	5.11488	5.18557	5.21331	5.24127	5.25443	5.33106	5.36019	5.38910
0.7	4.53822	4.59333	4.61424	4.63616	5.02073	5.08810	5.11447	5.14118	5.18772	5.26290	5.29093	5.31861
0.8	4.42928	4.48107	4.50052	4.52109	4.93780	5.00201	5.02716	5.05280	5.13245	5.20647	5.23366	5.26037
0.9	4.33445	4.38330	4.40151	4.42090	4.86418	4.92515	4.94915	4.97387	5.08704	5.15984	5.18638	5.21235
1.0	4.25128	4.29753	4.31466	4.33303	4.79803	4.85574	4.87859	4.90248	5.04996	5.12126	5.14722	5.17264
1.1	4.17782	4.22181	4.23799	4.25545	4.73806	4.79234	4.81404	4.83716	5.02010	5.08921	5.11451	5.13955
1.2	4.11255	4.15453	4.16988	4.18653	4.68319	4.73395	4.75444	4.77684	4.99613	5.06247	5.08689	5.11165
1.3	4.05423	4.09441	4.10902	4.12496	4.63261	4.67979	4.69906	4.72077	4.97718	5.04009	5.06323	5.08774
1.4	4.00185	4.04042	4.05436	4.06966	4.58567	4.62929	4.64735	4.66840	4.96206	5.02120	5.04264	5.06688
1.5	3.95458	3.99169	4.00503	4.01975	4.54196	4.58201	4.59889	4.61931	4.95004	5.00520	5.02443	5.04829

Declaration of competing interest

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

Data availability

No data was used for the research described in the article.

Acknowledgments

This Project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag. In addition, the third author thanks MAGMA Gießereitechnologie GmbH for the freedom to work on cutting-edge research topics.

Last but not least, the authors want to thank the reviewers of the Journal of Non-Newtonian Fluid Mechanics for their valuable comments.

Appendix. Poiseuille inflow conditions in the confined cylinder benchmark

As written in Section 5.1, we want to prescribe a fully developed Poiseuille flow at the inflow of the confined cylinder. This poses the question of whether an easy expression to specify $\boldsymbol{\Psi}$ exists. For $\boldsymbol{\tau}$ it is known that

$$\boldsymbol{\tau} = \begin{pmatrix} \tau_{xx} & \tau_{xy} \\ \tau_{xy} & 0 \end{pmatrix}, \tag{A.1}$$

with $\tau_{xx} = 2\lambda\mu_P \left(\partial_y u_x\right)^2$ and $\tau_{xy} = \mu_P \partial_y u_x$. The velocity $\mathbf{u}$ is given by

$$\mathbf{u} = \begin{pmatrix} \frac{3}{8}\bar{u}\left(4 - \frac{(y-2R)^2}{R^2}\right) \\ 0 \end{pmatrix}, \tag{A.2}$$

with mean inflow velocity $\bar{u} = 1\,\mathrm{m/s}$ in all benchmarks. The coordinate system is centered at the lower left corner of the confined cylinder domain (hence the $-2R$ term), as depicted in Fig. 2.

The corresponding conformation tensor is thus given by

$$\mathbf{C} = \mathbf{1} + \frac{\lambda}{\mu_P}\boldsymbol{\tau} = \mathbf{1} + \begin{pmatrix} 2l^2 & l \\ l & 0 \end{pmatrix}, \tag{A.3}$$

with $l = \lambda\,\partial_y u_x$.

We claim that $\boldsymbol{\Psi}$ is given by

$$\boldsymbol{\Psi} = \log \mathbf{C} = \frac{1}{2}\begin{pmatrix} p - ql^2/o & -ql/o \\ -ql/o & p + ql^2/o \end{pmatrix}, \tag{A.4}$$

with

$$o = \sqrt{l^2(1+l^2)} = |l|\sqrt{1+l^2} \tag{A.5}$$

$$p = \log(1+l^2) \tag{A.6}$$

$$q = \log\left(1 + 2(l^2 - o)\right) = 2\,\mathrm{arsinh}\,(-|l|)\,. \tag{A.7}$$

This is the same formulation as it was used for the actual computations in [11]. Even though the correct formula was used for computations in [11], a small error creeped into the formulas printed in [11], which unfortunately omitted factors of l in Ψ_{xy} and Ψ_{yy}. With this appendix we want to correct this error.

