Segregated Solver Memo

Weston Ortiz

March 2, 2022

Contents

1 Introduction

This document attempts to explain our implementation of decoupled (or segregated) methods in Goma. We implement two main methods of decoupling, the Characteristic Based Split (CBS) and a general decoupled method. Timing and verification results are shown for each method. Overall the CBS method proved to have easy matrix systems to solve but we were not able to achieve results for flow through problems. For the more general segregated methods we have promising results when no mesh motion is involved. In addition to describing our method and results this document also includes a tutorial for the general segregated method in Appendix B.

2 Implementation of Decoupled equations

In order to implement the segregated solver in Goma we needed to provide a way to decouple equations which up until now have been implemented as Fully Coupled when enabled. To allow for decoupled equations we needed to modify Goma in order to store multiple matrix systems for each set of decoupled equations and allow for information to be passed between matrix systems. Our implementation allows us to use field variables in Goma to transfer information between decoupled systems.

2.1 General Decoupling in Goma

To implement a more general segregated method in Goma we allowed any equation to be decoupled. To do this we created a system similar to our CBS split in which we solve decoupled matrices in a series of steps (within a time step). Values from previous steps are passed to the next through field variables which are used in assembling the Residual. However for it to be decoupled we then ignore all cross terms in the Jacobian from other steps.

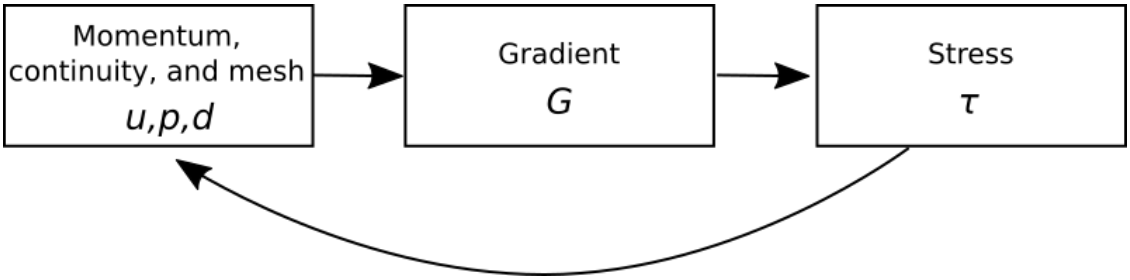

Figure 1: Decoupled systems illustration

Figure 1 shows an example of how a typical viscoelastic problem might be split. Our first matrix system consists of solving for momentum, continuity, and mesh using either the initial guess for gradient and stress terms at the first time step or using the previous time step values if we are not at the first time step. Then we solve for the gradient using the results from the previous step (and stress from past steps if required). Finally we solve for stress using the newly calculated gradient, momentum, continuity, and mesh from the previous two steps.

2.1.1 Steady state

We attempted to solve problems in a steady state fashion without time stepping by continuously iterating over steps until the L^2-norm of the change in the solution vector was small. However

this only seemed to work for simpler problems and in general the problem diverged as iteration continued.

2.1.2 March to steady state and transient problems

Instead of the finicky steady state iteration we employ a march to steady state for segregated steady state problems. Since our decoupled method does not work well without using time stepping we approach steady state problems as transient problems when we decouple them. The problem is setup the same as a transient problem and the only difference here is that we add a check to stop solving when we reach a steady state. We use the same criteria as mentioned above: we take the L^2-norm of the change in all solution vectors and when that is small we consider our problem to have reached a steady state.

3 Test problems

3.1 General segregated implementation

3.1.1 MilliFluidic Proppant Suspension

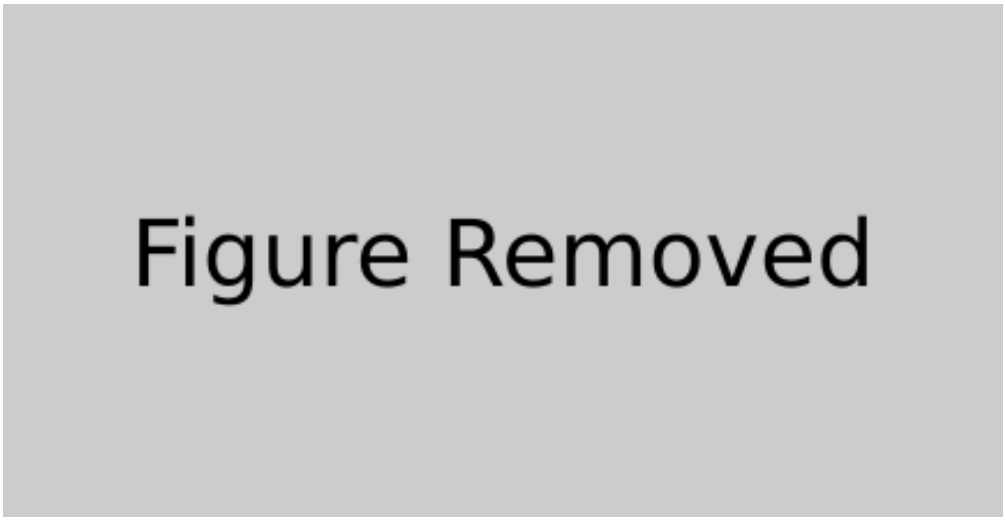

Figure 2: Millifluidic proppant experimental results.

Our large test problem for our general segregated method was a millifluidic proppant suspension which was modeled after experimental results. The MilliFluidic experiment was done to understand branching flow for proppant transport. This multiphysics problem includes species transport and viscoelastic flow (PTT).

In Goma, we decoupled as follows:

$$\begin{aligned} \rho[\frac{\partial u}{\partial t} + u \cdot \nabla u] - \nabla \cdot \{\mu_1(\epsilon(u) - G - G^T) + \sigma\} + \nabla p = 0 \\ \nabla \cdot u = 0 \end{aligned} \tag{1}$$

$$\frac{\partial \phi}{\partial t} + u \cdot \nabla \phi + \nabla \cdot J = 0 \tag{2}$$

$$G - \nabla u = 0 \tag{3}$$

$$\begin{aligned} \lambda[\frac{\partial \sigma_1}{\partial t} - u \cdot \nabla \sigma_1 - (1 - \frac{\chi}{2})(G^T \cdot \sigma_1 + \sigma_1 \cdot G) \\ + \frac{\chi}{2}(\sigma_1 \cdot G^T + G \cdot \sigma_1)] + \sigma_1 e^{\epsilon \lambda tr(\sigma_1)/\mu_1} - \mu_1(G + G^T) = 0 \end{aligned} \tag{4}$$

Where each set of equations (1, 2, 3, 4) represents a decoupled matrix. For this problem we used a 3D mesh with $47,416$ Hex27 elements and $430,335$ nodes (Figure 3). Momentum and species equations were run with Q2 interpolation while stress, continuity, gradient, and shear rate were run with Q1 interpolation.

