

Finite Element Modelling of Two- and Three-Dimensional Viscoelastic Polymer Flows

R. Tenchev^{*†}, O. Harlen⁺, P.K. Jimack^{*} and M.A. Walkley^{*}

^{*}School of Computing, University of Leeds, UK

⁺School of Mathematics, University of Leeds, UK

[†]Currently at LUSAS, Kingston upon Thames, UK

Abstract

We present an overview of a recently-developed finite element software tool for the simulation of non-Newtonian fluid flows arising in viscoelastic polymer melts. The software is based upon the use of quantitative macroscopic constitutive laws, defining the evolution of polymer stress, which contributes to the total fluid stress in the flow. A wide range of melt rheologies may be considered in this manner, and three specific examples are considered in this paper. In addition to providing a detailed overview of the software, a number of sample numerical simulations are presented to illustrate the capabilities of the code.

Keywords: polymer melts, viscoelastic flow, finite element methods, arbitrary Lagrangian-Eulerian meshes.

1 Introduction

Understanding and controlling the flow of polymer melts is of great importance in polymer processing industries. Both the qualitative and the quantitative features of such flows are determined by the rheology of the melt, which in turn depends upon the underlying molecular structures. Consequently, there are a number of different length scales at which simulations may be undertaken in order to obtain macro-scale flow descriptions. In this research we work only at the continuum scale by making use of quantitative macroscopic constitutive laws which are based upon theory involving the physics of molecular alignment and entanglement (see, for example, [12, 13, 16]).

In the remainder of this section some further background is provided to the problems of interest. Specifically, this includes a description of the underlying mathematical models that are used and a brief discussion of relevant numerical methods, applied elsewhere in the literature. Section 2 then describes the algorithmic approach that is

used in this work: this includes a discussion of Eulerian versus Lagrangian meshes and an overview of the discretisation which follows. Section 3 goes into more detail of the software implementation, ranging from the problem specification, through the numerical methods, onto the presentation of results. The paper concludes with sections that illustrate the execution of the software via a selection of computational results, and a final section that draws brief conclusions and suggests areas for further research.

1.1 Governing Equations

For completeness, this section provides a full mathematical description of the viscoelastic models considered by our software. Although the equations are provided in full we do not attempt to justify them here: instead we provide sample references which go more deeply into the model derivations.

1.1.1 Stokes' equations

The flows considered in this work comprise of solutions of high molecular weight polymers at relatively low velocities. This leads to a model of a high viscosity fluid with relatively small transient and inertial effects. Hence the flow can be mathematically modelled by Stokes' equations:

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{b} = \mathbf{0}; \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0; \quad (2)$$

where $\mathbf{u} = (u, v, w)$ represents the velocity field, \mathbf{b} represents the body forces present and $\boldsymbol{\sigma}$ is the viscoelastic stress tensor. This stress tensor may be decomposed into Newtonian and non-Newtonian parts, $\boldsymbol{\sigma}_N$ and \mathbf{T} respectively, with $\boldsymbol{\sigma}_N$ defined as

$$\boldsymbol{\sigma}_N = -p\mathbf{I} + \mu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) \quad (3)$$

where p is the pressure field and μ is the Newtonian viscosity.

The additional polymer stress \mathbf{T} is determined by the constitutive equation for the polymer and is considered in this work as an additional body force term in the momentum equation (1), which therefore re-arranges as:

$$-\nabla \cdot \boldsymbol{\sigma}_N = \mathbf{b} + \nabla \cdot \mathbf{T}. \quad (4)$$

The three viscoelastic constitutive models considered in this work are described in the following sub-sections.

1.1.2 An Oldroyd-B fluid

The Oldroyd-B model is a standard, single-mode, viscoelastic fluid model [2] for which the polymeric stress tensor \mathbf{T} is given by

$$\mathbf{T} = G(\mathbf{A} - \mathbf{I}), \quad (5)$$

where G is the elastic modulus of the fluid and the polymer deformation tensor \mathbf{A} is assumed to obey the equation

$$\frac{\partial \mathbf{A}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{A} - \mathbf{A} \cdot \nabla \mathbf{u} - (\nabla \mathbf{u})^T \cdot \mathbf{A} = \frac{-1}{\tau} (\mathbf{A} - \mathbf{I}), \quad (6)$$

with τ the relaxation time for the polymer. For a given problem the parameters G and τ must be specified to completely define the polymeric behaviour.

1.1.3 A Rolie-Poly fluid

The Rolie-Poly model [12] is a multi-mode constitutive law for linear (non-branched) polymers derived from a simplification of the tube theory for entangled polymers [13]. The stress tensor \mathbf{T} for an n -mode model is given by

$$\mathbf{T} = \sum_{i=1}^n G_i (\mathbf{A}_i - \mathbf{I}), \quad (7)$$

where G_i is the elastic modulus for mode i . The polymer deformation tensor for each mode, \mathbf{A}_i , satisfies

$$\frac{\partial \mathbf{A}_i}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{A}_i - \mathbf{A}_i \cdot \nabla \mathbf{u} - (\nabla \mathbf{u})^T \cdot \mathbf{A}_i = \frac{-1}{\tau_{A,i}} \mathbf{A}_i + \frac{1}{\tau_{d,i}^{eff}} \mathbf{I}. \quad (8)$$

This model includes two relaxation times for each mode, which satisfy the following equations:

$$\frac{1}{\tau_{R,i}^*} = 2 \frac{\left(1 - \sqrt{3/tr \mathbf{A}_i}\right)}{\tau_{R,i}}; \quad (9)$$

$$\frac{1}{\tau_{d,i}^{eff}} = \frac{1}{\tau_{d,i}} + \frac{1}{\tau_{R,i}^*} \beta_i \left(\frac{tr \mathbf{A}_i}{3}\right)^{\delta_i}; \quad (10)$$

$$\frac{1}{\tau_{A,i}} = \frac{1}{\tau_{d,i}^{eff}} + \frac{1}{\tau_{R,i}^*}. \quad (11)$$

The reptation time $\tau_{d,i}$, Rouse time $\tau_{R,i}$, constraint release parameter β_i , tube deformation parameter δ_i and modulus G_i must be specified for each mode to completely define the behaviour of the fluid for a given problem.

1.1.4 A pom-pom fluid

The pom-pom model is a multi-mode constitutive law for branched polymers derived from the tube theory applied to star and H-shaped polymers [16]. The stress tensor \mathbf{T} for an n -mode model is given by

$$\mathbf{T} = \sum_{i=1}^n \frac{G_i \lambda_i^2}{(tr \mathbf{A}_i / 3)} (\mathbf{A}_i - \mathbf{I}), \quad (12)$$

with elastic modulus G_i and stretch parameter λ_i given for each mode. The polymer deformation tensor for each mode, \mathbf{A}_i evolves according to a similar equation to Oldroyd-B,

$$\frac{\partial \mathbf{A}_i}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{A}_i - \mathbf{A}_i \cdot \nabla \mathbf{u} - (\nabla \mathbf{u})^T \cdot \mathbf{A}_i = \frac{-1}{\tau_{b,i}} (\mathbf{A}_i - \mathbf{I}), \quad (13)$$

but the stretch parameters λ_i in the stress tensor (12) are governed by additional differential equations:

$$\frac{\partial \lambda_i}{\partial t} + \mathbf{u} \cdot \nabla \lambda_i = \frac{\lambda_i}{tr \mathbf{A}_i} (\mathbf{A}_i : \nabla \mathbf{u}) - \frac{\lambda_i - 1}{\tau_{s,i}} e^{\left(2 \frac{\lambda_i - 1}{q_i}\right)}, \quad (14)$$

where q_i represents the maximum stretch, with λ_i satisfying $0 < \lambda_i \leq q_i$, and $\tau_{b,i}$ and $\tau_{s,i}$ are, respectively, the alignment and stretch relaxation times. For a given problem the parameters G_i , $\tau_{b,i}$, $\tau_{s,i}$ and q_i must be specified for each mode to completely define the polymeric behaviour.