Coming to the proof, we split $\boldsymbol{\Psi}$ into two parts: one which encodes the traceful part and one traceless part

$$\boldsymbol{\Psi} = \frac{p}{2}\mathbf{1} + \mathbf{B}, \tag{A.8}$$

with

$$\mathbf{B} = \frac{ql}{2o}\begin{pmatrix} -l & -1 \\ -1 & l \end{pmatrix}. \tag{A.9}$$

Note that the identity matrix $\mathbf{1}$ and $\mathbf{B}$ obviously commute and thus allow us to compute the matrix exponential as two factors

$$\exp\boldsymbol{\Psi} = \exp\left(\frac{p}{2}\right)\exp\mathbf{B} \tag{A.10}$$

$$= \sqrt{1+l^2}\,\exp\mathbf{B}\,. \tag{A.11}$$

In order to compute $\exp\mathbf{B}$ it is helpful to see that the following identity holds

$$\mathbf{B}^2 = \frac{q^2}{4}\mathbf{1}\,. \tag{A.12}$$

From this, it follows immediately

$$\mathbf{B}^{2n} = \left(\frac{q}{2}\right)^{2n}\mathbf{1} \tag{A.13}$$

$$\mathbf{B}^{2n+1} = \left(\frac{q}{2}\right)^{2n+1}\frac{2}{q}\mathbf{B}\,. \tag{A.14}$$

Therefore, we can split the computation of $\exp\mathbf{B}$ into two summands

$$\exp\mathbf{B} = \sum_{n=0}^{\infty}\frac{1}{n!}\mathbf{B}^n \tag{A.15}$$

$$= \sum_{n=0}^{\infty}\frac{1}{(2n)!}\mathbf{B}^{2n} + \sum_{n=0}^{\infty}\frac{1}{(2n+1)!}\mathbf{B}^{2n+1} \tag{A.16}$$

$$= \cosh\left(\frac{q}{2}\right)\mathbf{1} + \sinh\left(\frac{q}{2}\right)\frac{2}{q}\mathbf{B}\,. \tag{A.17}$$

Together with the identity $\cosh(\mathrm{arsinh}(-|l|)) = \sqrt{1+l^2}$ it follows

$$\exp\mathbf{B} = \frac{1}{\sqrt{1+l^2}}\begin{pmatrix} 1+2l^2 & l \\ l & 1 \end{pmatrix}. \tag{A.18}$$

In total we obtain

$$\exp \boldsymbol{\Psi} = \begin{pmatrix} 1+2l^2 & l \\ l & 1 \end{pmatrix}, \tag{A.19}$$

which is what had to be proven.

