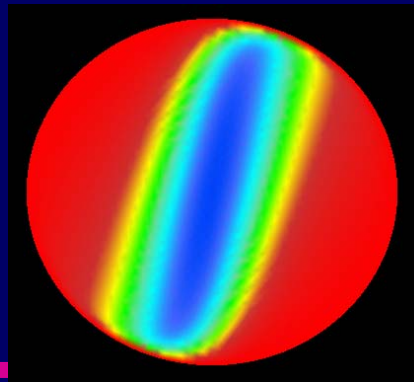


# Application of *Fastflo* to elasticity, thermoelasticity and elasto-viscoplasticity

CSIRO Mathematical and Information Sciences  
Clayton, Australia

*Fastflo*



Flexible finite element software for  
the numerical solution of PDEs

# Outline of presentation

- *Fastflo* – summary of features relevant to elasticity and associated topics
- 3 examples
  1. straightforward linear elasticity
  2. time-dependent thermoelasticity
  3. elasto - viscoplasticity

# Elasticity problems - relevant features of *Fastflo*

- can solve multiple PDEs
- flexible (in terms of geometry, equations, algorithms)
- (almost) any PDE can be solved
- self-contained (mesh generation, graphics)
- programming environment that empowers users
- able to specify and solve problems on boundaries
- very useful for rapid prototyping
- moving meshes and free surfaces are possible
- able to specify and solve problems in multiple regions

# Overview of *Fastflo*

- based on the finite element method, 2D and 3D
- range of element types (linear, quadratic; triangles, quadrilaterals, tetrahedra, hexahedra)
- internal mesh generator for 2D problems
- interface to commercial pre- and post-processors
- includes a high level macro command language to specify and solve PDEs
- graphical user interface

# Overview of *Fastflo* (cont'd)

- selection of sparse matrix solvers (direct and iterative)
- Tutorial Guide, on-line Reference Manual
- many well-documented applications
- incorporates feedback from dozens of licensees
- Fluids ToolBox released with *Fastflo V3*
- available in PC and UNIX versions, both written in C. The PC GUI is built using Borland C++ and makes use of Windows facilities. The UNIX GUI is built using Motif.

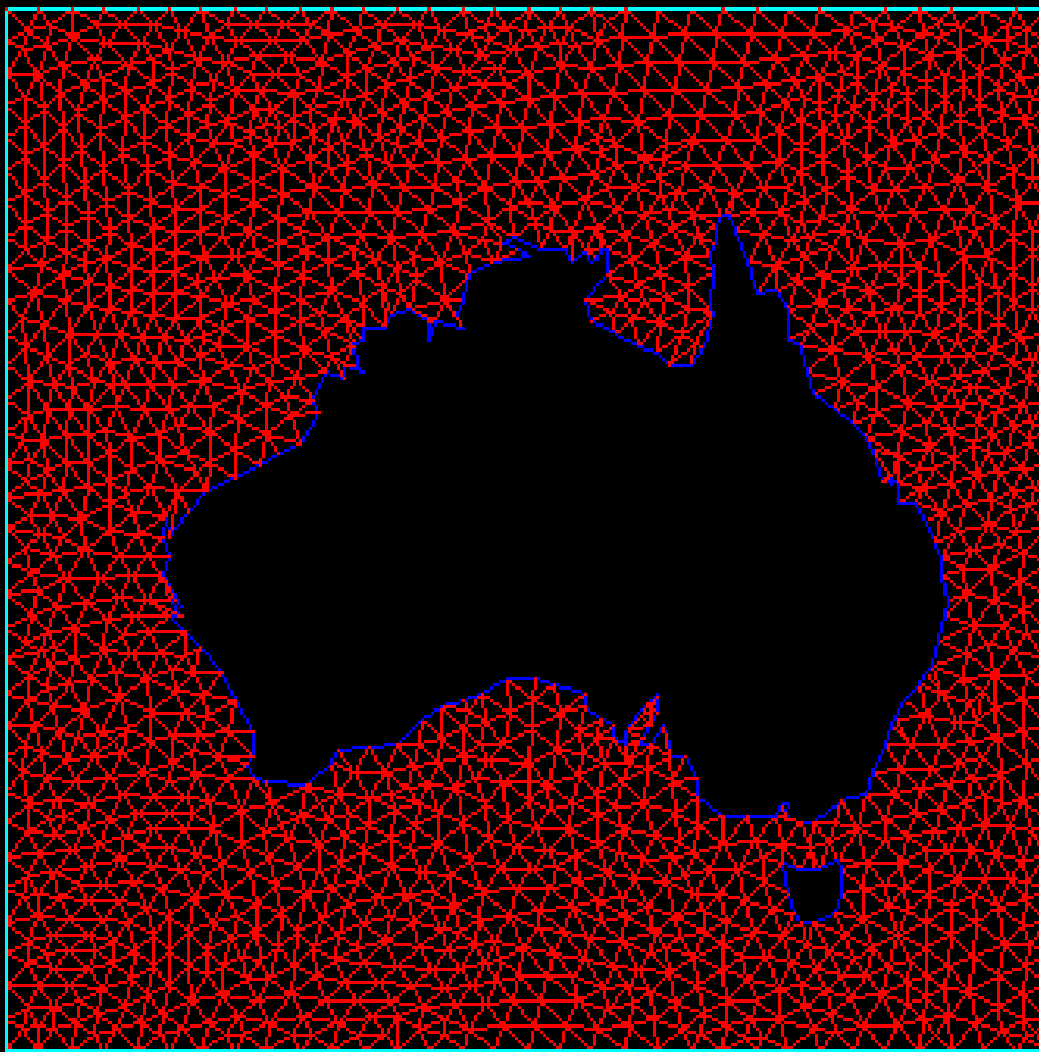
# Design features of *Fastflo*

- users present problems to *Fastflo* via two files:
  - \*.msh which contains geometrical information;
  - \*.prb which contains equations, boundary conditions, the algorithm, and commands to view the results
- data is stored on a vector stack (user-accessible)
- we think of *Fastflo* as a workbench, with tools to specify and solve PDEs; the workbench offers graphics, editing and printing facilities.