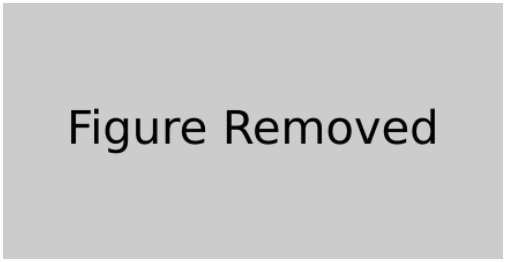

Figure 3: Millifluidic proppant mesh with $47,416$ Hex27 elements and $430,335$ nodes.

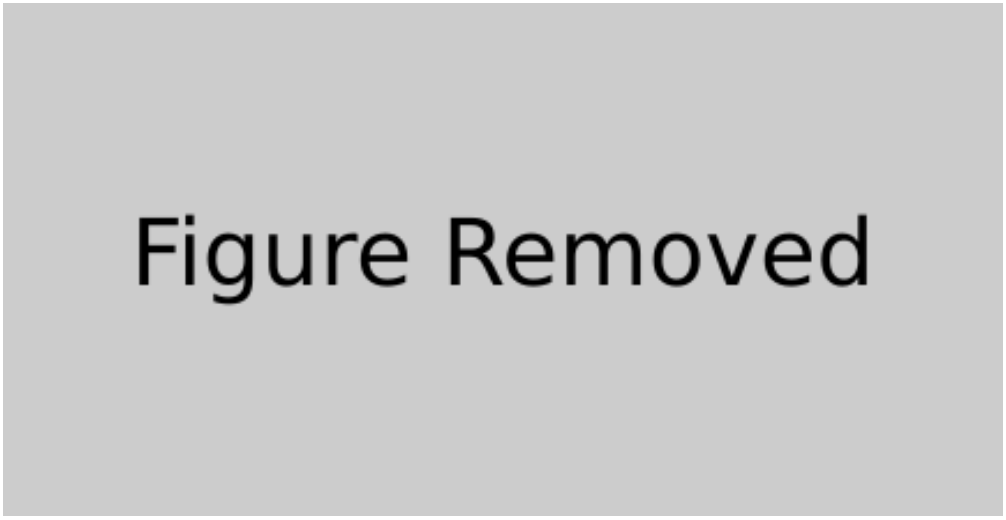

Figure 4: Millifluidic proppant simulation results showing the concentration of species in branches at times near 0, 45 and 90 seconds.

Timings comparing newton iterations can be seen in Table 1. From these timings we can see that per newton iteration the segregated solver has a 5 times speedup (140 seconds for segregated vs 700 seconds for fully coupled). However in actual run time terms our decoupled run was able to reach 900 time steps in the same time the fully coupled only reached 100 time steps completed. The results from the fully coupled and segregated runs matched well and as can be seen in Figure 4 performed similarly to the experiment.

Table 1: Timing results from the millifluidic proppant experiment. Timings are representative from newton iterations after the problem had begun to develop. Matrix timings are for individual decoupled matrices (equations 1, 2, 3, and 4)

	Assembly (sec)	Solve (sec)	Total (sec)
Matrix 1	32	70	102
Matrix 2	7	1	8
Matrix 3	16	2	18
Matrix 4	11	2	13
Decoupled Total	66	75	141
Fully Coupled	300	400	700

3.1.2 Drop in a constriction

Figure 5: Diagram of drop in constriction with labeled boundary conditions

Our final test problem involved introducing mesh equations into the segregated process. We chose an axisymmetric 2D viscoelastic problem which is diagrammed in Figure 5. This problem consists of a Newtonian drop in PTT suspending fluid. It is a moving mesh problem with a capillary free-surface around the drop that as the drop progresses requires remeshing to progress. The mesh consists of $36,871$ nodes and as reported from Goma we have $213,205$ unknowns.

To decouple this problem we separated systems in a similar way to the millifluidic problem. We tried two separate methods of decoupling. Method 1 can be seen in Figure 6, in this method we decouple mesh into its own matrix system. Method 2 can be seen in Figure 7, in this method mesh is coupled with momentum and continuity.

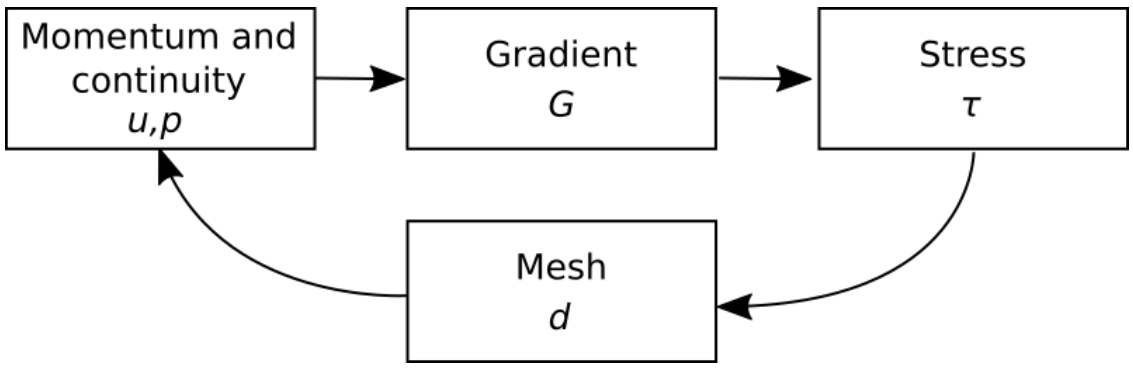

Figure 6: Method 1 decoupling mesh from momentum and continuity. Each box represents a separate matrix system

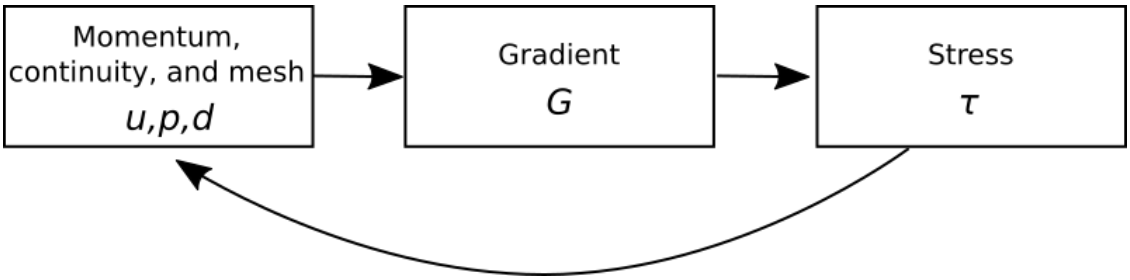

Figure 7: Method 2 coupling mesh with momentum and continuity. Each box represents a decoupled matrix system.