1.2 Numerical Methods

There is a very substantial body of previous work that has developed numerical methods for the simulation of viscoelastic, especially Oldroyd-B, fluids. Indeed, the literature is far too vast for us to attempt any sort of review here. Instead, we refer to a selection of such work, based upon finite volume methods in two dimensions [1] or three dimensions [20], spectral methods [17], stabilised finite element methods [6] or mixed finite element methods [9]. These papers, and the many references cited therein, provide an illustration of the richness of this field.

The Rolie-Poly and pom-pom fluids, introduced above, have been less widely studied numerically. Consequently, we confine this discussion to one such implementation, [4], and refer the interested reader to references cited therein for a broader selection of numerical approaches. The *FlowSolve* code described in [4] is based upon a two-dimensional Lagrangian formulation of the problem, using mixed finite elements on an adapting mesh of triangular elements. The mixed formulation for the Stokes flow is inherently stable [5], and the fact that the mesh evolves with the flow ensures that the polymer transport equations are also trivial to integrate in time in a stable manner. The primary drawbacks of this approach are the need to frequently re-connect the mesh to avoid tangling (this is relatively cheap and easy in two dimensions but would be much more problematic in three), and the need to ensure that the initial mesh has sufficient nodes ‘‘upstream’’ to ensure a satisfactory mesh resolution throughout the simulation (or else a mechanism is required to introduce new nodes at the inflow boundary and remove nodes from the outflow boundary automatically).

As described in the next section, in this work we have chosen an Eulerian approach ahead of the Lagrangian technique. This is primarily due to the technical mesh-quality issues that arise with the latter in three dimensions. In order to allow domains that evolve with time, this is then extended to an arbitrary Lagrangian-Eulerian (ALE) formulation.

2 Software Overview

This section provides a very brief description of the software that is the subject of this paper. More details concerning the implementation are then provided in Section 3. Note that since the discretisation used in this work is based upon finite elements, the use of finite element meshes and spaces will be assumed throughout.

2.1 Eulerian and Lagrangian Meshes

As described in the previous section, there are two basic classes of approach that may be taken. In the Lagrangian approach the points of the finite element mesh move at a velocity that is equal to the instantaneous local fluid velocity. This has significant advantages when simulating the transport of the polymer stress, as in equations (6), (8) or (13) for example. By selecting an Eulerian approach there is a need to consider carefully the stability of any discretisation of these equations, and this matter is discussed further in Section 3.3 below. Clearly, a significant advantage of the Eulerian approach is that, once an initial high-quality mesh has been obtained for the domain of interest, this may be re-used for all time. This leads to the possibility of significant savings if it can be exploited in the solution procedure. In particular, in Section 3 we describe a penalty formulation that allows the pressure to be eliminated from the Stokes equations and the resulting discrete system to be factorised using a sparse or a banded LU factorisation [11]. This permits subsequent Stokes solves (required at each time step) to be completed at the cost of just a forward and a backward substitution, which is extremely efficient.

In the case where the fluid domain is time-dependent, due to the presence of a moving boundary or a free surface for example, the Eulerian approach requires some form of modification. In this work we have opted to exploit an ALE formulation. The principle behind this approach is that the mesh at the boundary of the domain should move with the correct normal velocity, and the velocity of the mesh elsewhere is selected primarily in order to maintain the local quality. In order to achieve this we make use of a simple Laplacian smoothing procedure which attempts to keep the interior nodes from coming together or the mesh itself from becoming tangled [18]. Having defined an appropriate velocity field for the mesh, \mathbf{V} say, it is then necessary to modify the second terms appearing in the equations (6), (8) (13) and (14): replacing the advective velocity \mathbf{u} by $(\mathbf{u} - \mathbf{V})$.

2.2 Structure of the Software

The structure of each time step within the code may be described as follows. Given an initial stress field \mathbf{T} , solve equations (4) and (2) for the resulting velocity field \mathbf{u} . Then use this instantaneous velocity field in equations (6), (8) or (13) in order to take a time step. The value of the stress tensor at the end of the time step may then be used to update the right-hand side of (4) at the start of the next time step.

Note that in three dimensions (4) and (2) represent 4 scalar partial differential equations (PDEs) whilst (8), for example, represents a further $6n$ PDEs (since each tensor A_i is symmetric). Fortunately, each of the n sets of 6 PDEs given by (8) is independent of the others in the models that are considered here, and may therefore be solved independently.

As well as the time-stepping routines that lie at the core of the software, a number of pre- and post-processing routines are also provided. The former are described in Section 3.1 and include the ability to generate geometries, meshes and boundary conditions, as well as specifying fluid properties. The latter are described in Section 3.4, which provides an overview of the visualisation capabilities. The software itself is embedded in a user-friendly run-time environment that allows real-time visualisation of results and parameters to be modified during a computation. Figure 1 provides a sample snapshot of this environment.

3 Implementation Details

Having provided an overview of the software in the previous section, we now go on to describe a number of the key features in a little more detail. Space does not permit complete description of all aspects of the implementation and so the primary aim here is to present a discussion of selected features.

3.1 Problem Specification

The particular problem that is to be solved in any given run is specified by a number of components, such as its geometry, boundary conditions and fluid model parameters. In order to undertake a computation, additional numerical properties must also be specified, for example: the finite element mesh, the types of element to be used and the time-step size to take. Each of these attributes, and many more optional ones that will not be discussed here, must be defined through a single data file which is read by the program upon execution. Rather than describe the precise format of the data files here we focus on their key components.

The geometry is specified by the use of one or more quadrilateral or octahedral “super-elements” in 2-d or 3-d respectively (see Fig. 1a for example). These super-elements may have curved as well as planar edges and faces. They are also used as an integral part of the built-in mesh generator, which is based upon a block-structured mesh (as illustrated in Fig. 2a). By specifying the mesh dimensions within each block the user is able to build a mesh covering the entire geometry. It is also possible to grade these structured meshes towards, or away from, selected edges or faces: tools are provided to assist with ensuring that the mesh remains conforming between neighbouring blocks.

Having specified the geometry and the mesh for a given flow problem it is also necessary to prescribe the boundary conditions that should be used to solve the flow.

Different regions of the boundary may be identified in a variety of possible ways however the simplest is to use the faces of the super-elements. For a given portion of the boundary one of a number of conditions may be selected. The most widely used of these are briefly summarised in the following list.