# Design features of *Fastflo* (cont'd)

- *Fastflo* macro code is open and portable; there is no need for time-consuming low level programming
- users are free
  - # to specify what equation(s) to solve
  - # to design the algorithm used for the solution
  - # to control the computations intelligently
- substantial guidance is available from an extensive list of examples and extensive documentation
- on-line Help file available for users

# Mesh generation



- \* triangular mesh generator
- \* linear and quadratic approx
- \* 2D: triangles, quadrilaterals
- \* 3D: tetrahedra, hexahedra
- \* can interface to third-party software (especially FEMAP)
- \* isoparametric elements
- \* deformable boundaries
- \* block mesh generator
- \* axisymmetry

# Basic equations for linear elasticity

$$\operatorname{div} \boldsymbol{\sigma} + \mathbf{F} = \mathbf{0}$$

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon} + \lambda I \operatorname{trace} \boldsymbol{\varepsilon}$$

$$\lambda = \frac{\nu E}{(1 + \nu)(1 - 2\nu)}$$

$$\mu = \frac{E}{2(1 + \nu)}$$

$\mathbf{F}$  = body force,  $I$  = identity matrix

$$\sigma = \text{stress tensor, } \varepsilon = \text{strain tensor} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

$\nu$  = Poisson ratio,  $E$  = Young's modulus

# Linear elasticity (continued)

The basic equation to be solved is for the displacements

$$\frac{\partial}{\partial x_j} \left( \mu \frac{\partial u_i}{\partial x_j} \right) + \frac{\partial}{\partial x_j} \left( \mu \frac{\partial u_j}{\partial x_i} \right) + \frac{\partial}{\partial x_i} \left( \lambda \frac{\partial u_j}{\partial x_j} \right) + F_i = 0$$

Boundary conditions are usually:

prescribed displacement

$$u_i = g_i$$

prescribed stress

$$\sigma_{ij} n_j = h_i \text{ (just enter value } h_i \text{)}$$

stress free

boundary condition omitted

Note the natural boundary condition applies to  $\sigma_{ij} n_j$ .

# 2D cases - plane strain, plane stress

Plane strain: all terms in  $\epsilon$  involving  $z$ -coefficients are zero  
(long prism, displacements only in cross-section);  
equations unchanged; 2D range of indices

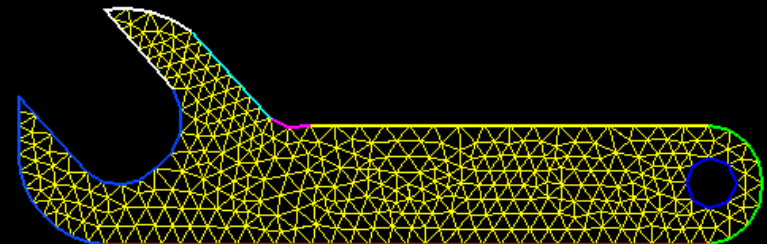
Plane stress: all terms in  $\sigma$  involving  $z$ -coefficients are zero  
(flat shapes); 2D range of indices; equations  
unchanged but

$$\lambda = \lambda_1 = \frac{\nu E}{(1 - \nu^2)}$$

# Example – deformation of a spanner

```
% stress_S.msh
400 0 4
0 0 0 0 0
List of vertices
30.0    0.0
198.0   0.0
213.0  17.0
198.0  33.0
198.0  24.0
206.0  17.0
198.0  10.0
192.0  17.0
87.0   33.0
76.0   35.0
54.0   59.0
30.0   65.0
49.0   43.0
44.0   21.0
25.0   20.0
6.0    41.0
6.0    25.0
13.0   8.0
```

```
List of boundary tags
1  1
2  2
3  2
4  0
5  3
6  3
7  3
8  3
5  0
4  4
9  5
10 6
11 7
12 8
13 9
14 9
15 10
16 11
17 12
18 12
End of boundary list
```



# Example – deformation of a spanner

```
% stress_S.prb
P  nu      0.3
P  E        1.0
P  mu      E/(2*(1+nu))
P  lambda  mu*nu*2/(1-nu)
D  8  fixed
D  9  fixed
D  10 fixed

A  displace
e  D_j{mu}D_j U1_i + D_j{mu}D_i U1_j + D_i{lambda} D_j U1_j
b  fixed    U1 = {0.,0.}
b  1  {0.0,mu}_i
b  2  curvature = 0.2071
b  3  curvature = -0.2071
b  5  curvature = -0.15
b  7  curvature = 0.1
b  9  curvature = -0.2071
b  12 curvature = 0.1

A  energy
e  V300 = [grad]{V100}
e  V302 = {(V302+V303)/2}
e  V303 = V302
e  V500 = V300 P V300
e  V701 = V301 + V304
e  V702 = V501 + V504
e  V001 = 2*mu*V702 + lambda*V701*V701

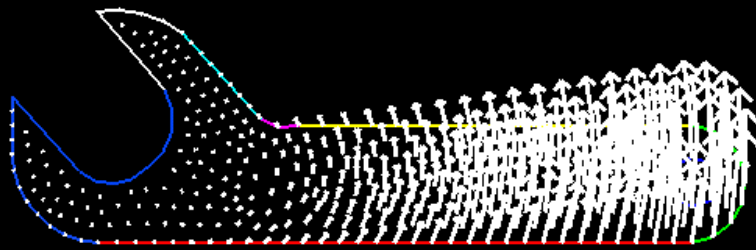
A  shear
e  V300 = [grad] {V100}
e  U1 = {sqrt((V301-V402)^2+(V302+V401)^2)}
```

```
< approx
    displace
    solve
    black
    arrow
>

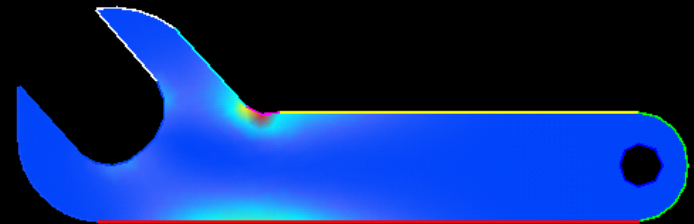
< showenergy
    energy
    black
    contour
    shade
    popp
>

< showshear
    shear
    solve
    black
    contour
    shade
    popp
>
```