All versions of the problem start from the same starting state (Figure 8). Method 1 was able to reach the beginning of the constriction but was not able to pass through the constriction (Figure 9) due to failure to converge. Method 2 was able to make it partially through the constriction before failure to converge (Figure 10). From the final states in figures 9 and 10 we can see that the mesh coupled with momentum and continuity was able to progress further without issue. From the runs of each method it should also be noted that when mesh was not coupled with momentum and continuity (Method 1) the amount of remeshes needed to progress in the problem increased greatly.

Figure 8: Start state of drop.

Figure 9: Final state of Method 1 before failure to converge.

Figure 10: Final state of Method 2 before failure to converge.

From table 2 we can see that for method 2 and fully coupled the total runtime for a problem is very similar and the segregated version with coupled mesh, momentum, and continuity does not provide any real benefit over running the problem as fully coupled. This is because in Method 2 the coupled matrix takes the majority of the time to solve as it is composed of Q2 momentum and Q2 mesh while other matrices are only composed of Q1 elements, so the majority of the time is spent in solving the coupled difficult to solve matrix.

Table 2: Runtime of decoupled versions vs fully coupled version of drop in a constriction.

Simulation time (s)	Method 1 (hours	Method 2 (hours)	Fully coupled (hours)
0-50		32.7	40.1
50-95		25.4	28.8
Full 0-95		58.1	68.9

4 Running segregated problems

4.0.1 General segregated support

We were not able to easily support segregation of all equations and boundary conditions in Goma as it requires developer intervention to enable support (See Appendix A, Developer Notes).

Equations that were explicitly supported in the general segregated method in goma:

- Momentum
- Continuity
- Gradient
- Stress
- Mesh (ALE only)

Boundary conditions that were explicitly supported:

- Kinematic
- Capillary
- Dirichlet
- Generalized Dirichlet

Other boundary conditions may work but need to be verified by ensuring that residual contributions are being used from other matrices.

To add support for more equations and boundary conditions see the Developer Notes (Appendix A).

4.1 Required modifications to input files

Under solver specifications a total number of matrices must be specified.

```
Total Number of Matrices         = 4
```

Then under each material you must specify the number of matrices for that material (for CBS only one material is allowed and 4 matrices must be used).

```
Number of Materials = 1

MAT = liquid    1
      Coordinate System = CARTESIAN
      Element Mapping                  = isoparametric
      Mesh Motion = ARBITRARY
      Number of bulk species = 0
Number of Matrices = 4
```

Then for each matrix under a material a 1-indexed number is specified and the equations/options for that matrix.

```
MATRIX = 1
      Number of EQ = 5
      EQ = momentum1  Q2      U1      Q2           1.    1.   1.   1.   0.   0.
      EQ = momentum2  Q2      U2      Q2           1.    1.   1.   1.   0.   0.
      EQ = continuity P1      P       P1                 1.             0.
      EQ = mesh1      Q2      D1      Q2      0.   0.    1.   1.   0.   0.
      EQ = mesh2      Q2      D2      Q2      0.   0.    1.   1.   0.   0.
```

1. Transient march to steady state.

   Transient march to steady state still uses `Time integration = steady` and the previously described `Steady State Tolerance`. Instead of using `Maximum steady state steps` marching uses the `Maximum number of time steps` card as a normal problem would. To declare that a problem should be marched to steady state rather than iterated a new card should be created after `Steady State Tolerance`:

   - `March to Steady State = yes`

   This will behave similar to a transient problem except that it will check for the steady state convergence at each step.

4.1.1 Solver specifications

As mentioned before the total number of matrices in the problem must be specified in solver specifications.

```
Total Number of Matrices         = 4
```

4.1.2 Boundary conditions

Boundary conditions for segregated solves are specified in the same manner as normal solves. Care should be taken to only use supported boundary conditions in Section 4.0.1 or verify that a boundary condition is properly modified for segregated support before use.

4.1.3 Handling materials and equations

1. Multiple materials

   Multiple materials are specified in input file in the normal way except that now there is a requirement for each material in the input file to specify the number of matrices which will be declared on that material. A full example of a multi-material problem can be found in Appendix C. For each material matrix indices correspond to the global system, so matrix 1 on material 1 is also matrix 1 on material 2.

   It is expected that if equations are found in multiple materials then the matrices for those equations will match across materials (e.g. if momentum and continuity are in matrix 2 for material 1 then if material 2 also is solving for momentum and continuity those equations must exist in matrix 2). This is not strictly enforced when reading the input so care must be taken here.

2. Time step control option

   `Disable time step control = yes` is used to avoid running predict solution on an irrelevant matrix (e.g. G):

```
 MATRIX = 2
       Disable time step control = yes
       Number of EQ = 5
       EQ = gradient11 Q1      G11     Q1                1.             1.
       EQ = gradient12 Q1      G12     Q1                1.             1.
       EQ = gradient21 Q1      G21     Q1                1.             1.
       EQ = gradient22 Q1      G22     Q1                1.             1.
       EQ = gradient33 Q1      G33     Q1                1.             1.
```

3. Per matrix solvers

   Per matrix solver support is very limited and must use stratimikos input files.

   Under solver specifications `Solution Algorithm = stratimikos` is specified.

   Then under each matrix equation declaration (similar to the time step control) a stratimikos file is specified:

```
MATRIX = 1
       Stratimikos File = teko.xml
       Number of EQ = 3
       EQ = momentum1  Q2      U1      Q2          1.    1.   1.   1.   0.   0.
       EQ = momentum2  Q2      U2      Q2          1.    1.   1.   1.   0.   0.
       EQ = continuity Q2_D    P       Q2_D              1.             0.


MATRIX = 2
       Disable time step control = yes
       Stratimikos File = mumps.xml
       Number of EQ = 5
       EQ = gradient11 Q1      G11     Q1                1.             1.
       EQ = gradient12 Q1      G12     Q1                1.             1.
       EQ = gradient21 Q1      G21     Q1                1.             1.
       EQ = gradient22 Q1      G22     Q1                1.             1.
       EQ = gradient33 Q1      G33     Q1                1.             1.
```

5 Current Limitations

The majority of testing has been done on 2D problems with a focus on Oldroyd-B viscoelastic problems.

Segregated works reasonably well so long as mesh and momentum and continuity are tied together. Less so when mesh is solved segregated.

Few boundary conditions and equations have been checked and modified for support in the segregated solver. So manual intervention is required to run many problems.

Bugs have been hard to find and there may be more global variables that contain information that should be separated by system but are not.

Segregated solves only support basic transient and steady state solves. Continuation and Augmenting conditions have not been supported. Many post processing functions have not been checked for support as well.