- No slip boundary at a solid wall. This simply involves prescribing the tangential (no slip) and normal (solid wall) components of the velocity to be zero.
- Inflow boundary. Rather similar to the above, this permits a specific fluid velocity profile to be prescribed. In this case however the normal flow is non-zero and into the domain.
- Developed inflow boundary. Frequently, especially for non-Newtonian fluids, the precise velocity profile of a fully developed flow is not known *a priori* however it is desirable for such a flow to be the input to the region of interest. This condition is imposed by requiring the user to define an estimate to the velocity profile and a plane that is parallel to the inflow boundary but a little way downstream of it. Then, at each time step the inflow velocities imposed are those that were recorded at the parallel plane for the previous time step. After a number of such time steps a fully developed flow is usually obtained.
- Outflow boundary. Here we allow use of the “no boundary condition” outflow boundary as described in [8]. This is designed to reduce disturbances in the solution at the outflow boundary and is very successful at doing so.
- Symmetry boundary condition. This is extremely important, especially in three dimensions, for reducing the size of a computational problem whose solution contains known symmetries. This condition applies to all of the unknowns and may be applied around a line in two dimensions or a plane in three dimensions.
- Free-surface condition. In the case where there is a free surface then it is necessary that the geometry is updated in a manner that is consistent with the fluid velocity at the free surface. This may be undertaken in a Lagrangian manner (i.e. the nodes on the free surface move with their fluid velocity), based upon the normal velocity only (i.e. the nodes on the free surface move perpendicular to the free surface with a consistent velocity), or based upon the velocity in a prescribed direction (e.g. through the use of predetermined “spines”, [10]). In addition, there is a stress boundary condition which modifies the surface forces according to the local curvature and the surface tension (see, for example, [22]).

Other properties of a given problem that must be specified include the size and number of the time steps to be taken, the frequency at which outputs of the simulation should be written to file, and the properties of the fluid that is to be considered. For the latter, the user is able to select between each of the models described in Subsection 1.1, as well as a Newtonian fluid, and then to specify the number of modes to be used (for the Rolie-Poly and the pom-pom fluids) and the parameters that must be prescribed to uniquely define each of the governing equations.

3.2 Finite Element Solution

As described in Subsection 2.2, the equations for the evolution of the polymer deformation tensor (equations (6), (8) or (13)), as well as equation (14) in the pom-pom case, are solved using implicit time-stepping but with the velocity field frozen at the value from the beginning of the time step. This requires the use of a stabilised finite element method (see the following subsection) and yields sparse non-symmetric systems of linear algebraic equations that must be solved at each time step. In all of the calculations included in this paper piecewise bilinear or trilinear finite elements have been used for the trial spaces in two or three dimensions respectively. The resulting algebraic equations have been solved using the SPARSKIT library, [19], typically selecting the BiCGStab solver with an incomplete ILU(0) preconditioner.

Once the update to the polymer deformation field has been computed over the time step, a modified forcing term on the right-hand side of (4) may be computed and the Stokes' equations resolved. Two different approaches to solving these equations have been implemented in this work: one based upon a direct solver and one based upon an iterative method. The direct approach has significant advantages when a problem on a fixed domain, and therefore a fixed mesh, is solved since it is able to exploit the fact that only the right-hand side of (4) changes at each time step. The relative merits of the two approaches in the case of an evolving ALE mesh are less clear however, depending primarily on the quality of the preconditioner that can be obtained for the iterative solver. Both approaches are briefly outlined, although only the former is used for the results shown here (in fact the results obtained by the two methods are identical to a number of significant digits: the precise number of digits depending primarily on the choice of the penalty and convergence parameters that are used in the respective methods).

In order to develop a direct solver for the finite element discretisation of (4) it is desirable that the system be as small as possible, preferably with a small band-width. In order to reduce the size of the discrete problem we first apply a penalty approach in order to eliminate the pressure from the Stokes' system. This is achieved by replacing the incompressibility condition (2) with a new relation of the form

$$\nabla \cdot \mathbf{u} = -\frac{1}{\chi} p, \quad (15)$$

where the penalty parameter χ is suitably large (in particular $\chi \gg \mu$). Substituting (15) into (4) then yields

$$-\nabla \cdot \left(\mu \nabla \mathbf{u} + \chi (\nabla \mathbf{u})^T \right) = \mathbf{b} + \nabla \cdot \mathbf{T}, \quad (16)$$

which is the form used for the finite element discretisation. Note that the discrete system, obtained using a standard Galerkin approach, involves only the velocity unknowns. Due to the large penalty parameter it is not suitable for iterative solution (it is poorly conditioned) but, so long as the choice of penalty value is not too large for the precision of floating point arithmetic that is used, it is well suited to a banded or

a sparse direct solution method. In this work we use a banded solver from the IMSL library but any other such solver is suitable (e.g. [11]). The key requirement is that the LU factorisation of the Stokes' matrix should be stored, either in primary memory or on disk (depending upon the size of the problem), so that subsequent Stokes' solvers require only forward and backward substitutions.

An iterative solver based upon a mixed finite element formulation of the Stokes problem has also been implemented. This follows the standard approach that is developed in detail in [5], for example. In its simplest form a stable pair of finite element spaces must be selected for the velocities and pressure respectively: we use bi-quadratic/bilinear and triquadratic/trilinear in two and three dimensions respectively. This leads to a coupled discrete system for the velocity and the pressure unknowns that is symmetric but indefinite. Suitable choices of iterative solver are described in [5] however the effectiveness of this solver depends critically upon the efficiency of the preconditioner that is used. Highly efficient preconditioners are possible however implementation of an optimal iterative solver is not a task that we have so far undertaken.

3.3 Stabilisation

A number of stability issues need to be addressed in this work, some of which have already been introduced in the sections above. These include the stability of the finite element discretisation, both for hyperbolic equations or for mixed systems (such as the Stokes' equations), and the requirement to ensure that certain quantities remain within predetermined bounds. Each of these issues is discussed in turn.

The Galerkin method is not appropriate for the discretisation of purely hyperbolic equations such as (6), (8), (13) or (14). In this work we make use of a Petrov-Galerkin approach based upon streamline upwinding [14]. This has a stabilising effect through the introduction of a minimal amount of diffusion along the streamlines. We also permit the mixed finite element solution of the Stokes' equations to be undertaken using unstable pairings of finite element spaces for the velocities and the pressures, typically piecewise bilinear (or trilinear in three dimensions) spaces for all unknowns. Here the stabilisation may be introduced at the solution stage, as described in [5] for example.

Certain components of the solutions that are computed have specific physical constraints whose violation can lead to further instabilities in the simulations. Examples include the requirement that all components of the deformation tensors remain positive, or the upper and lower bounds on the stretch parameters λ_i in the pom-pom model. There are a number of possible ways of ensuring that these bounds are respected by the numerical solution, two of which are used within our code. The first of these is to put a large penalty term on the right-hand side of each evolution equation, which only becomes active when a constraint for the degree of freedom in question is violated: this acts as a crude barrier and is effective when a low order time integration scheme is being used, as is typically the case here. The second approach is to

prevent the constraint from being violated in the first place by ramping up the penalty term gradually as the constraint is being approached from within the set of allowed values. This is more appropriate when it is essential that the constraint is never violated, even for a single time step, or when a higher degree time integration scheme is being used. It does tend to over-penalise however, meaning that the limiting value is never reached.

3.4 User Interface

We conclude this section with a brief description of the user interface that is provided with our software. This is an optional Windows interface and the solver may also be run without it, under Linux for example. In the latter case a choice of output formats are available for compatibility with the visualisation tools GMV [7] and Tecplot [21]. Indeed, some of the results presented in the following sections exploit this option.

The user interface that is provided comes with substantially more functionality than can be described here. We provide a brief flavour of this however in the following paragraphs on the pre-processing and the solution phases respectively.

