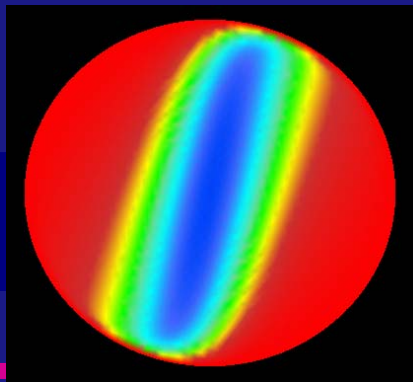


Application of *Fastflo* to porous media problems

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 porous media problems
- 4 examples
 1. flow through a saturated porous medium; effect of stress on permeability
 2. effect of buoyancy
 3. calculation of the free surface between saturated and dry soil
 4. porous media flow driven by liquid distribution system

Porous media 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* (continued)

- 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.

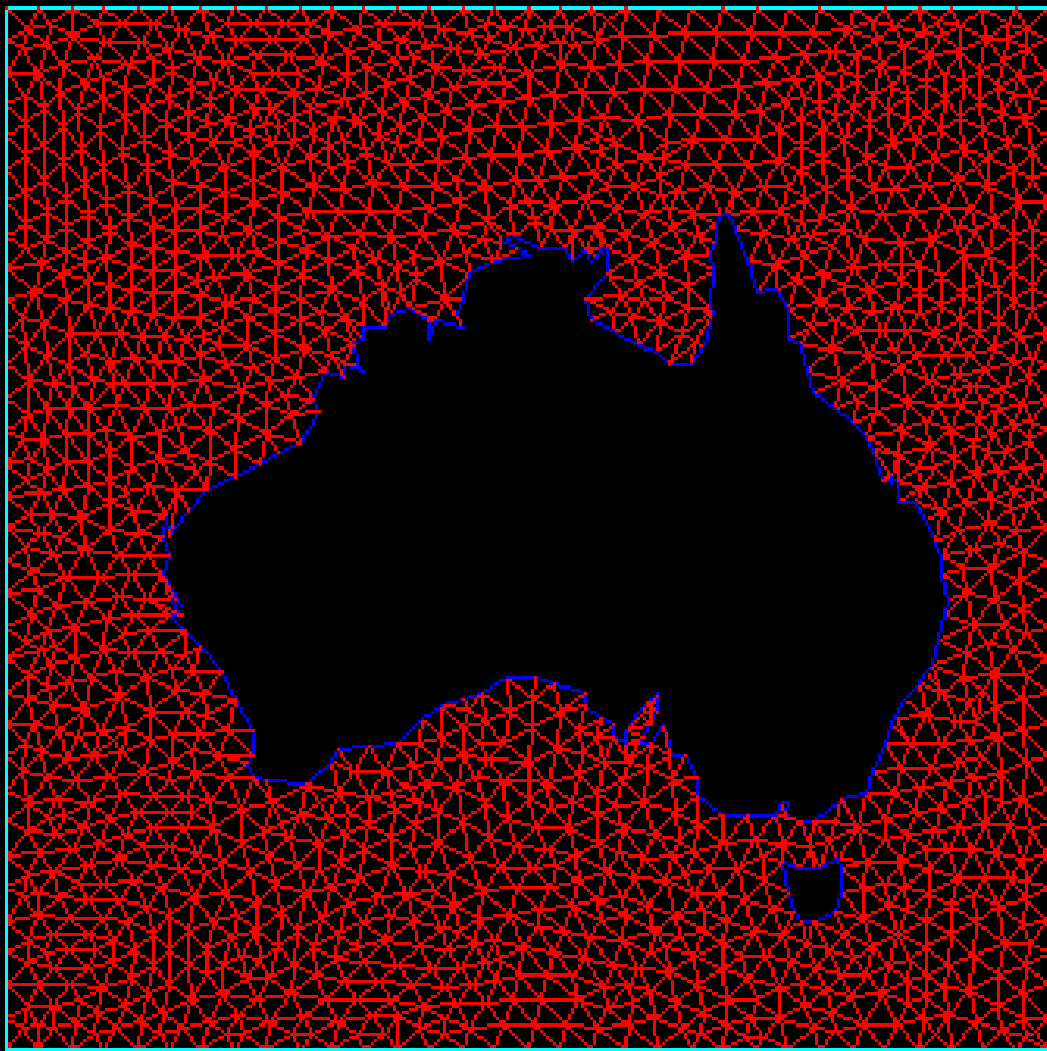
Design features of *Fastflo*

- 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* (continued)

- *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

Model equations for saturated flow

$$\rho \left(\frac{1}{\Phi} \frac{\partial u_i}{\partial t} + C_F u_j \frac{\partial}{\partial x_j} u_i \right) + C_D \frac{\mu}{\kappa} u_i = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left(\mu \left[\frac{\partial}{\partial x_j} u_i + \frac{\partial}{\partial x_i} u_j \right] \right) + B_i$$
$$\frac{\partial u_j}{\partial x_j} = 0$$

ρ = density, Φ = porosity, μ = viscosity

$C_F = 1$ in free liquid, 0 in porous material

$C_D = 0$ in free liquid, μ/κ in porous material

Retain usual viscous terms in porous material, even though they are negligible compared to Darcy-Forchheimer term.