A Developer Notes

A new structure was added to store information about the global matrices

```
struct Problem_Graph
{
  int   imtrx;                  /* Current active matrix index */

  /* Temporarily make some things global */
  struct Matrix_Data *matrices;

  int time_step_control_disabled[MAX_NUM_MATRICES];
};
```

With this structure a new corresponding global was added, `pg`.

Where matrices contains data for each matrix system in the problem and can be accessed through the `pg` global:

```
struct Matrix_Data {
  struct Aztec_Linear_Solver_System *ams;
  double *x;                 /* Solution vector */
  double *x_old;             /* Solution vector , previous last time step */
  double *x_older;           /* Solution vector , previous prev time step */
  double *xdot;              /* xdot of current solution                  */
  double *xdot_old;          /* xdot_old of current solution              */
  double *x_update;          /* last update vector */
  double *resid_vector;      /* Residual vector */
  double *scale;
};
```

Many structures were also changed to accommodate additional matrices. A full list is not available but a grep for `imtrx` should give you an idea of the scope.

The most important changes for updating equations and boundary conditions are:

`pd->v`, `pd->e`, `pd->etm`, `upd->vp`, `upd->ep`, and `ei` all now have an index for which matrix they correspond to. Many more arrays and structures were also changed to have an additional index corresponding to a matrix system.

In addition a two new arrays were added to check if an equation is present in any matrix system. `pd->gv` which is used in a similar manner to `pd->v` (but is a global check), and `pd->mi` which will return the corresponding matrix index for a given equation and -1 if it is not present in the problem.

`pg->imtrx` is a global set to the current global matrix begin solved.

To update boundary conditions and assemble functions, we update all relevant subroutines to use `pd->gv` for the residual contributions and keep the traditional `pd->v` access for Jacobian contributions. This ensures that our Jacobian contributions are only from other equations in the same matrix, while other equations outside that matrix system still contribute to the residual.

Special care should be taken here to make sure that any values used in the residual contributions that come from subroutines also used the proper `pd->gv` checks.

All field variables for active equations in all matrices, see `pd->gv`, are loaded at each Gauss point step.

B Segregated die swell tutorial

B.1 Introduction

This tutorial presents running a simple die swell problem with a march to steady state for both fully coupled and segregated solves.

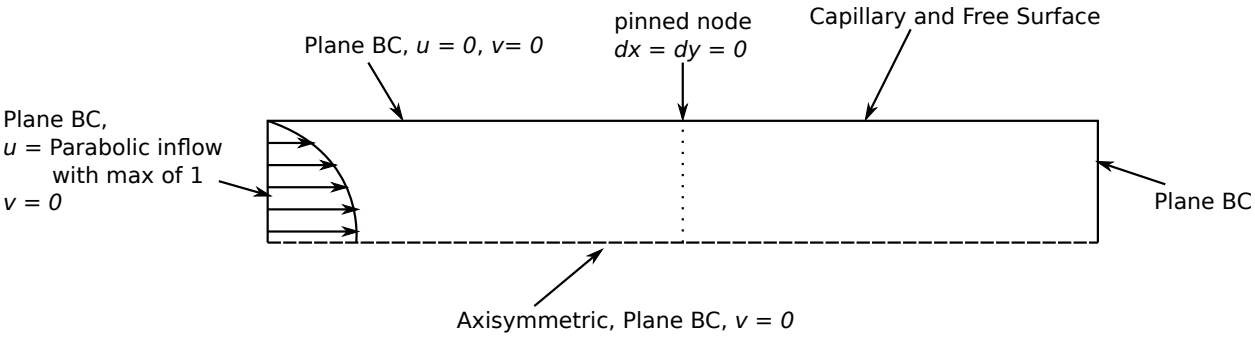

Figure 11: Diagram of simple die swell problem

B.2 Input file specifics

Some specific changes to the input file should be noted from running a regular transient goma problem.

B.2.1 Time integration specifications

Time integration is setup as a transient problem when marching to steady state so cards such as `delta_t` and `Maximum number of time steps` are required. For both segregated and fully coupled we add a steady state tolerance and a flag to enable marching to steady state. One notable behavior change for march to steady state is that all timesteps are saved to the exodus file so printing frequency is used. Listing 1 shows the tolerance and march to steady state specifications.

Listing 1: Flags to indicate a march to steady state, indicated in bold

```
------------------------------------------------------------
                Time Integration Specifications
------------------------------------------------------------
Time integration                = transient
delta_t                         = 1.0e-3
Maximum number of time steps    = 1000
Maximum time                    = 50.0e+0
Minimum time step               = 1.0e-6
Maximum time step               = 0.5
Time step parameter             = 0.
Time step error                 = -.001  0   1   1   1   1  1 1 1
Printing Frequency              = 1
Steady State Tolerance = 1e-6
March to Steady State = yes
```

B.3 Solver specifications

The only difference for solver specifications is that for the segregated solve we specify the total number of matrices in the problem (as seen in Listing 2).

Listing 2: Segregated solver specifications for die swell.

```
------------------------------------------------------------
                Solver Specifications
------------------------------------------------------------
Total Number of Matrices = 2
Solution Algorithm              = amesos
Amesos Solver Package           = superlu
Number of Newton Iterations     = 10
Newton correction factor        = 1
Normalized Residual Tolerance   = 1.0e-10
```

B.4 Boundary Conditions

Boundary conditions are setup normally from Figure 11. Note that there is no difference between segregated and fully-coupled boundary condition specifications.

Listing 3: Boundary condition specifications for die swell.

```
------------------------------------------------------------
       Boundary Condition Specifications
------------------------------------------------------------
Number of BC                    = -1

# inlet
BC = V NS 1 0.
BC = PLANE SS 1  1. 0. 0. 0.
BC = GD_LINEAR SS 1 R_MOMENTUM1 0 VELOCITY1 0 0. -1.
BC = GD_PARAB  SS 1 R_MOMENTUM1 0 MESH_POSITION2  0 1. 0. -1.

# Bottom plane
BC = PLANE SS 2  0. 1. 0. 0.
BC = V NS 2 0.

# Top, plane
BC = PLANE SS 5  0. 1. 0. -1
BC = U NS 5 0.
BC = V NS 5 0.

# Top, free surface
BC      = KINEMATIC SS 4 0.
BC      = CAPILLARY SS 4 1.0 0. 0.

# Pinned node
BC = DX NS 100 0.
BC = DY NS 100 0.

# right side
BC = PLANE SS 3  1. 0. 0. -7.

END OF BC
```

B.5 Material and equations specifications

This section contains the major change for segregated problems. We still specify equations under materials but for segregated problems we also have to specify what matrix system a set of equations belongs to.