# Example – deformation of a spanner



displacements



strain energy

682.5  
654.9  
620.5  
586  
551.5  
517.1  
482.6  
448.1  
413.6  
379.2  
344.7  
310.2  
275.8  
241.3  
206.8  
172.4  
137.9  
103.4  
68.94  
34.47  
6.894

# Thermoelasticity

Volumetric strain is directly proportional to temperature, so there is a thermally-induced effect in the term  $\varepsilon_{kk}$  in the linear elasticity equation.

Use the following algorithm:

1. Calculate the temperature increment  $\Delta T$  in a timestep by solving the heat equation

$$\rho c \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T)$$

by a backward Euler method. Here the temperature is usually assumed independent of the stress-strain.  $\rho$  is the density,  $c$  the specific heat,  $k$  the diffusivity.

# Thermoelasticity (continued)

2. Solve the PDE

$$\frac{\partial}{\partial x_j} \left( \mu \frac{\partial}{\partial x_j} \Delta u_i \right) + \frac{\partial}{\partial x_j} \left( \mu \frac{\partial}{\partial x_i} \Delta u_j \right) + \frac{\partial}{\partial x_i} \left( \lambda \frac{\partial}{\partial x_j} \Delta u_j \right) = \frac{\partial}{\partial x_i} \left( \frac{\alpha E \Delta T}{1 - 2\nu} \right)$$

for the incremental displacements in the current timestep.

3. Move the mesh to account for the incremental displacements.
4. Go to the next timestep.

```

% elas_th3.prb
P nu      0.33      %%% parameters for Fe
P E       209000
P Expan   1.1e-5
P mu      E/(2*(1+nu))
P lambda  mu*nu*2/(1-nu)
P k       80
P rhocp   450*7860
P dt      300.      %%% other parameters
P nstep   13

A displace
e D_j{mu}D_j U1_i + D_j{mu}D_i U1_j +\
  D_i{lambda}D_jU1_j=D_i{V201*E*Expan/(1-2*nu)}
b 1 U1={0.,0.,0.}
b 2 U1={0.,0.,0.}
b 3 U1={0.,0.,0.}
b 4 U1={0.,0.,0.}

A tempcalc
e {rhocp}U1-D_j{k*dt}D_jU1={rhocp*V202}
b 1 U1=500.
b 2 U1=500.
b 3 U1=500.
b 4 U1=500.
< run
  iter=1
  mapm
  V103=V103/10.
  mapm
  V202=25
  while iter<nstep
    step
    iter=iter+1
  endwhile
>

```

This bit of code squashes the original cube shape into a slab.

```

< step
  show iter*dt
  heat
  stretch
  V400=V400+V100
  black
  show V200
  contour 202
  shade 202
  show V400
  arrow 400
>

< heat
  tempcalc
  solve
  V301=V101-V302
  V302=V101
  show V300
  popp
>

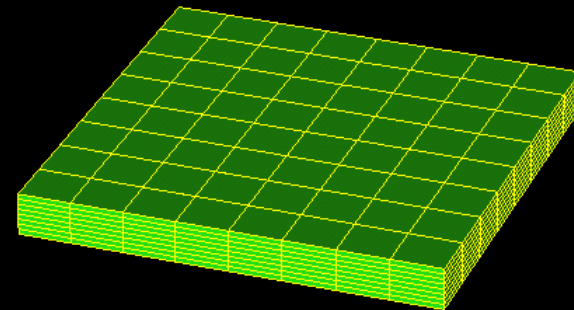
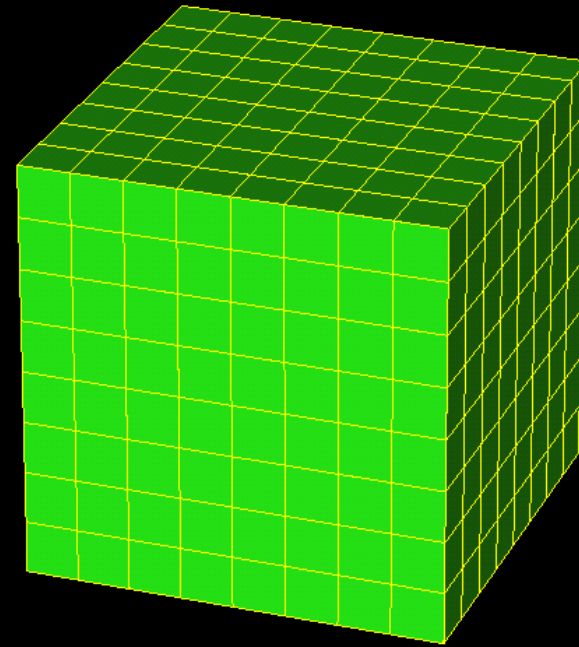
< stretch
  displace
  solve
  V200=V100
  mapm
  V100=V100+V200
  mapm
  popp
>