The preprocessing phase allows the user to undertake a number of tasks that are designed to assist with the setting up of the problem. These include options to: plot the super elements and their associated numberings, as illustrated in Fig. 1(a), plot the mesh and its dimensions, plot boundary faces of different types, plot and display node numbers, show different types of boundary condition, etc. All of these facilities allow the user to check that the problem has been properly defined, and all objects may be viewed from different positions and with varying levels of magnification. Once the user is satisfied that the problem is correctly defined they may then begin the execution.

As a run is executing the graphical output is updated after each time step, regardless of the frequency with which data is being saved to file. It is possible to pause the execution after any time step and then to resume, possibly having changed certain parameters (e.g. the time step size). There is a large choice of solution fields that may be displayed: any of the velocity components, illustrated in Fig. 1(b), any of the deformation components (either for individual modes or summed over all modes), the parameters λ_i , pressure, etc. The default is to use colouring for contours however discrete contour lines may be selected instead. In addition, streamlines, velocity vectors and birefringence patterns may be displayed. As with the geometry and the mesh, each of the above displays may be rotated or magnified as required.

4 Computational Results

The following sections describe numerical experiments that illustrate the functionality of the software. Three rheologies, given in Tables 1-3, are used as examples of the Oldroyd-B, Rolie-Poly and pom-pom fluid models.

G	τ
1.36667	0.3

Table 1: Oldroyd-B fluid model: $\mu = 0.59$

i	G_i	$\tau_{d,i}$	$\tau_{R,i}$	β_i	δ_i
1	1286.0e-6	2.3780	0.05263	0.0	-0.5
2	4407.0e-6	0.7519	0.03008	0.0	-0.5
3	11639.0e-6	0.2378	0.01530	0.0	-0.5
4	21043.0e-6	0.0752	0.00752	0.0	-0.5

Table 2: Rolie-Poly fluid model: $\mu = 1123.0e - 6$, $n = 4$

4.1 Two-Dimensional Flow Around a Cylinder

Flow past a solid cylinder is computed in 2d for the Rolie-Poly fluid with a unit radius cylinder and a solid wall at $y = \pm 2$. A symmetric half of the domain is solved and the inflow and outflow boundaries are placed at $x = -16$ and $x = 10$ respectively. Fig. 2 shows the super-elements employed and the bilinear quadrilateral mesh that is produced from the super-element description. The computed steady-state solution shows the expected acceleration due to the contraction around the cylinder and also the increase in polymer stress on the cylinder surface and on the exterior wall.

4.2 Two-Dimensional Contraction Flow

The contraction geometry, shown in Fig. 3, is modelled with the Rolie-Poly fluid. A symmetric half of the domain is solved. The re-entrant corner is rounded to prevent the formation of a stress-singularity at, or just after, this point. The streamlines clearly show the presence of a recirculation region prior to the contraction. The size and shape of this region is strongly determined by the fluid rheology. The stress birefringence pattern represents visually regions where the stress gradient is high. It can be seen here that there is stress build up on the centreline before the contraction and a subsequent release farther downstream. At the wall there is a high stress gradient at the corner as expected.

i	G_i	$\tau_{b,i}$	$\tau_{b,i}/\tau_{s,i}$	q_i
1	8314.05e-6	0.393901	5.0	3.0
2	4590.42e-6	1.18237	5.0	4.0
3	1907.74e-6	3.73898	5.0	4.0
4	612.395e-6	11.8237	4.0	7.0
5	159.889e-6	37.3898	3.0	8.0
6	26.2254e-6	118.237	2.0	9.0

Table 3: Pom-pom fluid model: $\mu = 5122.9157e - 6$, $n = 6$

4.3 Three-Dimensional Contraction-Expansion Flow

The contraction-expansion flow shown in Fig. 4 is modelled with the Oldroyd-B fluid. The mesh for a symmetric quarter of the domain is visualised with GMV [7] in Fig. 4(a). Contours of velocity magnitude are shown in Fig. 4(b) with velocity vectors superimposed. A small recirculation region is visible in the corners above the contraction region.

4.4 Three-Dimensional Y-shaped Channel Flow

This Rolie-Poly flow comprises of a three-dimensional domain with two converging channels meeting at an angled junction. The re-entrant corner is rounded to prevent a stress singularity at that point. A symmetric half of the domain is visualised through GMV [7] in Fig. 5(a) coloured with contours of velocity magnitude. The acceleration caused by the convergence into one channel is clearly seen. Fig. 5(b) shows a component of the total fluid stress, σ in Eq. (1), illustrating the peak in stress that is produced at the re-entrant corner.

4.5 Three-Dimensional Flow Past a Cylinder

The flow of a typical pom-pom fluid through a thin channel with a cylindrical obstruction is depicted in Fig. 6 (in fact this calculation is undertaken using just a single pom-pom mode, rather than the 6-mode model given in Table 3). Qualitative differences in the stress pattern for this fluid are obvious in comparison with the two-dimensional simulation shown in Fig. 2. It is also possible to visualize the additional λ field for each mode, satisfying the equations (14).

5 Further Enhancements

The previous sections presents some representative results for flows in relatively simple, fixed, geometries in two and three dimensions. We now illustrate some of the enhancements of the software, to allow more complex geometries which may be time-dependent.

5.1 Flow With Rigid Particles

The direct simulation of particle suspensions represents a very important class of problem in the area of polymer processing [15]. We consider a unit-cell from an infinite domain with the particle size varied to produce different volume-fractions. The particle undergoes shear and a rotation is induced, which can be simulated without moving the mesh. Two-dimensional results for the Rolie-Poly fluid are shown in Fig 7(a) and the extension of these ideas to the Oldroyd-B fluid in three-dimensions in Fig. 7(b).

In each case the rotation rates obtained converge to the theoretical results as volume-fraction is reduced.

5.2 Arbitrary Lagrangian Eulerian Implementation

Shear flow around a solid elliptical particle is used to illustrate the ALE remeshing possible within the software. The shear flow drives a periodic rotation of the particle that requires mesh movement due to the variation in the radius. Results in Fig. 8 illustrate the mesh at two different times in the simulation. It can be seen clearly how the ALE adaptivity copes with the particle rotation. The mesh is continuously deformed through a Laplacian smoothing approach in order to maintain the quality of the quadrilateral elements. Contours of the vertical velocity component illustrate the variation in the flow.

5.3 Free-Surface Flows

Die-swell represents an important industrial process and, although the current software implementation is designed only for problems with a relatively small boundary deformation, it is possible to model such cases. A Rolie-Poly fluid is used and initially the fluid surface is horizontal. A symmetric half of the domain is used and the mesh is refined near the outlet. The spine method is used, as described in Section 3.1, [10], where the mesh motion is limited to vertical displacements, driven by the free surface motion. Results in Fig. 9 show contours of u-velocity at three times in the simulation. Initially the free surface bulges at the outlet but the magnitude of this expansion decreases slightly as a steady state is reached.

6 Conclusions

We have presented an introduction to a new software tool that has been developed to allow the numerical solution of two- and three-dimensional viscoelastic flow problems. The tool allows a variety of constitutive laws to be considered for fluids within a wide range of geometries, including those with moving or free boundaries. A selection of numerical results has been presented in order to convey the versatility of the software.

Current work is ongoing to benchmark the results produced by this code against those obtained using other computational tools and from experimental observation and measurement. Preliminary results suggest that the robustness of the software is matched by its accuracy and computational efficiency.

