Automated modelling of viscoelastic flow using FEniCS

Luisa Molari
DISTART, University of Bologna, Italy
E-mail: luisa.molari@mail.ing.unibo.it

Garth N. Wells
Faculty of Civil Engineering and Geosciences, Delft University of Technology
E-mail:g.n.wells@tudelft.nl

Keywords: viscoelastic flow, finite element method, metaprogramming.

SOMMARIO Con un alto livello di astrazione, è possibile automatizzare in modo efficiente lo sviluppo dei modelli ad elementi finiti, con vantaggi in termini di rapidità di sviluppo, di riduzione degli errori di programmazione e con la possibilità di ottimizzare il codice. L'importanza di tutto questo è illustrata, usando gli strumenti sviluppati nel progetto FEniCS [1], con lo studio di un fluido viscoelastico in un contesto euleriano. In particolare si è simulata la caduta di una sfera in un tubo cilindrico con fluido viscoelastico.

ABSTRACT Using high-level abstractions, it is possible to efficiently automate the development of finite element models. This has advantages in terms of rapid development, a dramatic reduction in programming errors and offers the possibility of performing special optimisations to produced highly efficient code. The power of this concept is illustrated using tools from the FEniCS project [1] for the simulation of a viscoelastic fluid in an Eulerian framework, and the benchmark problem of a sphere falling in a cylinder pipe is simulated.

1. INTRODUCTION

The FEniCS project [1] provides tools for the automation of computational mathematical modelling. It aims to facilitate the translation of mathematical abstractions into optimised computer code. The link between governing equations and computer implemention is shortened, maintaining an emphasis on the underlying mathematical representation and reduces exposure to code complexities, which reduces development time and programming errors.

Several tools from the FEniCS project are utilised here for solving a viscoelastic flow problem. FIAT [2] provides automatic generation of finite element bases and integration schemes, FFC [3], the FEniCS Form Compiler, is a variational form compiler which interprets conventional or mixed variational forms and produces optimised code for element matrices and vectors, and DOLFIN [4] provides automatic assembly and solution of the ensuing equations. FFC in particular is an example of the application of metaprogramming - programs which write programs - for the finite element method. This approach provides scope for performance optimisations due to the automated translation of the mathematical model to machine code in several steps. The effectiveness of this approach is illustrated through the simulation of a viscoelastic flow in an Eulerian framework.

2. GOVERNING EQUATIONS

Under the hypotheses of incompressible, isothermal flow the equations for momentum and mass conservation are:

$$-\nabla p + \nabla \cdot \mathbf{T} + \mathbf{f} = \mathbf{0}, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where **u** is the velocity, p the pressure and **T** is the stress tensor. The stress tensor is given by:

$$\mathbf{T} \equiv 2\eta_e \mathbf{D} + \tau,$$

where η_e is the effective viscosity and $\mathbf{D} \equiv \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T)$ is the strain rate. The extra stress tensor τ is specified through the constitutive relation:

$$\lambda \check{\tau} + \tau - 2\eta \mathbf{D} = 0, \tag{3}$$

where λ is the characteristic relaxation time and η is the viscosity. The stress rate can be specified through the Upper-Convected Model (UCM):

$$\check{\tau} \equiv \frac{\partial \tau}{\partial t} + \mathbf{u} \cdot \nabla \tau - (\nabla \mathbf{u})^T \cdot \tau - \tau \cdot \nabla \mathbf{u}. \tag{4}$$

The variables involved are: stress, pressure and velocity. The problem is non linear and time-dependent. For weighting functions **S**, **v** and q, the variational problem can be stated as: find τ, **u** and p such that

$$(\mathbf{S}, \lambda \check{\tau} + \tau - 2\eta \mathbf{D}) = 0 \quad \forall\, \mathbf{S}, \tag{5}$$

$$(\mathbf{v}, -\nabla(p) + \nabla \cdot \mathbf{T} + \mathbf{f}) = 0 \quad \forall\, \mathbf{v}, \tag{6}$$

$$(q, \nabla \cdot \mathbf{u}) = 0 \quad \forall\, q. \tag{7}$$

The Crank-Nicolson method and a Newton-Raphson approach are used to solve the problem.