References

[1] R. Fattal, R. Kupferman, Constitutive laws for the matrix-logarithm of the conformation tensor, J. Non-Newton. Fluid Mech. 123 (2) (2004) 281–285.
[2] M.A. Hulsen, A sufficient condition for a positive definite configuration tensor in differential models, J. Non-Newton. Fluid Mech. 38 (1) (1990) 93–100.
[3] G.C. Sarti, G. Marrucci, Thermomechanics of dilute polymer solutions: Multiple bead-spring model, Chem. Eng. Sci. 28 (4) (1973) 1053–1059.
[4] H.C. Booij, The energy storage in the rouse model in an arbitrary flow field, J. Chem. Phys. 80 (9) (1984) 4571–4572.
[5] M. Grmela, P.J. Carreau, Conformation tensor rheological models, J. Non-Newton. Fluid Mech. 23 (1987) 271–294.
[6] P. Wapperom, M.A. Hulsen, Thermodynamics of viscoelastic fluids: The temperature equation, J. Rheol. 42 (5) (1998) 999–1019.
[7] B. Jourdain, C. Le Bris, T. Lelièvre, F. Otto, Long-time asymptotics of a multiscale model for polymeric fluid flows, Arch. Ration. Mech. Anal. 181 (1) (2006) 97–148.
[8] D. Hu, T. Lelièvre, New entropy estimates for the Oldroyd-B model and related models, Commun. Math. Sci. 5 (4) (2007) 909–916.
[9] S. Boyaval, T. Lelièvre, C. Mangoubi, Free-energy-dissipative schemes for the Oldroyd-B model, ESAIM Math. Model. Numer. Anal. 43 (03) (2009) 523–561.
[10] P. Knechtges, Simulation of Viscoelastic Free-Surface Flows (Ph.D. thesis), RWTH Aachen, 2018, http://dx.doi.org/10.18154/RWTH-2018-229719.
[11] P. Knechtges, M. Behr, S. Elgeti, Fully-implicit log-conformation formulation of constitutive laws, J. Non-Newton. Fluid Mech. 214 (2014) 78–87, http://dx.doi.org/10.1016/j.jnnfm.2014.09.018, arXiv:1406.6988.
[12] P. Knechtges, The fully-implicit log-conformation formulation and its application to three-dimensional flows, J. Non-Newton. Fluid Mech. 223 (2015) 209–220, http://dx.doi.org/10.1016/j.jnnfm.2015.07.004, arXiv:1503.03863.
[13] F. Pimenta, M. Alves, RheoTool, 2016, https://github.com/fppimenta/rheoTool.
[14] F. Pimenta, M. Alves, Stabilization of an open-source finite-volume solver for viscoelastic fluid flows, J. Non-Newton. Fluid Mech. 239 (2017) 85–104.
[15] M. Alves, P. Oliveira, F. Pinho, Numerical methods for viscoelastic fluid flows, Annu. Rev. Fluid Mech. 53 (2021) 509–541.
[16] N. Balci, B. Thomases, M. Renardy, C.R. Doering, Symmetric factorization of the conformation tensor in viscoelastic fluid models, J. Non-Newton. Fluid Mech. 166 (11) (2011) 546–553.
[17] T. Vaithianathan, L.R. Collins, Numerical approach to simulating turbulent flow of a viscoelastic polymer solution, J. Comput. Phys. 187 (2003) 1–21.
[18] M.A. Carrozza, M.A. Hulsen, M. Hütter, P.D. Anderson, Viscoelastic fluid flow simulation using the contravariant deformation formulation, J. Non-Newton. Fluid Mech. 270 (2019) 23–35.
[19] C. Fernandes, A fully implicit log-conformation tensor coupled algorithm for the solution of incompressible non-isothermal viscoelastic flows, Polymers 14 (19) (2022) 4099.
[20] P. Saramito, On a modified non-singular log-conformation formulation for Johnson–Segalman viscoelastic fluids, J. Non-Newton. Fluid Mech. 211 (2014) 16–30, http://dx.doi.org/10.1016/j.jnnfm.2014.06.008.
[21] M.A. Hulsen, R. Fattal, R. Kupferman, Flow of viscoelastic fluids past a cylinder at high Weissenberg number: stabilized simulations using matrix logarithms, J. Non-Newton. Fluid Mech. 127 (1) (2005) 27–39.
[22] C. Moler, C. Van Loan, Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later, SIAM Rev. 45 (1) (2003) 3–49.
[23] G. Guennebaud, B. Jacob, et al., Eigen v3, 2010, http://eigen.tuxfamily.org.