The next stage of the work will involve applying the code for the simulation of a variety of real polymers in flow regimes of practical interest and importance. This is likely to involve further development, for example to allow more substantial evolution of free surfaces in both two and three dimensions. For the ALE formulation in three

dimensions it is also likely to be beneficial to improve the efficiency of the iterative Stokes' solver that has been implemented, through the use of optimal preconditioning for example [5].

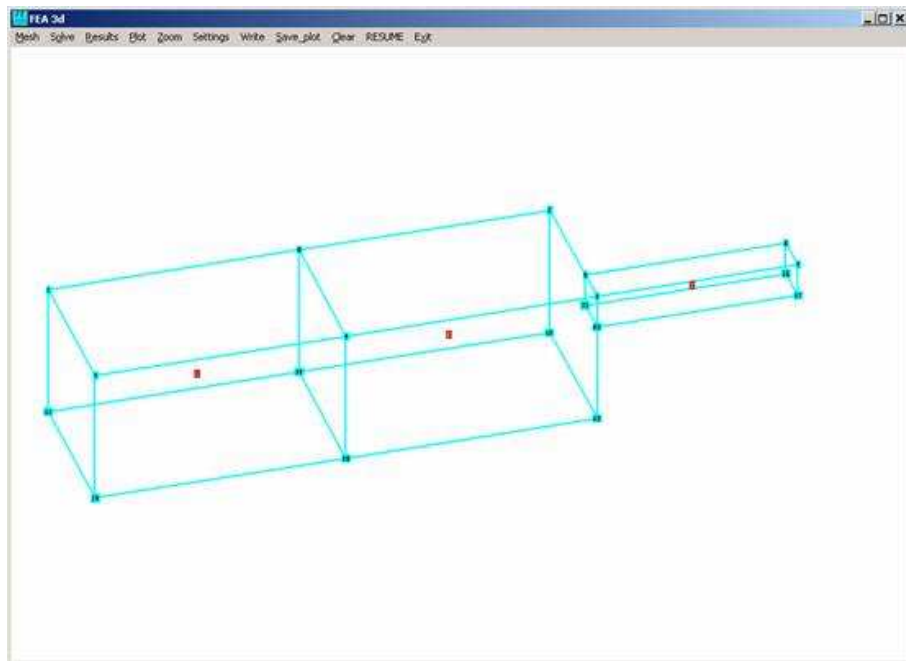
Acknowledgements

This research was undertaken as part of EPSRC grant GR/T11807/01.

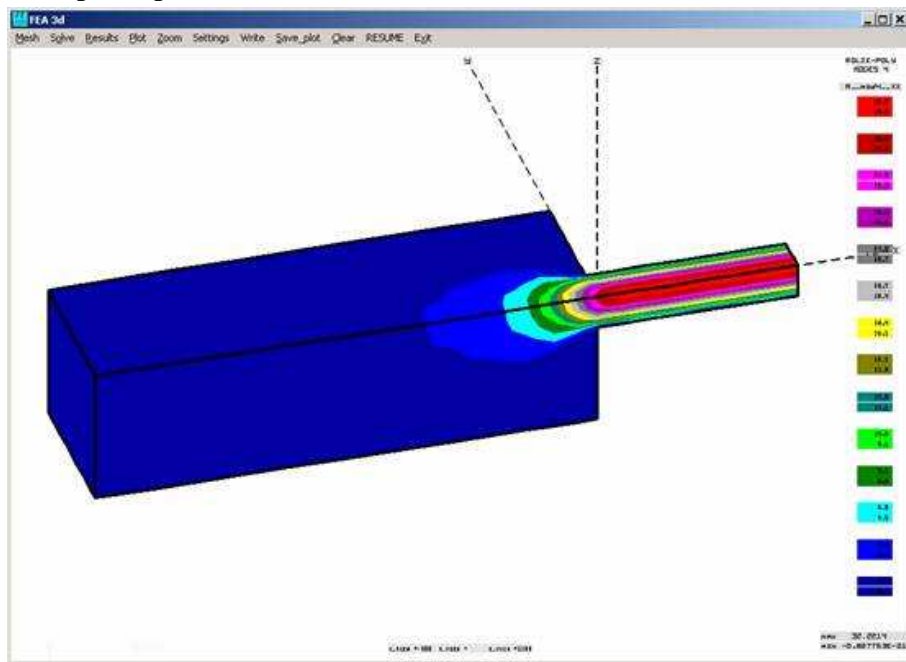
References

- [1] M.A. Alves, P.J. Oliveira and F.T. Pinho. The flow of viscoelastic fluids past a cylinder: finite-volume high-resolution methods. *J. Non-Newtonian Fluid Mech.*, 97:207–232, 2001.
- [2] H.A. Barnes, J.F. Hutton and K. Walters. *An Introduction to Rheology*. Elsevier, 4th edn, 1996.
- [3] F. Bassi and s. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *J. Comput. Phys*, 131:267–279, 1997.
- [4] G.B. Bishko, O.G. Harlen, T.C.B. McLeish and T.M. Nicholson. Numerical simulation of the transient flow of branched polymer melts through a planar contraction using the 'pom-pom' model. *J. Non-Newtonian Fluid Mech.*, 82:255–273, 1999.
- [5] H.C. Elman, D.J. Silvester and A.J. Wathen. *Finite Elements and Fast Iterative Solvers with applications in incompressible fluid dynamics*, OUP, 2005.
- [6] Y.R. Fan, R.I. Tanner and N. Phan-Thien. Galerkin/least-squares finite-element methods for steady viscoelastic flows. *J. Non-Newtonian Fluid Mech.*, 84:233–256, 1999.
- [7] GMV - The General Mesh Viewer. <http://laws.lanl.gov/XCM/gmv/>.
- [8] D.F. Griffiths. The “no boundary condition” outflow boundary condition. *Int. J. Numer. Meths. Fluids*, 24:393–411, 1997.
- [9] R. Guenette and M. Fortin. A new mixed finite element method for computing viscoelastic flows. *J. Non-Newtonian Fluid Mech.*, 60:27–52, 1995.
- [10] S.F. Kistler and L.E. Scriven. Coating flows. In *Computational Analysis of Polymer Processing*, London Applied Science Publishers (eds. J.R.A. Perason & S.M. Richardson), 243–299, 1983.
- [11] X.Y.S. Li. An overview of SuperLU: Algorithms, implementation, and user interface. *ACM Trans. on Math. Software* 31:302–325, 2005.
- [12] A.E. Likhtman and R. Graham. A simple constitutive equation for linear polymer melts derived from molecular theory: the Rolie-Poly model. *J. Non-Newtonian Fluid Mech.*, 114(1):1–12, 2003.
- [13] A.E. Likhtman and T.C.B. McLeish. Quantitative theory for linear dynamics of linear entangled polymers. *Macromol.* 35:6332–6343, 2002.