```

# Time-dependent thermoelasticity

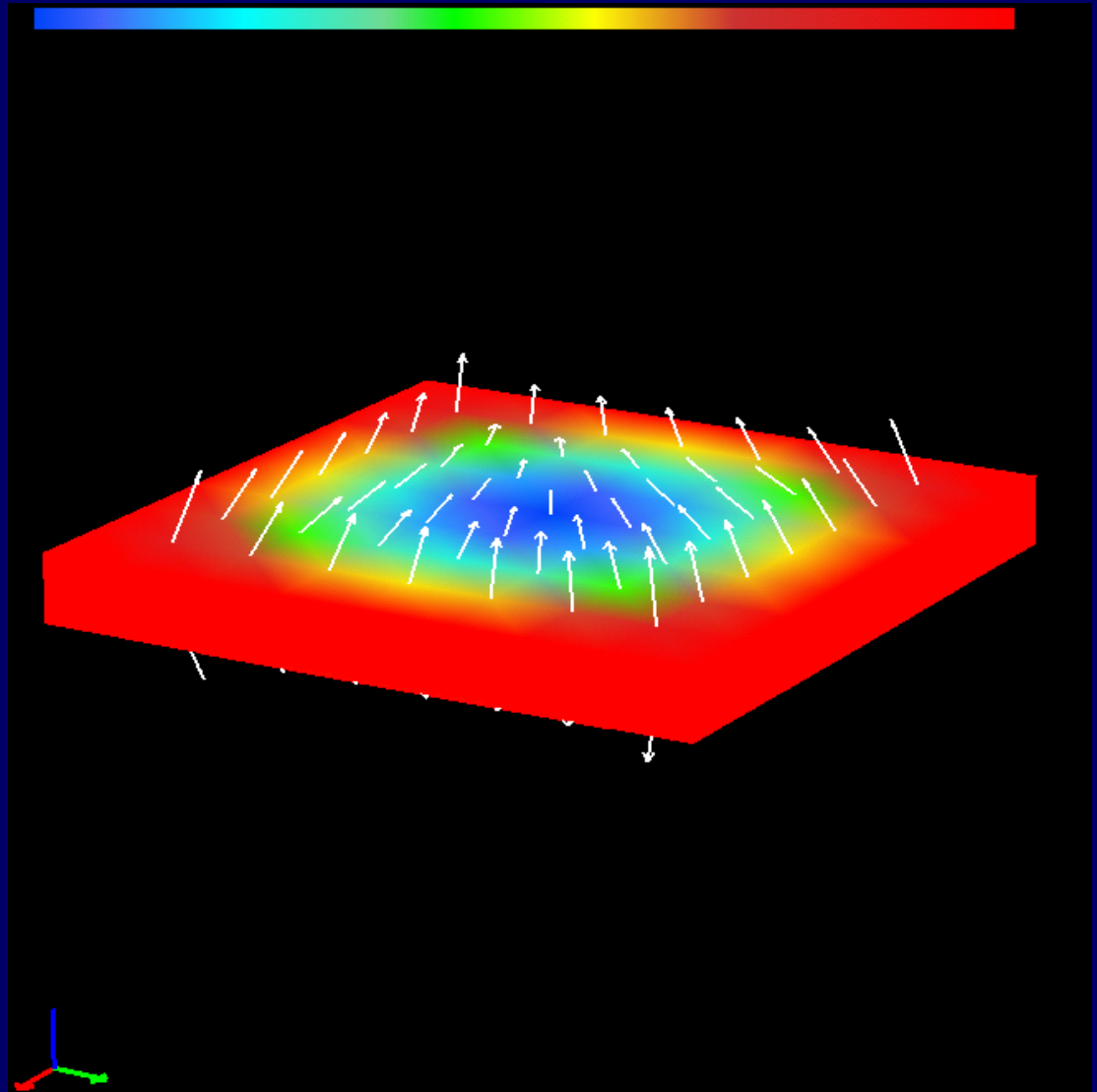
cube as generated by *Fastflo*'s block mesh generator

cube squashed into a thick slab (through the first use of the mapm" construction)



# Time-dependent thermoelasticity

- Colours indicate temperature
- Arrows indicate total displacements
- The coordinates of the mesh have changed, but can not be seen on this scale.



# Time-dependent thermoelasticity

Exercise: how would you calculate the steady state solution?

Options:

1. Use timestepping to the steady state.
2. Write the equations in a steady form; solve an enlarged set of equations.

# Elasto-viscoplasticity

Fundamental assumption is that the total rate-of-strain tensor can be separated into elastic and viscoplastic parts

$$\dot{\boldsymbol{\epsilon}} = \dot{\boldsymbol{\epsilon}}^{\text{E}} + \dot{\boldsymbol{\epsilon}}^{\text{P}}$$

We now develop a PDE for the incremental displacement between timesteps. The elastic part and viscoplastic part will be treated separately.

# Elasto-viscoplasticity (continued)

Differentiation of the elastic part with respect to time gives

$$\dot{\sigma}_{ij} = 2\mu(\dot{\varepsilon}_{ij} - \dot{\varepsilon}_{ij}^P) + \lambda \delta_{ij} (\dot{\varepsilon}_{kk} - \dot{\varepsilon}_{kk}^P)$$

which can be re-arranged to

$$\sigma_{ij}^l = \sigma_{ij}^{l-1} + \mu \left( \frac{\partial}{\partial x_j} \Delta u_i + \frac{\partial}{\partial x_i} \Delta u_j \right) + \lambda \delta_{ij} \frac{\partial}{\partial x_k} \Delta u_k - \Delta t \cdot V^{P,l-1}$$

where  $\Delta u$  is the incremental displacement in each timestep and

$$V^{P,l-1} = 2\mu \dot{\varepsilon}_{ij}^{P,l-1} + \lambda \delta_{ij} \dot{\varepsilon}_{kk}^{P,l-1}$$

accounts for the plastic strain rate at each timestep;  $l$  is the time level.