Illustrated in the change from Listing 4 to Listing 5, 2 matrices are used to solve this problem. We create a matrix 1 for solving momentum and continuity in 5 where we specify the number of equations for that matrix then list equations as normal. Then we specify the matrix 2 with mesh equations. Another note is that for matrix 2 we disable time step control so that matrix 2 does not contribute to the size of our Δt.

Take note that we specify the number of matrices for each material as materials may not always contain all matrices. Listing 5 illustrates the differences with segregated specific changes marked in bold. This is in comparison with Listing 4 for a fully coupled problem.

Listing 4: Fully coupled material and equation specification

```
Number of Materials = 1

MAT = coating_liq    1
      Coordinate System = CYLINDRICAL
      Element Mapping  = isoparametric
      Mesh Motion = ARBITRARY
      Number of bulk species = 0

        Number of EQ   = 5
        EQ = momentum1  Q2  U1  Q2        0.   1.   1.   1.   0.   0.
        EQ = momentum2  Q2  U2  Q2        0.   1.   1.   1.   0.   0.
        EQ = continuity Q1  P   Q1   1.                       0.
        EQ = mesh1      Q2  D1  Q2        0.   0.   1.   1.   0.
        EQ = mesh2      Q2  D2  Q2        0.   0.   1.   1.   0.
```

Listing 5: Segregated material and equation specification, segregated specifics are marked in bold

```
Number of Materials = 1

MAT = coating_liq    1
      Coordinate System = CYLINDRICAL
      Element Mapping  = isoparametric
      Mesh Motion = ARBITRARY
      Number of bulk species = 0

Number of Matrices = 2

MATRIX = 1
        Number of EQ   = 3
        EQ = momentum1  Q2  U1  Q2        0.   1.   1.   1.   0.   0.
        EQ = momentum2  Q2  U2  Q2        0.   1.   1.   1.   0.   0.
        EQ = continuity Q1  P   Q1   1.                       0.
                                    div   ms  adv  bnd  dif  src  por

MATRIX = 2
        Disable time step control = yes
        Number of EQ   = 2
        EQ = mesh1      Q2  D1  Q2        0.   0.   1.   1.   0.
        EQ = mesh2      Q2  D2  Q2        0.   0.   1.   1.   0.
```

B.6 Running the die swell problems

Because we are using march to steady state, both fully coupled and segregated solves are expected to be ran from the segregated branch on github. This is because march to steady state has not been implemented on the main branch.

Problems are run in the normal way `goma -a -i inputfile` and the output will look familiar.

In the included files there is a `die_swell_fc.inp` input file for fully coupled as well as `die_swell_seg.inp` input file which contains the changes as described to change the problem into a segregated problem.

B.7 Notable changes in goma output

When running a march to steady state, when there are two time steps available to compare (current and previous) a new output indicating the change in the solution vector appears:

```
Delta x 0.00542071
```

The number indicates L^2 norm of the difference between the current solution vector and the previous steps solution vector. This is what is used to check if we have reached our indicated `Steady State Tolerance`.

The major change in output comes from a segregated solve. For each time step we now have multiple newton iterations (depending on the number of matrices). So for our die swell problem we can see in Listing 6 that there is now an indication for which matrix is being solved (this is 0 indexed differing from the input file which is 1 indexed).

Listing 6: Single time step output for segregated solve.

```
=> Try for soln at t=0.002 with dt=0.001 [1 for 1] (BE)

====================  SOLVING MATRIX 0  ============================


   N e w t o n  C o n v e r g e n c e  - I m p l i c i t  T i m e  S t e p

  ToD    itn   L_oo    L_1     L_2     L_oo    L_1     L_2   lis  asm/slv (sec)
-------- --- ------- ------- ------- ------- ------- ------- --- ---------------
09:50:28 [0] 1.0e+00 1.1e+01 3.0e+00 1.2e+01 2.5e+03 1.1e+02  1  4.0e-01/6.0e-02
09:50:29 [1] 1.3e-04 2.0e-02 9.6e-04 2.2e-01 4.6e+01 2.4e+00  1  4.0e-01/7.0e-02
09:50:29 [2] 1.1e-07 1.5e-05 7.1e-07 8.6e-05 2.7e-02 1.2e-03  1  4.1e-01/6.0e-02
09:50:30 [3] 8.2e-14 1.3e-11 5.0e-13 6.1e-11 1.7e-08 5.4e-10  1  4.0e-01/6.0e-02
scaled solution norms   1.268719e+01  5.867433e-01  1.770073e+00

=> Try for soln at t=0.002 with dt=0.001 [1 for 1] (BE)

====================  SOLVING MATRIX 1  ============================


   N e w t o n  C o n v e r g e n c e  - I m p l i c i t  T i m e  S t e p

  ToD    itn   L_oo    L_1     L_2     L_oo    L_1     L_2   lis  asm/slv (sec)
-------- --- ------- ------- ------- ------- ------- ------- --- ---------------
09:50:30 [0] 2.1e-07 1.0e-06 3.3e-07 3.2e-07 2.1e-05 8.2e-07  1  3.6e-01/5.0e-02
09:50:31 [1] 1.3e-11 7.8e-11 2.0e-11 7.0e-11 2.1e-08 5.6e-10  1  3.5e-01/5.0e-02
scaled solution norms   1.309076e-04  8.193678e-06  1.612681e-05

CONSTANT DELTA_T          [, 2.3e-01, 7.8e-02]
Predictor-corrector norm: [,   1 v  ,   1 P  ]

Maximum Delta x 0.182505
```

B.8 Using separate solvers for each matrix

Now that we have covered changes needed to run a segregated die swell problem we can use another feature of segregated solves: separate solvers for each system. To accomplish this we need to compile segregated Goma with Stratimikos and Teko support (Teko for using block based preconditioners). This can be done by adding `HAVE_STRATIMIKOS` and `HAVE_TEKO` to the defines in the `settings.mk` file:

```
DEFINES =  -Dlinux \
           -DCOMPILER_64BIT \
           -DENABLE_AMESOS \
           -DTRILINOS \
           -DCHECK_FINITE\
           -DNO_CHEBYSHEV_PLEASE \
           -DMDE=27 \
           -DMAX_PROB_VAR=15 \
           -DMAX_EXTERNAL_FIELD=4 \
           -DMAX_CONC=4 \
           -DCOUPLED_FILL \
           -DPARALLEL \
           -DEIGEN_SERIAL \
           -DHAVE_STRATIMIKOS \
           -DHAVE_TEKO
```

B.8.1 Extra input files

To run with separate solvers we need to use Stratimikos input files which are specified in XML. Examples can be found under the trilinos Stratimikos and teko packages source and further parameters can be found from Stratimikos documentation at Trilinos documentation.