- [14] X.-L. Luo and R.I. Tanner. A decoupled finite element streamline-upwind scheme for viscoelastic flow problems. *J. Non-Newtonian Fluid Mech.*, 31:143-162, 1989.
- [15] A. Malidi and O. Harlen. A Lagrangian Simulation Method for Suspensions in Viscoelastic Fluids. *Proceedings of ECCOMAS CFD*, 2006.
- [16] T.C.B. McLeish and R.G. Larson. Molecular constitutive equations for a class of branched polymers: the pom-pom polymer. *J. Rheol.* 42(1):81–110, 1998.
- [17] R.G. Owens, C. Chauvire and T. N. Philips. A locally-upwinded spectral technique (LUST) for viscoelastic flows. *J. Non-Newtonian Fluid Mech.*, 108:49–71, 2002.
- [18] R.C. Peterson, P.K. Jimack and M.A. Kelmanson. The solution of two-dimensional free-surface problems using automatic mesh generation. *Int. J. Numer. Meths. Fluids*, 31:937–960, 1999.
- [19] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS, 1996.
- [20] M. Sahin and H.J. Wilson. A semi-staggered dilation-free finite volume method for the numerical solution of viscoelastic fluid flows on all-hexahedral elements. *J. Non-Newtonian Fluid Mech.*, 147:79-91, 2007.
- [21] Tecplot - CFD Post Processing, Plotting, Graphing and Data Visualization Software. <http://www.tecplot.com/>.
- [22] M.A. Walkley, P.H. Gaskell, P.K. Jimack, M.A. Kelmanson and J.L. Summers. Finite element simulation of three-dimensional free-surface flow problems. *J. Sci. Comput.*, 24:147-162, 2005.