# Elasto-viscoplasticity (continued)

The force balance equation then gives

$$\frac{\partial}{\partial x_j} \left( \mu \frac{\partial}{\partial x_j} \Delta u_i \right) + \frac{\partial}{\partial x_j} \left( \mu \frac{\partial}{\partial x_i} \Delta u_j \right) + \frac{\partial}{\partial x_i} \left( \lambda \frac{\partial}{\partial x_j} \Delta u_j \right) + \underline{\text{div } \sigma^{l-1} + \mathbf{F}}$$
$$= \text{div}(\Delta t.V^{P,l-1})$$

This is the basic PDE to be solved each timestep. The underlined terms cancel for  $l > 0$  because the body force is assumed to be independent of time. (We are solving for the increments, and the body force is already taken into account in calculating  $\sigma^0$ .)

Note that the LHS is the same as the usual elasticity equation.

# Elasto-viscoplasticity (continued)

Model for the plastic strain rate:

$$\dot{\varepsilon}^P = \gamma \Phi(F) \frac{\partial F}{\partial \sigma}$$

where

$$\Phi(F) = \begin{cases} \left(\frac{F}{C_Y}\right)^N & \text{for } F > 0 \\ 0 & \text{for } F < 0 \end{cases}$$

$$\frac{\partial F}{\partial \sigma} = \begin{pmatrix} \sqrt{3} \sigma'_{11} / 2\bar{\sigma} & \sqrt{3} \sigma'_{12} / \bar{\sigma} \\ \sqrt{3} \sigma'_{21} / \bar{\sigma} & \sqrt{3} \sigma'_{22} / 2\bar{\sigma} \end{pmatrix}$$

and

$$F = \sqrt{3} \bar{\sigma} - C_Y$$

(von Mises yield condition)

$$\bar{\sigma} = \sqrt{\frac{1}{2} \sigma'_{ij} \sigma'_{ij}}$$

$$\sigma'_{ij} = \sigma - \frac{1}{2} \delta_{ij} (\sigma_{kk})$$

(deviatoric stress tensor)

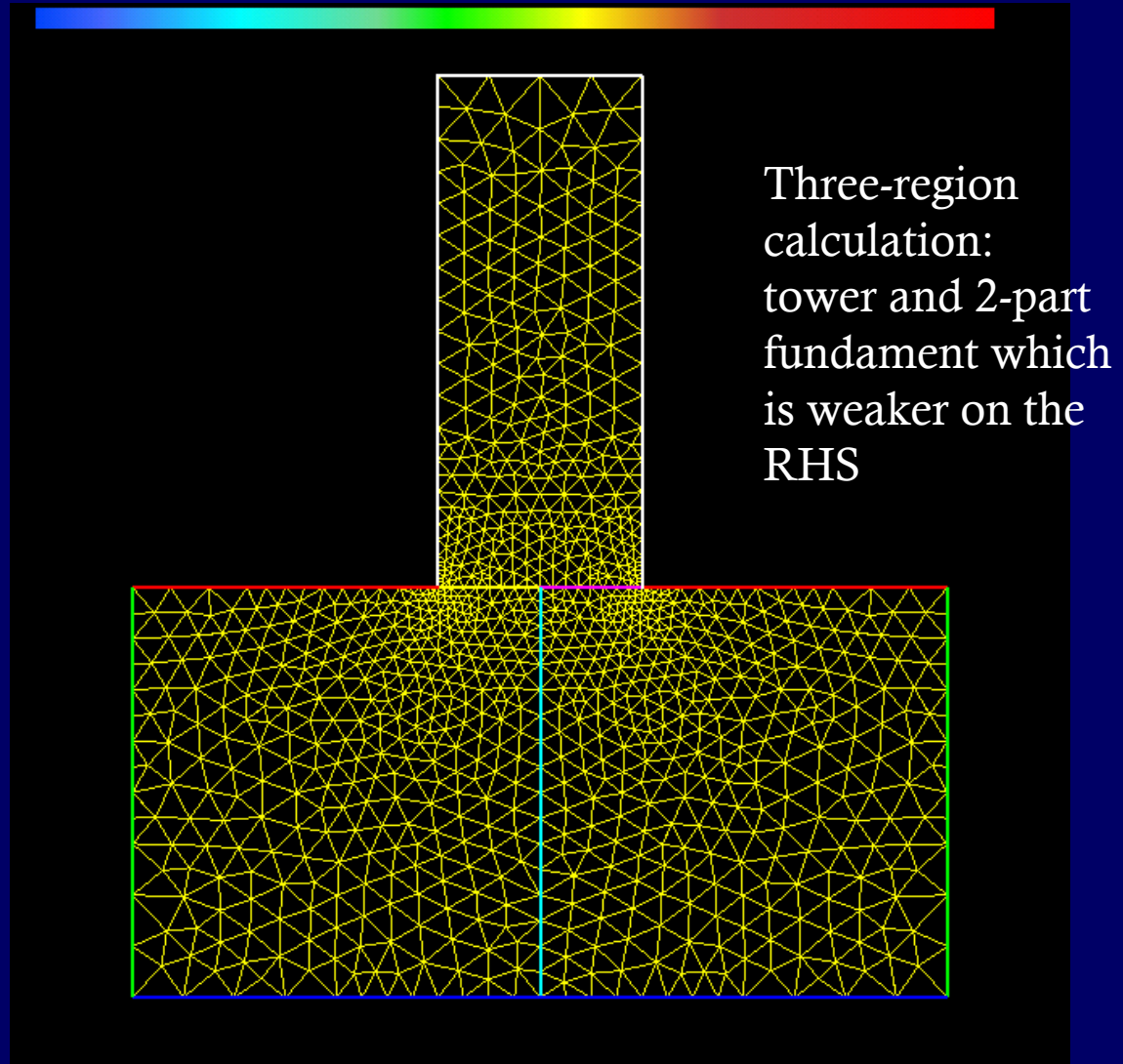
# Elasto-viscoplasticity (continued)

## Algorithm:

1. Evaluate RHS term of basic PDE.
2. Solve basic PDE for incremental displacements.
3. Calculate the stresses, move the mesh to account for incremental displacements.
4. Calculate the new viscoplastic term  $V^P$ .
5. Proceed with the next timestep.