3. APPLICATION

The variational problem provides input for the compiler FFC, which is shown in Table 1 for the viscoelastic problem. The format of the input closely reassembles the mathematical notation. From this input, FFC generates optimised code (in C++) for the finite element assembler DOLFIN, which assembles and solves the required linear systems. The viscoelastic problem represents an application which benefits significantly from automation due to the coding complexities involved in a mixed three-field formulation. Using FFC, the implementation of mixed formulations with any number of fields and arbitrary bases is straightforward.

The benchmark of a sphere falling in a cylinder (Fig. 1) has been studied [5, 6]. For low Deborah number ($De = \lambda V/\eta$), the steady state is achieved without stabilisation terms. In Fig. 1, the pressure and the velocity in both the directions, for $De = 0.1$, are reported.

References

[1] `www.fenics.org`.

[2] Kirby RC, Algorithm 839: FIAT, A New Paradigm for Computing Finite Element Basis Functions, ACM Transactions on Mathematical Software, 2004, 33: 502-516.

[3] Logg A, FFC Manual, 2006, (`www.fenics.org`).

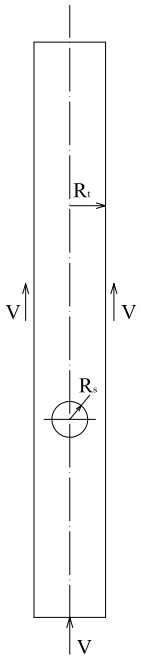
[4] Hoffman J, Jansson J, Logg A, Wells GN, DOLFIN Manual, 2006, (`www.fenics.org`).

[5] Baaijens FPT, Mixed finite element methods for viscoelastic flow analysis:a review, Journal of Non-Newtonian Fluid Mechanics, 1998, 79:192-199.

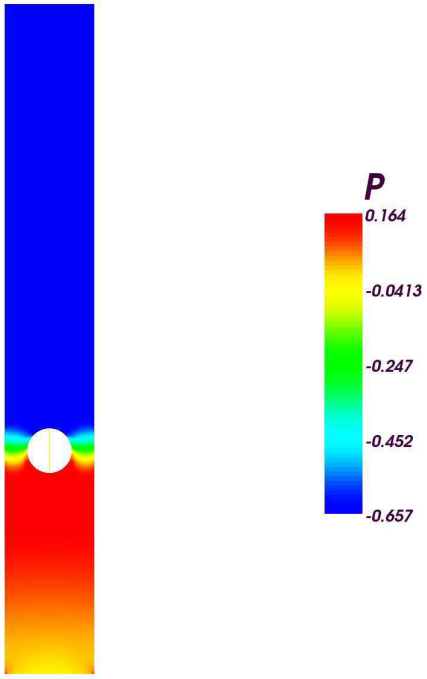
[6] Lunsmann WJ, Genieser L, Armstrong RC, Brown RA, Finite element analysis of steady viscoelastic flow around a sphere in a tube: calculations with constant viscosity models, Journal of Non Newtonian Fluid Mechanics, 1993, 48:63-99.

```
P = FiniteElement("Lagrange", "triangle", 1, 1)
T = FiniteElement("Vector Lagrange", "triangle", 1, 3)
U = FiniteElement("Vector Lagrange", "triangle", 2, 2)

element = U + T + P

( v, vs, vp) = TestFunctions(element)
( u, s,  p)  = TrialFunctions(element)
(fu, fs, fp) = Functions(TH)
(uc, sc, pc) = Functions(TH)
(du, ds, dp) = Functions(TH)

lam   = Constant() #characteristic relaxation time
eta   = Constant() #viscosity
etaE  = Constant()
dt    = Constant()
theta = Constant()

# Strain rate
def D(q):
return 0.5*(grad(q) + transp(grad(q)))

sthatlin = - mult(transp(grad(uc)),ms) - mult(transp(grad(u)),msc) \
           - mult(ms,grad(uc)) - mult(msc,grad(u))

#Upper convected model
sthat = s3 - mult(transp(grad(uc)),msc) - mult(msc,grad(uc))

# Bilinear forms
a1 = lam*dot(ms,mvs)*dx + theta*dt*lam*dot(s1,mvs)*dx  + theta*dt*lam*dot(s2,mvs)*dx
    + theta*dt*lam*dot(sthatlin,mvs)*dx + theta*dt*dot(ms, mvs)*dx - theta*dt*2*eta*dot(D(u),mvs)*dx
a2 =  - p*div(v)*dx + 2*etaE*dot(D(u),grad(v))*dx + dot(ms,grad(v))*dx
a3 =  div(u)*vp*dx
a =   a1 + a2 + a3

# Linear forms
L1 = theta*dt*lam*dot(sthat,mvs)*dx + theta*dt*dot(msc, mvs)*dx \
     - theta*dt*2*eta*dot(D(uc),mvs)*dx +theta*dt*dot(mds,mvs)*dx
L2 = dot(fu, v)*dx + pc*div(v)*dx - 2*etaE*dot(D(uc),grad(v))*dx + dot(msc,grad(v))*dx
L3 = div(uc)*vp*dx
L  = L2 - L1 - L3
```