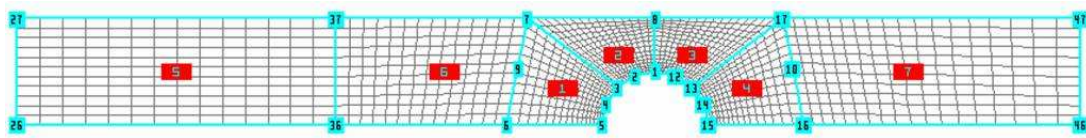


(a) Input super-element mesh

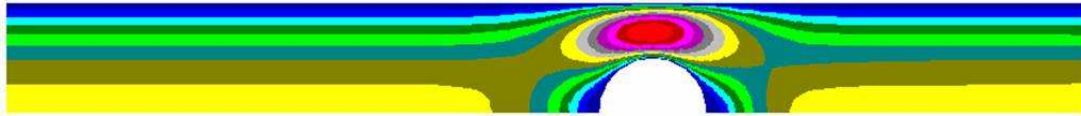


(b) Computed velocity profile

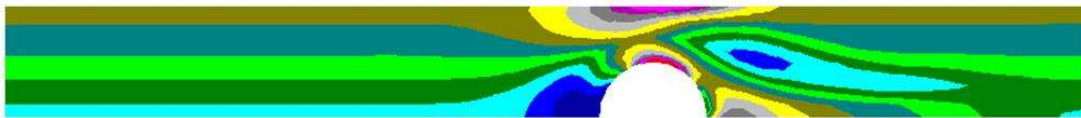
Figure 1: The Windows program interface



(a) Super-element mesh

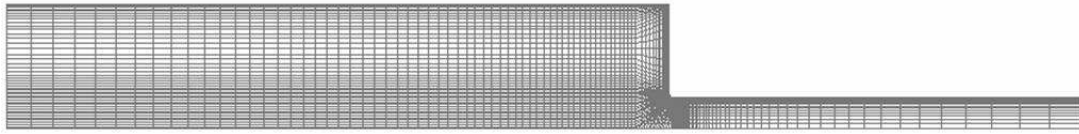


(b) U velocity

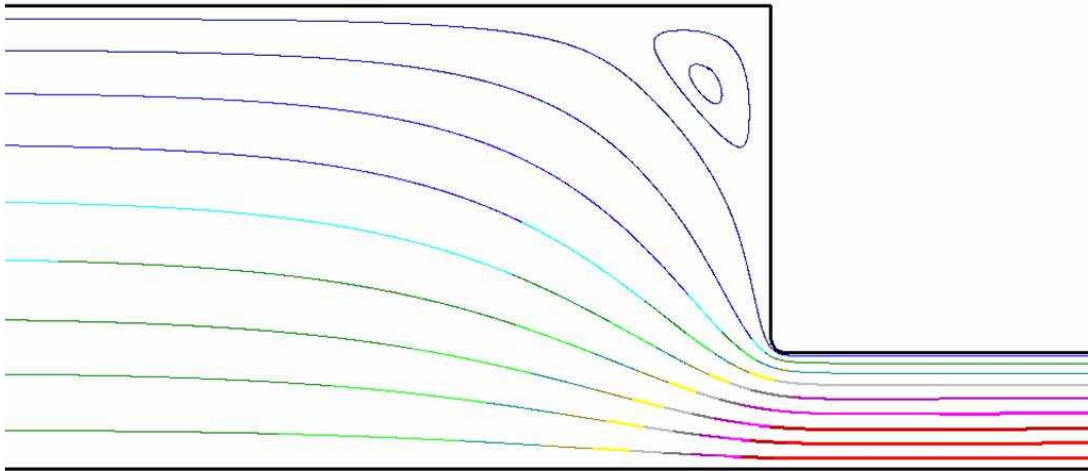


(c) Total A_{xx} stress component

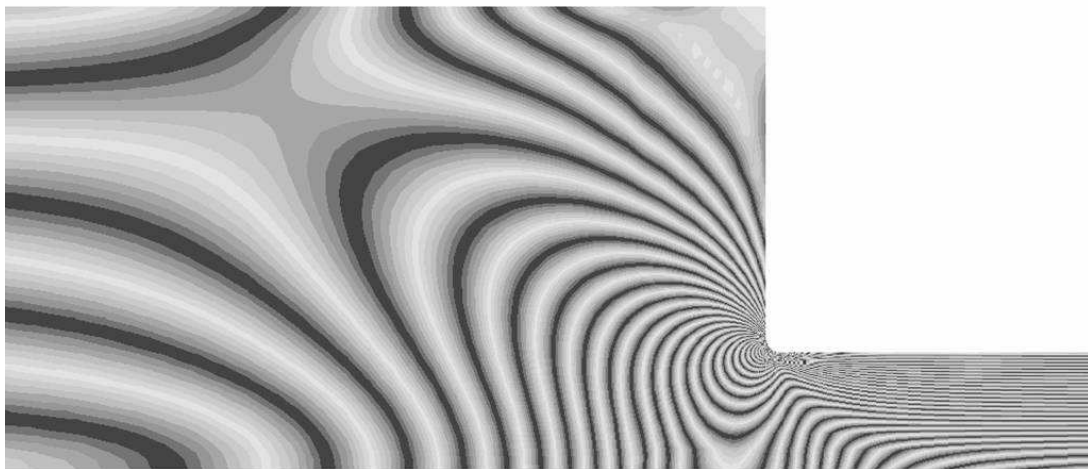
Figure 2: Flow around a cylinder



(a) Mesh

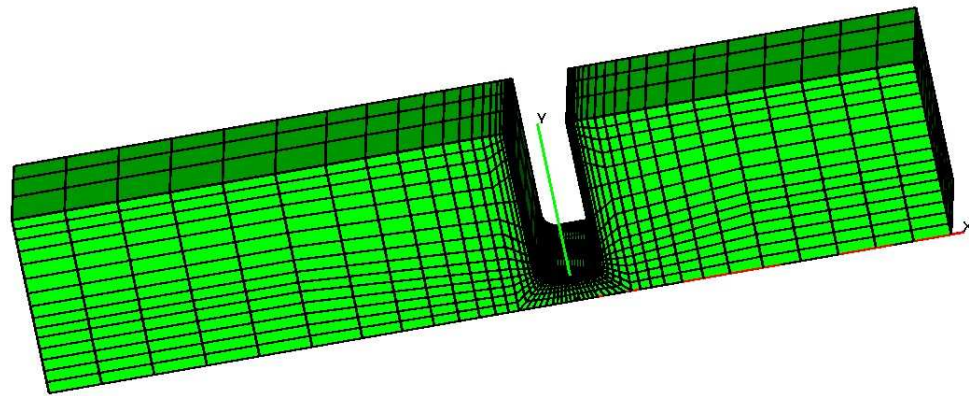


(b) Velocity streamlines

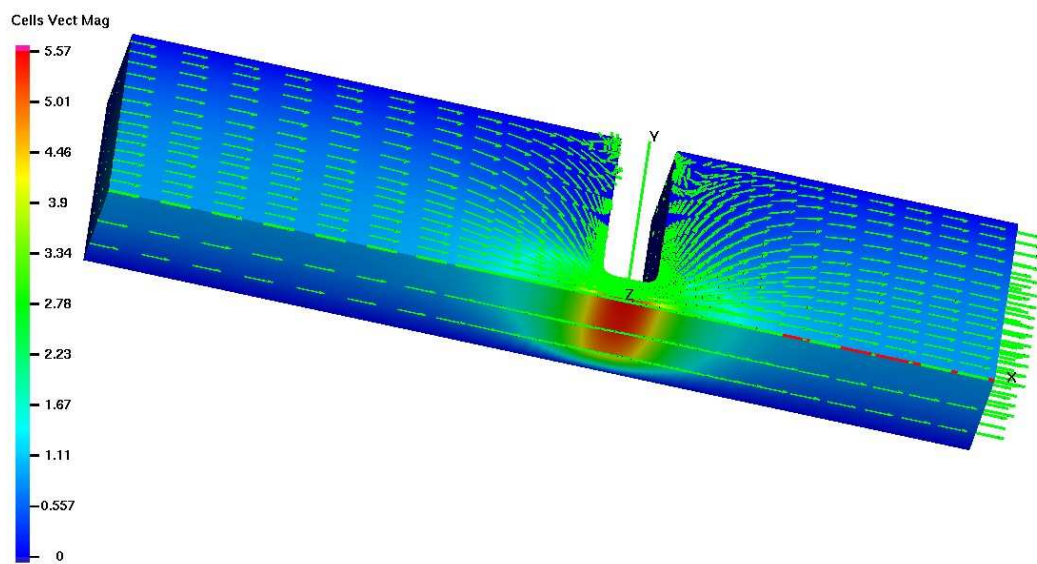


(c) Stress birefringence pattern

Figure 3: Contraction flow

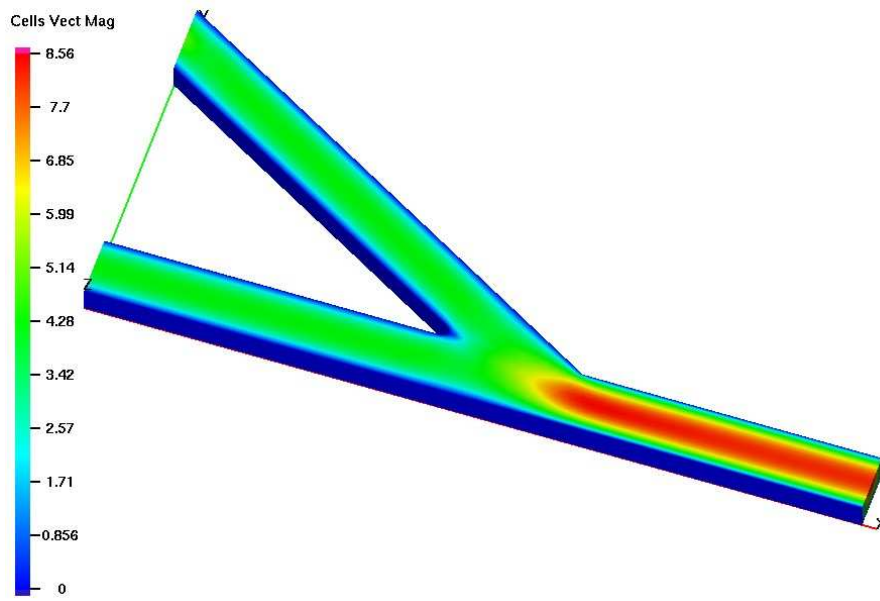


(a) Mesh



(b) Velocity vectors

Figure 4: Contraction-expansion flow



(a) Contours of velocity magnitude

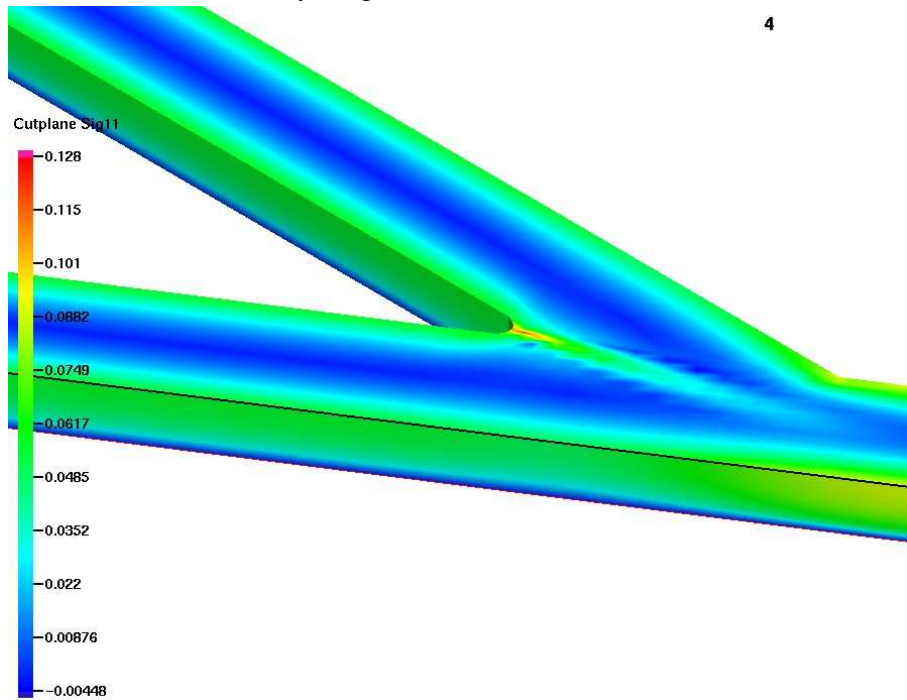
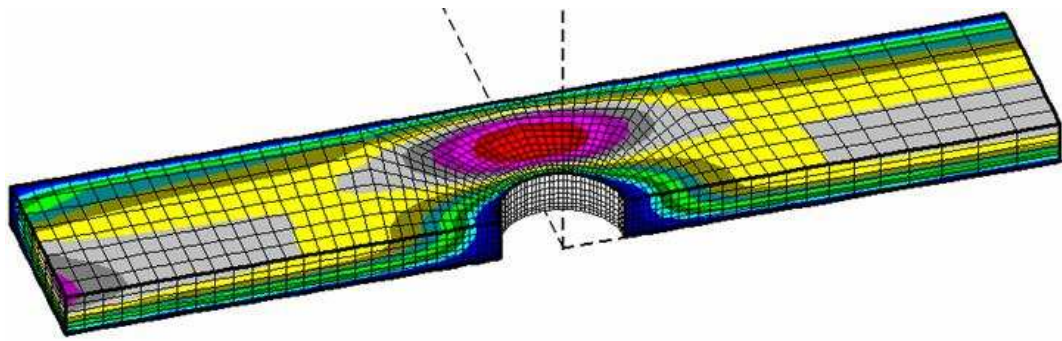
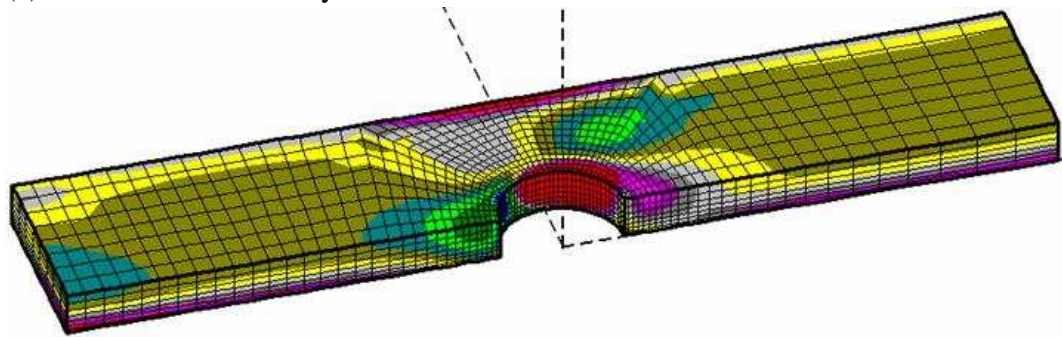
(b) Contours of total stress σ_{xx}