We first create a generic GMRES with ILUT and fill factor 3 Stratimikos xml file (`gmres.ilut.3.xml`):

```
<ParameterList>
  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="string" value="GMRES"/>
          <Parameter name="Convergence Test" type="string" value="r0"/>
          <Parameter name="Size of Krylov Subspace" type="int" value="300"/>
        </ParameterList>
        <Parameter name="Max Iterations" type="int" value="400"/>
        <Parameter name="Tolerance" type="double" value="1e-13"/>
      </ParameterList>
    </ParameterList>
  </ParameterList>
  <Parameter name="Preconditioner Type" type="string" value="Ifpack"/>
  <ParameterList name="Preconditioner Types">
    <ParameterList name="Ifpack">
      <Parameter name="Prec Type" type="string" value="ILU"/>
      <Parameter name="Overlap" type="int" value="1"/>
      <ParameterList name="Ifpack Settings">
        <Parameter name="fact: level-of-fill" type="int" value="3"/>
      </ParameterList>
    </ParameterList>
  </ParameterList>
</ParameterList>
```

Next we create a file for the SIMPLE preconditioner with GMRES as the solver (`simple.xml`):

```
<ParameterList name="Stratimikos">

  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <Parameter name="Preconditioner Type" type="string" value="Teko"/>

  <ParameterList name="Linear Solver Types">
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="string" value="GMRES"/>
          <Parameter name="Convergence Test" type="string" value="r0"/>
          <Parameter name="Size of Krylov Subspace" type="int" value="1000"/>
          <Parameter name="Output Frequency" type="int" value="10"/>
        </ParameterList>
        <Parameter name="Max Iterations" type="int" value="1000"/>
        <Parameter name="Tolerance" type="double" value="1e-5"/>
      </ParameterList>
    </ParameterList>
  </ParameterList>
  <ParameterList name="Preconditioner Types">

    <ParameterList name="Teko">
      <Parameter name="Strided Blocking" type="string" value="2 1"/>
      <Parameter name="Inverse Type" type="string" value="SIMPLE"/>

      <ParameterList name="Inverse Factory Library">

        <ParameterList name="SIMPLE">
          <Parameter name="Type" type="string" value="NS SIMPLE"/>
          <Parameter name="Inverse Velocity Type" type="string" value="ML-Velocity"/>
          <Parameter name="Inverse Pressure Type" type="string" value="ML-Pressure"/>
        </ParameterList>

        <ParameterList name="ML-Pressure">
          <Parameter name="Type" type="string" value="ML"/>
          <Parameter name="Base Method Defaults" type="string" value="SA"/>
          <ParameterList name="ML Settings">
            <Parameter name="PDE equations" type="int" value="1"/>
            <Parameter name="max levels" type="int" value="2"/>
            <Parameter name="cycle applications" type="int" value="1"/>
            <Parameter name="aggregation: type" type="string" value="Uncoupled"/>
            <Parameter name="smoother: type" type="string" value="Gauss-Seidel"/>
            <Parameter name="smoother: pre or post" type="string" value="both"/>
            <Parameter name="smoother: sweeps" type="int" value="2"/>
            <Parameter name="coarse: type" type="string" value="Amesos-KLU"/>
          </ParameterList>
        </ParameterList>

        <ParameterList name="ML-Velocity">
          <Parameter name="Type" type="string" value="ML"/>
          <Parameter name="Base Method Defaults" type="string" value="SA"/>
          <ParameterList name="ML Settings">
            <Parameter name="PDE equations" type="int" value="2"/>
            <Parameter name="max levels" type="int" value="2"/>
            <Parameter name="cycle applications" type="int" value="1"/>
            <Parameter name="aggregation: type" type="string" value="Uncoupled"/>
            <Parameter name="smoother: type" type="string" value="Gauss-Seidel"/>
```

```
           <Parameter name="smoother: pre or post" type="string" value="both"/>
           <Parameter name="smoother: sweeps" type="int" value="2"/>
           <Parameter name="coarse: type" type="string" value="Amesos-KLU"/>
         </ParameterList>
       </ParameterList>

     </ParameterList> <!-- End "Inverse Factory Library -->
   </ParameterList>
  </ParameterList>
</ParameterList>
```

B.8.2 Input file changes to use simple and multiple solvers

In order to use multiple solvers we must set the solution algorithm to Stratimikos as this is the only supported way to run multiple solvers. Because we are using Stratimikos the matrix storage format must also be set to epetra. Finally we must as PSPP pressure stabilization so that we can use the SIMPLE preconditioner.

```
------------------------------------------------------------
                Solver Specifications
------------------------------------------------------------
Total Number of Matrices = 2
Solution Algorithm = stratimikos
Matrix storage format = epetra
Number of Newton Iterations               = 10
Newton correction factor                  = 1
Normalized Residual Tolerance             = 1.0e-10
Pressure Stabilization = pspp
Pressure Stabilization Scaling = 1
```

Next under each matrix we indicate a Stratimikos file for that matrix to use, we will use the two previously created xml files: `simple.xml` and `gmres.ilut.3.xml`. For the momentum and continuity matrices we change this to both be equal order Q2 elements so that simple's block preconditioner matches and we use the simple XML file. For the mesh matrices we use the GMRES with ILUT XML file.

```
MAT = coating_liq    1

        Coordinate System = CYLINDRICAL

        Element Mapping  = isoparametric

        Mesh Motion = ARBITRARY

        Number of bulk species = 0

Number of Matrices = 2

MATRIX = 1
        Stratimikos File = simple.xml
        Number of EQ   = 3
        EQ = momentum1  Q2  U1  Q2        0.   1.   1.   1.   0.   0.
        EQ = momentum2  Q2  U2  Q2        0.   1.   1.   1.   0.   0.
        EQ = continuity Q2  P   Q2  1.                       0.
                                    div   ms  adv  bnd  dif  src  por
```

```
MATRIX = 2
  Disable time step control = yes
        Stratimikos File = gmres.ilut.3.xml
        Number of EQ   = 2
        EQ = mesh1      Q2  D1  Q2        0.   0.   1.   1.   0.
        EQ = mesh2      Q2  D2  Q2        0.   0.   1.   1.   0.
```

Now that Stratimikos input files have been added to the input file we can now run segregated goma with separate solvers. When running you might notice some extra output due to Teko compiling with debugging turned on by default (see Listing 7).