# Simple model of “leaning tower of Pisa”

```
% elvipla.msh
1000 0 3
-2 0 0 0 0
4 8
List of vertices
0.0 0.0 %1
8.0 0.0 %2
8.0 8.0 %3
6.0 8.0 %4
0.0 8.0 %5
16.0 0.0 %6
16.0 8.0 %7
10.0 8.0 %8
6.0 18.0 %9
10.0 18.0 %10
List of boundary tags
1 3
2 6
3 4
4 1
5 2
0 0
2 3
6 2
7 1
8 5
3 6
0 0
3 5
8 7
10 8
9 7
4 4
End of boundary list
```



```

% elvipla.prb

% list of parameters, including regionally dependent parameters

P      dt                0.005
P 1    rhog              3*1
P 2    rhog              3*1
P 3    rhog              10*1
P 1    nu                0.3
P 2    nu                0.3
P 3    nu                0.3
P 1    E                 5000
P 2    E                 5000
P 3    E                 20000
P 1    Cy                200
P 2    Cy                100
P 3    Cy                3000
P 1    mu                E_1/(2*(1+nu_1))
P 2    mu                E_2/(2*(1+nu_2))
P 3    mu                E_3/(2*(1+nu_3))
P 1    lambda            mu_1*nu_1*2/(1-2*nu_1)
P 2    lambda            mu_2*nu_2*2/(1-2*nu_2)
P 3    lambda            mu_3*nu_3*2/(1-2*nu_3)
P 1    gamma            1.
P 2    gamma            1.
P 3    gamma            1.
P 1    Npowerlaw        1.
P 2    Npowerlaw        1.
P 3    Npowerlaw        1.
P      R3                sqrt 3

```

```
% elvipla.prb
```

```
% list of problems
```

```
A EQNelastic0
```

```
e  $D_j\{\mu\}D_jU1_i + D_j\{\mu\}D_iU1_j + D_i\{\lambda\} D_j U1_j = -\{0.0,-\rho g\}_i$   
b 3 U1 = {0.0,0.0}
```

```
A EQNelastic
```

```
e  $U1_i = + \{dt*V1401,dt*V1402\}_i$   
b 3 U1 = {0.0,0.0}
```

```
A EQNstress1
```

```
e  $U1_i = [grad] \{V201\}_i$ 
```

```
A EQNstress2
```

```
e  $U1_i = [grad] \{V202\}_i$ 
```

```
A EQNsigma11
```

```
e  $V001 = V301 + 2*\mu*V801 + \lambda*(V801+V902) - dt*V501$ 
```

```
A EQNsigma12
```

```
e  $V001 = V302 + \mu*(V802 + V901) - dt*V502$ 
```

```
A EQNsigma21
```

```
e  $V001 = V401 + \mu*(V901 + V802) - dt*V601$ 
```

```
A EQNsigma22
```

```
e  $V001 = V402 + 2*\mu*V902 + \lambda*(V801+V902) - dt*V602$ 
```

```
A EQNvonMises
```

```
e  $V001 = R3*V1002 - Cy$ 
```

```
% elvipla.prb
```

```
% list of problems (continued)
```

```
A EQNviscoplastic1
```

```
e V1201 = gamma*(V1102/Cy)^Npowerlaw * R3*(V301-0.5*V1001)/(2*V1002)
```

```
e V1202 = gamma*(V1102/Cy)^Npowerlaw * R3*V302/V1002
```

```
e V1302 = gamma*(V1102/Cy)^Npowerlaw * R3*(V402-0.5*V1001)/(2*V1002)
```

```
e U1_1 = {2*mu*V1201 + lambda*(V1201+V1302)}
```

```
e U1_2 = {2*mu*V1202}
```

```
A EQNviscoplastic2
```

```
e V1201 = gamma*(V1102/Cy)^Npowerlaw * R3*(V301-0.5*V1001)/(2*V1002)
```

```
e V1301 = gamma*(V1102/Cy)^Npowerlaw * R3*V401/V1002
```

```
e V1302 = gamma*(V1102/Cy)^Npowerlaw * R3*(V402-0.5*V1001)/(2*V1002)
```

```
e U1_1 = {2*mu*V1301}
```

```
e U1_2 = {2*mu*V1302 + lambda*(V1201+V1302)}
```

```
A EQNdivergence1
```

```
e U1 = [div] {V500}
```

```
A EQNdivergence2
```

```
e U1 = [div] {V600}
```

```
% elvipla.prb

% list of macros

< run
  nostack
  initialise
  iteration = 0
  while iteration < 250
    onestep
  endwhile
>

< initialise
% calculate displacements in linear theory; store matrix factors
  EQNelastic0
  solve
  saveF 1
  V200 = V100
  V1500 = V100
  show V201
  show V202
  arrow 200
% calculate matrix factors for stress equation
  EQNstress1
  solve
  saveF 2
% move mesh and evaluate terms for entry into timestep loop
  NEWmesh
  EVALstress
  EVALvonMises
  EVALviscoplastic
>
```

```
% elvipla.prb

% list of macros (continued)

< onestep
  iteration = iteration + 1
  show iteration
  EVALdivergence
  CALCdisplacement
  NEWmesh
  EVALstress
  EVALvonMises
  EVALviscoplastic
>

< EVALdivergence
% V1401 = D_j(VP_1j), V1402 = D_j(VP_2j)
  EQNdivergence1 1 500 V500
  solve
  V1401 = V101
  EQNdivergence2 1 600 V600
  solve
  V1402 = V101

< NEWmesh
% adjusts the mesh to account for elasto-viscoplastic displacements
  mapm 1600
  V1600 = V1600 + V200
  mapm 1600
>
```

```

% elvipla.prb (list of macros, continued)
< CALCdisplacement % solves stress equation for displacements
  EQNelastic 1 1400 V1400
  getF 1
  resolve
  V200 = V100
  V1500 = V1500 + V100
  black
  arrow 200
  black
  arrow 1500
  show V200
  show V1500
  NormOfDisplacement = (L1 V101) + (L1 V102)
  show NormOfDisplacement
>

< EVALstress
% V300,400 = elastic stress tensor including viscoplastic stresses
  EQNstress1
  getF 2
  resolve
  V800 = V100
  EQNstress2
  getF 2
  resolve
  V900 = V100
  EQNsigma11 1 300 V300 500 V500 800 V800 900 V900
  V301 = V101
  EQNsigma12 1 300 V300 500 V500 800 V800 900 V900
  V302 = V101
  EQNsigma21 1 400 V400 600 V600 800 V800 900 V900
  V401 = V101
  EQNsigma22 1 400 V400 600 V600 800 V800 900 V900
  V402 = V101
>