Figure 5: Y-shaped channel

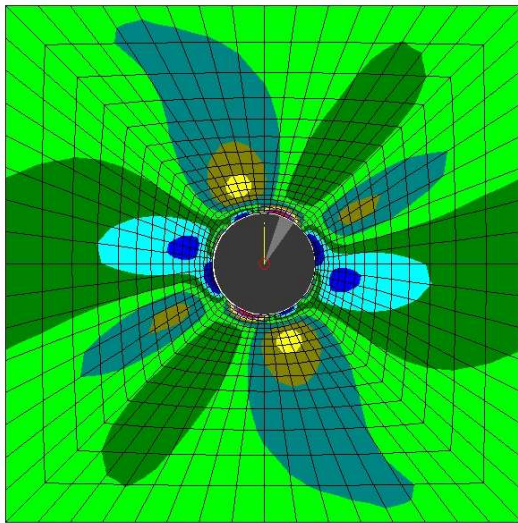


(a) Contours of u-velocity

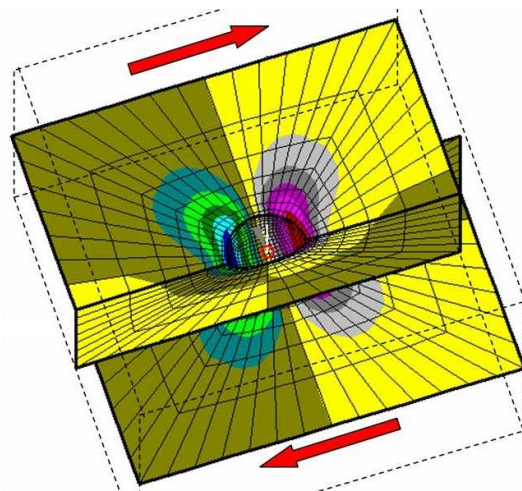


(b) Total A_{xx} stress component

Figure 6: Channel flow past a cylinder

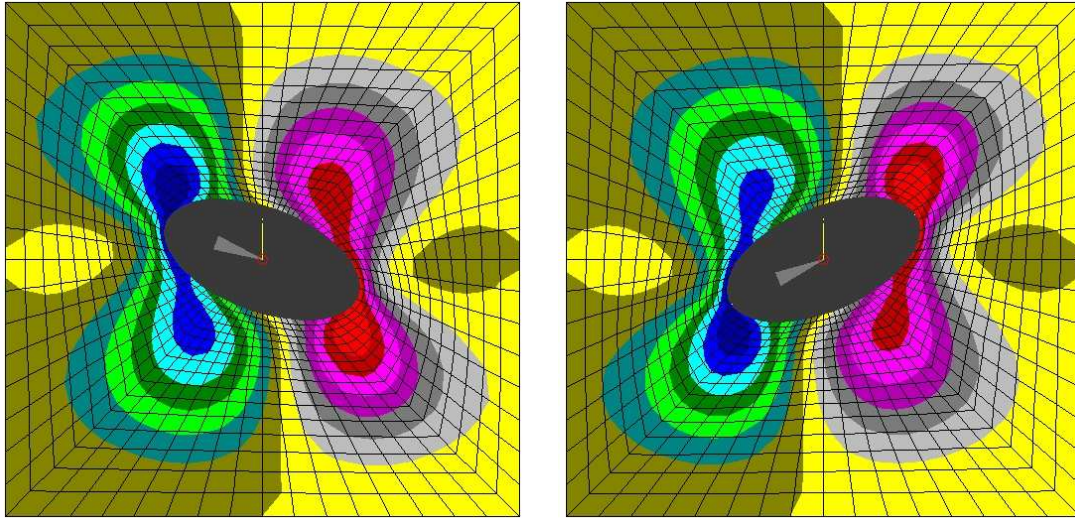


(a) 2d circular particle



(b) 3d spherical particle

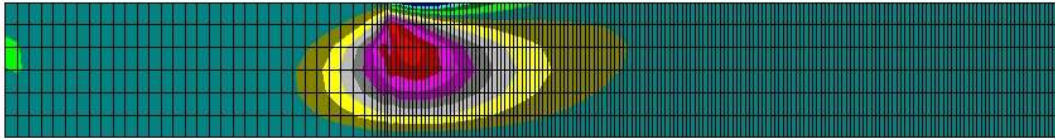
Figure 7: Shear flow around a rigid particle



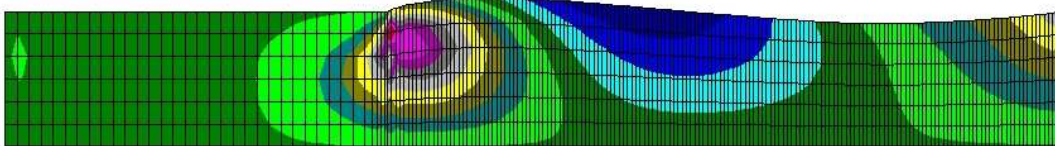
(a) Time step 40

(b) Time step 400

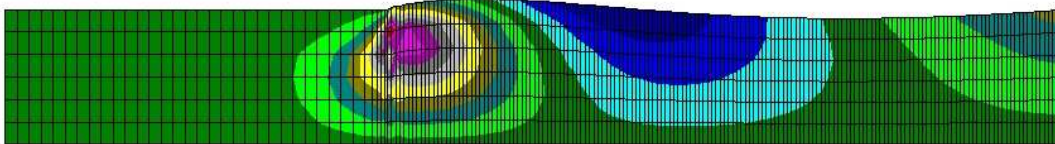
Figure 8: ALE grid for shear flow around an elliptical particle



(a) Time step 20, max $u = 0.6$



(b) Time step 600, max $u = 1.16$



(c) Time step 1500, max $u = 1.12$

Figure 9: 2d die swell problem, u-velocity contours