Listing 7: First time step for die swell with two solvers.

```
=> Try for soln at t=0.001 with dt=0.001 [0 for 0] (BE)

====================  SOLVING MATRIX 0  ============================


   N e w t o n  C o n v e r g e n c e  - I m p l i c i t  T i m e  S t e p

  ToD    itn   L_oo    L_1     L_2     L_oo    L_1     L_2   lis  asm/slv (sec)
-------- --- ------- ------- ------- ------- ------- ------- --- ---------------
08:25:29 [0] WARNING:  mm_fill_util.c:158: The beer_belly for multiple matrices might have issues.
1.0e+00 1.1e+01 3.0e+00   Teko: Inverse "SIMPLE" is of type strat prec = 0, strat solv = 0, block
    ↪ prec = 1
     Teko: Begin debug MSG
        Looked up "NS SIMPLE"
        Built Teuchos::RCP<Teko::PreconditionerFactory>{ptr=0x5c38f78,node=0x5be6180,strong_count
            ↪ =1,weak_count=0}
     Teko: End debug MSG
     Teko: Begin debug MSG
        SIMPLE Parameters:
           inv type    = ""
           inv v type  = "ML-Velocity"
           inv p type  = "ML-Pressure"
           alpha       = 1
           use mass    = 0
           vel scaling = Diagonal
        SIMPLE Parameter list:
        Inverse Velocity Type = ML-Velocity
        Inverse Pressure Type = ML-Pressure
     Teko: End debug MSG
  Teko: Inverse "ML-Velocity" is of type strat prec = 1, strat solv = 0, block prec = 0
  Teko: Inverse "ML-Pressure" is of type strat prec = 1, strat solv = 0, block prec = 0
1.2e+01 6.7e+03 2.2e+02  42 5.8e-01/2.8e-01
08:25:30 [1] 1.3e-04 2.0e-02 9.5e-04   Teko: Inverse "SIMPLE" is of type strat prec = 0, strat
    ↪ solv = 0, block prec = 1
     Teko: Begin debug MSG
        Looked up "NS SIMPLE"
        Built Teuchos::RCP<Teko::PreconditionerFactory>{ptr=0x5bdee68,node=0x5919a70,strong_count
            ↪ =1,weak_count=0}
     Teko: End debug MSG
     Teko: Begin debug MSG
        SIMPLE Parameters:
           inv type    = ""
           inv v type  = "ML-Velocity"
           inv p type  = "ML-Pressure"
           alpha       = 1
           use mass    = 0
```

```
          vel scaling = Diagonal
       SIMPLE Parameter list:
       Inverse Velocity Type = ML-Velocity
       Inverse Pressure Type = ML-Pressure
    Teko: End debug MSG
  Teko: Inverse "ML-Velocity" is of type strat prec = 1, strat solv = 0, block prec = 0
  Teko: Inverse "ML-Pressure" is of type strat prec = 1, strat solv = 0, block prec = 0
1.7e-01 1.5e+02 4.6e+00  51 4.9e-01/2.1e-01
08:25:31 [2] 9.5e-08 1.4e-05 6.5e-07   Teko: Inverse "SIMPLE" is of type strat prec = 0, strat
    ↪ solv = 0, block prec = 1
    Teko: Begin debug MSG
       Looked up "NS SIMPLE"
       Built Teuchos::RCP<Teko::PreconditionerFactory>{ptr=0x5bdcf08,node=0x5915cf0,strong_count
           ↪ =1,weak_count=0}
    Teko: End debug MSG
    Teko: Begin debug MSG
       SIMPLE Parameters:
          inv type    = ""
          inv v type  = "ML-Velocity"
          inv p type  = "ML-Pressure"
          alpha       = 1
          use mass    = 0
          vel scaling = Diagonal
       SIMPLE Parameter list:
       Inverse Velocity Type = ML-Velocity
       Inverse Pressure Type = ML-Pressure
    Teko: End debug MSG
  Teko: Inverse "ML-Velocity" is of type strat prec = 1, strat solv = 0, block prec = 0
  Teko: Inverse "ML-Pressure" is of type strat prec = 1, strat solv = 0, block prec = 0
7.4e-05 6.4e-02 1.9e-03  45 4.1e-01/1.5e-01
08:25:31 [3] 1.5e-12 2.2e-10 6.4e-12   Teko: Inverse "SIMPLE" is of type strat prec = 0, strat
    ↪ solv = 0, block prec = 1
    Teko: Begin debug MSG
       Looked up "NS SIMPLE"
       Built Teuchos::RCP<Teko::PreconditionerFactory>{ptr=0x5c26c58,node=0x57305e0,strong_count
           ↪ =1,weak_count=0}
    Teko: End debug MSG
    Teko: Begin debug MSG
       SIMPLE Parameters:
          inv type    = ""
          inv v type  = "ML-Velocity"
          inv p type  = "ML-Pressure"
          alpha       = 1
          use mass    = 0
          vel scaling = Diagonal
       SIMPLE Parameter list:
       Inverse Velocity Type = ML-Velocity
       Inverse Pressure Type = ML-Pressure
    Teko: End debug MSG
  Teko: Inverse "ML-Velocity" is of type strat prec = 1, strat solv = 0, block prec = 0
  Teko: Inverse "ML-Pressure" is of type strat prec = 1, strat solv = 0, block prec = 0
4.9e-10 1.4e-07 3.3e-09  48 4.0e-01/1.6e-01
scaled solution norms   1.231698e+01  1.141057e+00  2.851774e+00

=> Try for soln at t=0.001 with dt=0.001 [0 for 0] (BE)

==================== SOLVING MATRIX 1 ===========================
```

```
   N e w t o n  C o n v e r g e n c e  - I m p l i c i t  T i m e  S t e p

 ToD    itn   L_oo    L_1     L_2     L_oo    L_1     L_2   lis  asm/slv (sec)
-------- --- ------- ------- ------- ------- ------- ------- --- ---------------
08:25:32 [0] 6.0e-05 6.9e-04 1.6e-04 6.1e-05 1.5e-02 4.9e-04  16 4.0e-01/1.9e-01
08:25:32 [1] 9.6e-08 3.2e-07 1.1e-07 4.0e-07 5.7e-06 4.2e-07  15 3.9e-01/1.3e-01
08:25:33 [2] 1.1e-12 5.9e-12 1.5e-12 6.6e-12 1.4e-09 3.7e-11  17 3.4e-01/1.1e-01
scaled solution norms   6.069561e-05  4.000307e-06  7.931002e-06

CONSTANT DELTA_T          [, 2.0e-01, 1.3e-01]
Predictor-corrector norm: [,   1 v  ,   1 P  ]
```