[24] N.J. Higham, The scaling and squaring method for the matrix exponential revisited, SIAM J. Matrix Anal. Appl. 26 (4) (2005) 1179–1193.
[25] A.H. Al-Mohy, N.J. Higham, A new scaling and squaring algorithm for the matrix exponential, SIAM J. Matrix Anal. Appl. 31 (3) (2010) 970–989.
[26] N.J. Higham, Newton's method for the matrix square root, Math. Comp. 46 (174) (1986) 537–549.
[27] N.J. Higham, Functions of Matrices: Theory and Computation, SIAM, 2008.
[28] H.G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object orientated techniques, Comput. Phys. 12 (1998) 620–631.
[29] R. Guénette, M. Fortin, A new mixed finite element method for computing viscoelastic flows, J. Non-Newton. Fluid Mech. 60 (1) (1995) 27–52.
[30] J.P. Van Doormaal, G.D. Raithby, Enhancements of the SIMPLE method for predicting incompressible fluid flows, Numer. Heat Transfer 7 (2) (1984) 147–163.
[31] J.H. Ferziger, M. Perić, R.L. Street, Computational Methods for Fluid Dynamics, Springer, 2019.
[32] A. Syrakos, S. Varchanis, Y. Dimakopoulos, A. Goulas, J. Tsamopoulos, A critical analysis of some popular methods for the discretisation of the gradient operator in finite volume methods, Phys. Fluids 29 (12) (2017) 127103, http://dx.doi.org/10.1063/1.4997682.
[33] C.M. Rhie, W.-L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, AIAA J. 21 (11) (1983) 1525–1532.
[34] G. Negrini, N. Parolini, M. Verani, The Rhie–Chow stabilized box method for the Stokes problem, 2023, arXiv:2308.01059.
[35] S. Claus, T. Phillips, Viscoelastic flow around a confined cylinder using spectral/hp element methods, J. Non-Newton. Fluid Mech. 200 (2013) 131–146.
[36] Y. Fan, R. Tanner, N. Phan-Thien, Galerkin/least-square finite-element methods for steady viscoelastic flows, J. Non-Newton. Fluid Mech. 84 (2) (1999) 233–256.
[37] A.W. Liu, D.E. Bornside, R.C. Armstrong, R.A. Brown, Viscoelastic flow of polymer solutions around a periodic, linear array of cylinders: comparisons of predictions for microstructure and flow fields, J. Non-Newton. Fluid Mech. 77 (3) (1998) 153–190.
[38] J. Sun, M. Smith, R. Armstrong, R. Brown, Finite element method for viscoelastic flows based on the discrete adaptive viscoelastic stress splitting and the discontinuous Galerkin method: DAVSS-G/DG, J. Non-Newton. Fluid Mech. 86 (3) (1999) 281–307.
[39] A. Afonso, P. Oliveira, F. Pinho, M. Alves, The log-conformation tensor approach in the finite-volume method framework, J. Non-Newton. Fluid Mech. 157 (1) (2009) 55–65.
[40] W.J. Lunsmann, L. Genieser, R.C. Armstrong, R.A. Brown, Finite element analysis of steady viscoelastic flow around a sphere in a tube: calculations with constant viscosity models, J. Non-Newton. Fluid Mech. 48 (1) (1993) 63–99.
[41] R.G. Owens, T.N. Phillips, Steady viscoelastic flow past a sphere using spectral elements, Internat. J. Numer. Methods Engrg. 39 (9) (1996) 1517–1534.
[42] C. Chauvière, R.G. Owens, How accurate is your solution?: Error indicators for viscoelastic flow calculations, J. Non-Newton. Fluid Mech. 95 (1) (2000) 1–33.
[43] Y. Fan, Limiting behavior of the solutions of a falling sphere in a tube filled with viscoelastic fluids, J. Non-Newton. Fluid Mech. 110 (2) (2003) 77–102.
[44] G.G. Stokes, On the effect of the internal friction of fluids on the motion of pendulums, Trans. Camb. Phil. Soc. IX Part II (1856) 8–106.
[45] P.C.F. Pau, J. Berg, W. McMillan, Application of Stokes' law to ions in aqueous solution, J. Phys. Chem. 94 (6) (1990) 2671–2679.
[46] F. Zwicke, P. Knechtges, M. Behr, S. Elgeti, Automatic implementation of material laws: Jacobian calculation in a finite element code with TAPENADE, Comput. Math. Appl. 72 (11) (2016) 2808–2822.