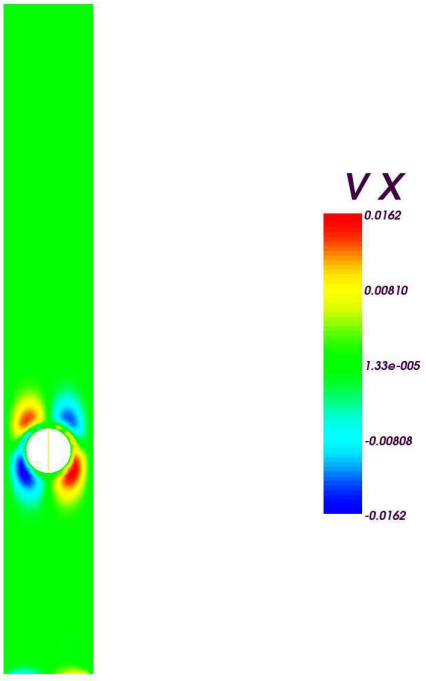
Tabella 1: FFC input code for the unsteady viscoelastic model.



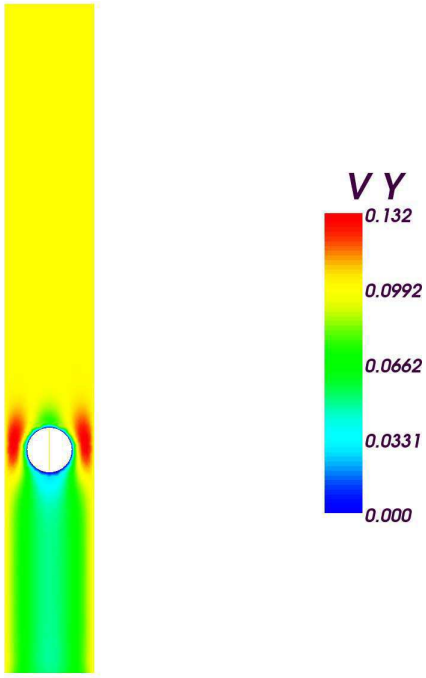
(a) Geometry $R_t/R_s = 2$.



(b) Pressure.



(c) Velocity in y direction.



(d) Velocity in x direction.

Figura 1: Flow in a cylinder past a sphere.