C Full Segregated example input file

This is an example input for Goma for a drop in a constriction problem. It showcases how multiple materials are handled in the segregated input file.

```
#{include("params")}
---------------------------------------------------------
                FEM Problem Specifications
---------------------------------------------------------
FEM file                        = mesh.exoII
Output EXODUS II file           = out_rem00.exoII
GUESS file                      = contin.dat
SOLN file                       = soln.dat
Write intermediate results      = no
---------------------------------------------------------
                General Specifications
---------------------------------------------------------
Output Level                    = 0
Debug                           = 0
Initial Guess                   = read_exoII_file cont.exoII
---------------------------------------------------------
                Time Integration Specifications
---------------------------------------------------------
Time integration                = transient
delta_t                         = 0.05
Maximum number of time steps    = 100
Maximum time                    = {total_time}
Minimum time step               = 5e-06
Maximum time step               = {time_step_max}
Time step parameter             = 0.5
#Time step error                = err=.005  mesh=1  v=1  T=0   C=0  p=0
Time step error                 = -0.05  0  1 0 1 0 1
Initial Time                    = 0.0296875
Printing Frequency              = 1
Jacobian quality weight         = 0.4
Angle quality weight            = 0.2
Triangle quality weight         = 0.2
Element quality tolerance       = 0.2
Element quality tolerance type  = wtmin

---------------------------------------------------------
                Solver Specifications
---------------------------------------------------------

#Solution Algorithm              = umf
```

```
#Number of Newton Iterations      = 10
#Newton correction factor         = 1
#Normalized Residual Tolerance    = 1e-7
#Residual Ratio Tolerance         = 1e-2

Total Number of Matrices       = 3
Solution Algorithm               = amesos
Amesos Solver Package            = superlu
#Amesos Solver Package            = umf
UMF XDIM                         = 10330088
UMF IDIM                         = 10330088
Preconditioner                   = ls
Polynomial                       = 1
Size of Krylov subspace          = 64
Orthogonalization                = classical
Maximum Linear Solve Iterations  = 1000
Number of Newton Iterations      = 15
Newton correction factor         = 1
#Newton correction factor         = 1.0 1.0e-6 0.4 1.0e-1 0.1 1.0e+1
Normalized Residual Tolerance    = 1.0e-7
#Residual Ratio Tolerance         = 1.0e-2
------------------------------------------------------------
      Boundary Condition Specifications
------------------------------------------------------------

Number of BC = -1

#outlet
BC = PLANE  SS 44 1.0 0. 0. -88.

#top
#BC = PLANE SS 11 0. 1.0 0. -3.
BC = GD_LINEAR  SS 11  R_MESH_NORMAL  0  MESH_POSITION2  0  0.0  -1.0
BC = GD_TABLE   SS 11  R_MESH_NORMAL  0  MESH_POSITION1  0  1.0  LINEAR FILE=wall.dat

#inlet
BC = PLANE  SS 22 1.0 0.  0.  0.0

#bottom
BC = PLANE  SS 33 0.  1.0 0.  0.0

# Centerline symmetry condition
BC = V NS 3 0.
# No slip on the solid boundary
#BC = VELO_NORMAL SS 11 0.
#BC = VELO_SLIP SS 11 0.001 0. 0. 0.
BC = U NS 1 0.
BC = V NS 1 0.
# Outflow pressure
$$BC = FLOW_PRESSURE SS 44 0.
#BC = P NS 5 0.
# Inflow velocities and stresses
BC = GD_LINEAR SS 22 R_MOMENTUM1 0 VELOCITY1         0    0.       1.
BC = GD_PARAB  SS 22 R_MOMENTUM1 0 MESH_POSITION2    0    {-2/9} 0. {2/81}
BC = V NS 2 0.

# Outflow velocities
$$BC = GD_LINEAR SS 44 R_MOMENTUM1 0 VELOCITY1         0    0.       1.
$$BC = GD_PARAB  SS 44 R_MOMENTUM1 0 MESH_POSITION2    0    {-2/9} 0. {2/81}
```

```
  BC = V NS 4 0.

  # Inflow second normal stress
  BC = S22      NS     2    0

  # Inflow first normal stress
  BC = GD_LINEAR SS 22 R_STRESS11  0 POLYMER_STRESS11  0    0.       1.
  BC = GD_PARAB  SS 22 R_STRESS11  0 MESH_POSITION2    0    0.       0. {-2*we*(4/81)^2}

  BC = KINEMATIC SS 56 0.
  BC = CAPILLARY SS 56 15.38  0. 0.  #inverse of capillary number

  BC = CA NS 200  1.5707963 0.  1. 0.
  BC = CA NS 400  1.5707963 0.  1. 0.


  END OF BC

  ----
  Problem Description
  ---

  Number of Materials = 2

MAT = ptt  1

        Coordinate System = CYLINDRICAL

        Element Mapping = isoparametric

        Mesh Motion = ARBITRARY

        Number of bulk species = 0
        Number of viscoelastic modes = 1

Number of Matrices = 3


MATRIX = 1
        Number of EQ = 5
        EQ = momentum1  Q2      U1      Q2           1.    1.   1.   1.   0.   0.
        EQ = momentum2  Q2      U2      Q2           1.    1.   1.   1.   0.   0.
        EQ = continuity P1      P       P1                1.             0.
        EQ = mesh1      Q2      D1      Q2      0.    0.    1.   1.    0. 0.
        EQ = mesh2      Q2      D2      Q2      0.    0.    1.   1.    0. 0.


MATRIX = 2
        Disable time step control = yes
        Number of EQ = 5
        EQ = gradient11 Q1      G11     Q1                1.             1.
        EQ = gradient12 Q1      G12     Q1                1.             1.
        EQ = gradient21 Q1      G21     Q1                1.             1.
        EQ = gradient22 Q1      G22     Q1                1.             1.
        EQ = gradient33 Q1      G33     Q1                1.             1.

MATRIX = 3
        Number of EQ = 4
        EQ = stress11   Q1      S11     Q1           1.    1.   1.   1.   1.
```

```
       EQ = stress12   Q1      S12     Q1           1.    1.   1.   1.   1.
       EQ = stress22   Q1      S22     Q1           1.    1.   1.   1.   1.
       EQ = stress33   Q1      S33     Q1           1.    1.   1.   1.   1.



MAT = oil    2

  Coordinate System = CYLINDRICAL
  Element Mapping   = isoparametric
  Mesh Motion = ARBITRARY
  Number of bulk species = 1

Number of Matrices = 1

MATRIX = 1
       Number of EQ                     = 5
       EQ = momentum1  Q2      U1      Q2      1. 1.  1.  1.0   0.0   0.
       EQ = momentum2  Q2      U2      Q2      1. 1.  1.  1.0   0.0 0.
       EQ = continuity P1      P       P1  1.                          0.
       EQ = mesh1      Q2      D1      Q2      0.    0.    1.   1.    0. 0.
       EQ = mesh2      Q2      D2      Q2      0.    0.    1.   1.    0. 0.

Post Processing Specifications

Stream Function = yes
Streamwise normal stress = no
Pressure contours = yes
Second Invarient of Strain = no
Mesh Dilatation = no
Navier Stokes Residuals = no
Moving Mesh Residuals = no
Mass Diffusion Vectors = no
Mass Fluxlines = no
Energy Conduction Vectors = no
Energy Fluxlines = no
Time Derivatives = no
Mesh Stress Tensor = no
Mesh Strain Tensor = no
```