```

```

% elvipla.prb

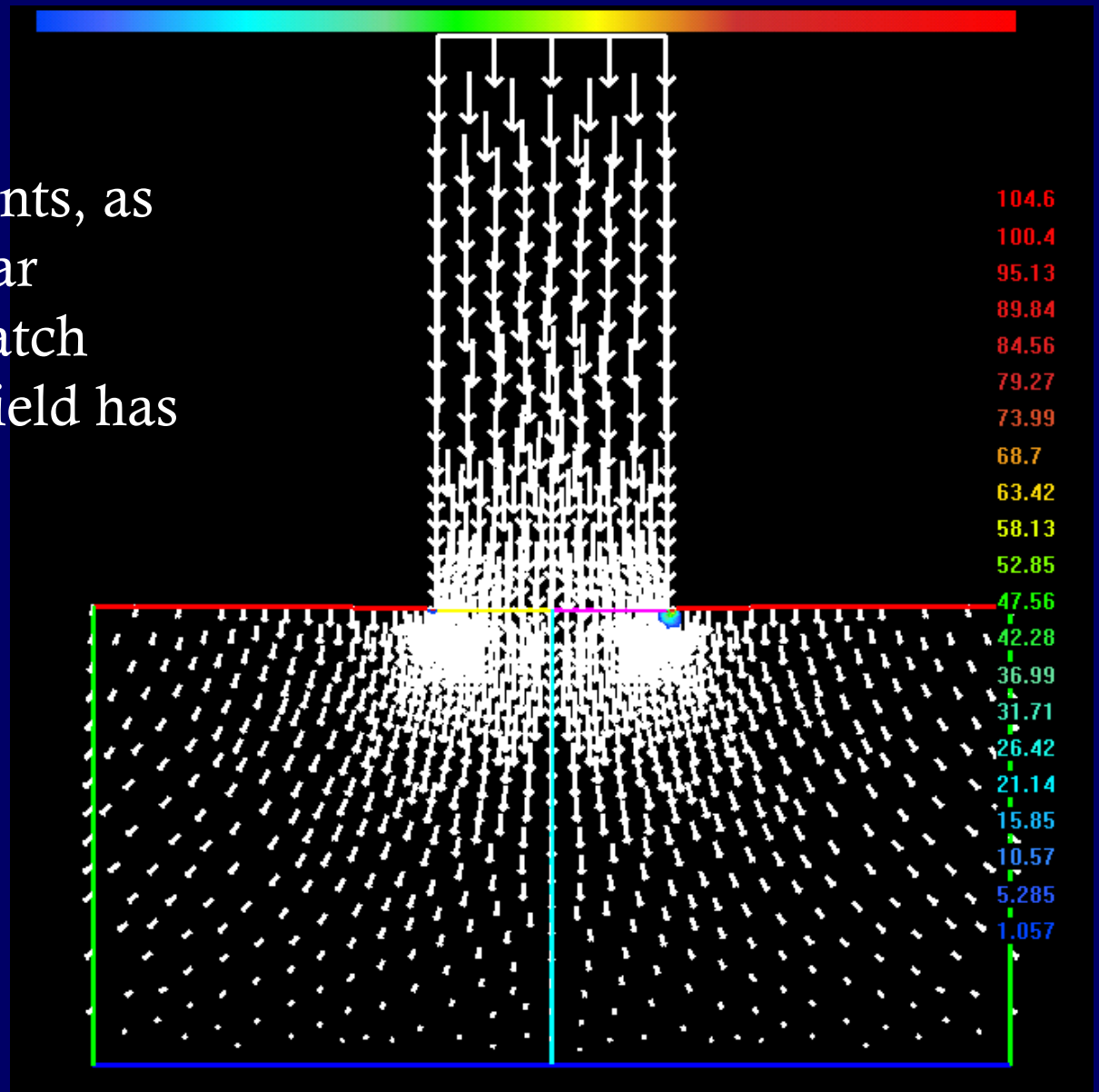
% list of macros (continued)

< EVALvonMises
% evaluates terms required in von Mises yield criterion
% V1001 = trace(sigma), V1002 = sigma_bar
% V1101 = von Mises function F
% V1102 = F when F>0, 0 when F<0
    V1001 = V301 + V402
    V1002 = sqrt(0.5*((V301-0.5*V1001)^2 + V302^2 + V401^2 + (V402-0.5*V1001)^2)
    EQNvonMises 1 1000 V1000
    V1101 = V101
    V1102 = V1101*(V1101>0.0)
    contour 1102
>

< EVALviscoplastic
% V500,600 = VP_ij (occurs in viscoplastic part of stress tensor)
% V1200,1300 = (epsilon^P)^dot_ij = plastic strain-rate tensor
    EQNviscoplastic1 1 300 V300 400 V400 1000 V1000 1100 V1100
    getF 2
    resolve
    V500 = V100
    EQNviscoplastic2 1 300 V300 400 V400 1000 V1000 1100 V1100
    getF 2
    resolve
    V600 = V100
>

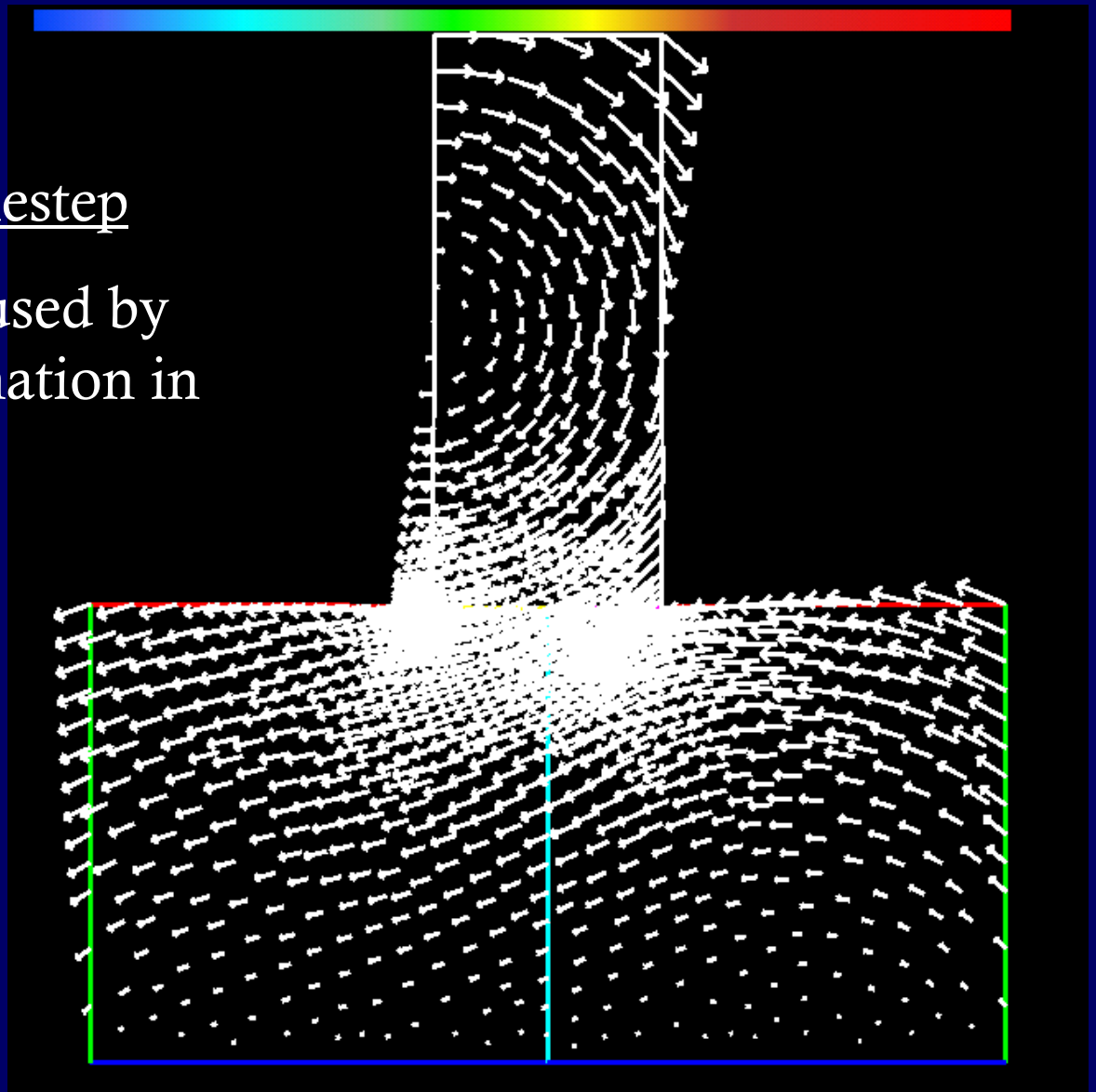
```

Initial displacements, as computed by linear elasticity. Blue patch indicates where yield has occurred.



Results for first timestep

Displacements, caused by viscoplastic deformation in yielding region



## Further results and discussion on elvipla.prb

- For the parameters shown, the code will converge slowly. The parameters for this example are artificial.
- There is a timestep restriction for this explicit algorithm.
- The code looks rather cumbersome and long. This has been done in part to avoid solving overly large matrix problems. (It's best to adopt a segregated strategy so as to minimise the size of matrices being solved.)
- The code provides the basis for other elasto –viscoplastic calculations, e.g. extrusion, forging, stamping, ...

# Summary of presentation

- *Fastflo* – summary of features relevant to elasticity and associated topics
- 3 examples
  1. straightforward linear elasticity
  2. time-dependent thermoelasticity
  3. elasto - viscoplasticity