Simplifications

$$\frac{\mu}{\kappa} u_i = -\frac{\partial p}{\partial x_i} - \rho g$$

$$\nabla \left(\frac{\kappa}{\mu} \nabla p \right) = 0$$

Incorporate gravity effect
in modified pressure

$$p = p_0 + \rho g z$$

Complications - stress

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

Kozeny-Carmen equation
for permeability

$$\kappa = \frac{d_p^2}{180} \frac{\varepsilon^3}{(1-\varepsilon)^2}$$

ε is the intergranular void fraction

$$\varepsilon = (e_0 + dv/v) / (1 + dv/v)$$

where dv/v is the volumetric strain; d_p is the particle size

Complications - buoyancy

Body force in stress equation

$$B_i = \alpha \rho g (\theta - \theta_0)$$

Energy balance
equation for
temperature θ

$$\rho c \left(\frac{\partial \theta}{\partial t} + \mathbf{v} \cdot \nabla \theta \right) = \nabla \cdot (k \nabla \theta)$$

Many other complications are possible, notably chemical reactions, nonlinear dependence of parameters, partially saturated regions, and demarcation between saturated and other regions.

Algorithms

- always reduce the problem to a set of linear equations, which the FE representation reduces to a large sparse system
- for nonlinear problems – introduce iterative scheme, typically Picard
- for time-dependent problems – introduce suitable timestepping scheme; implicit schemes (e.g. Crank-Nicolson, Backward Euler) are most commonly used
- for related flow calculations, penalty or augmented Lagrangian methods are simple and generally effective

Boundary conditions

- need to understand principles of FE method, which involves integration by parts
- example – the heat equation

$$\text{equation: } \rho c \frac{\partial \theta}{\partial t} = \nabla \cdot (k \nabla \theta)$$

$$\text{backward Euler scheme: } \left(\frac{\rho c}{dt} \right) (\theta^n - \theta^{n-1}) = \nabla \cdot (k \nabla \theta^n)$$

$$\left(\frac{\rho c}{dt} \right) \theta^n - \nabla \cdot (k \nabla \theta^n) = \left(\frac{\rho c}{dt} \right) \theta^{n-1}$$

natural boundary conditions are applied to $k \nabla \theta$
(which results from integration by parts)

Boundary conditions (continued)

The principal options are:

- do nothing – equivalent to natural boundary expression is zero (e.g. zero heat flux)
- supply alternative value/function for natural boundary expression (e.g. non-zero heat flux)
- apply Dirichlet condition (e.g. temperature)

$$U1 = \{\text{expression}\}$$

Coding the algorithm

The principal steps usually are ...

- declare parameters
- define “problems” = equations + BC’s
- type the name of the problem to assemble the FE system; type solve to solve the sparse matrix
- generally, manage the computations (e.g. assembly solving, timestepping, iteration, error control, graphics, file management, ...) within “macros”

Derivative expressions

38 expressions hard-wired into the package

$$D_j A D_j U1 \quad - \nabla \cdot (a \nabla u)$$

$$A_j D_j U1_i \quad a \cdot \nabla u$$

$$D_i A D_j U1_j \quad - \nabla (a \nabla \cdot u)$$

| | | |
|----|-----------------------|--|
| 1 | $D_j A D_j U1$ | $-\nabla \cdot (a \nabla u)$ |
| 2 | $A U1$ | au |
| 3 | $A_j D_j U1$ | $a \cdot \nabla u$ |
| 4 | $D_j A_j U1$ | $-\nabla \cdot (au)$ |
| 5 | $D_j A_{jk} D_k U1$ | $-\nabla \cdot (A \nabla u)$ |
| 6 | $D_j A U1_j$ | $-\text{div}(au)$ |
| 7 | $A D_j U1_j$ | $a \text{ div } u$ |
| 8 | $A_j U1_j$ | $a \cdot u$ |
| 9 | $D_j A_k D_k U1_j$ | $-\nabla (a \cdot \nabla u)$ |
| 10 | $D_j A_j D_k U1_k$ | $-\text{div}(a \text{ div } u)$ |
| 11 | $D_j A_{jk} U1_k$ | $-\text{div}(A u)$ |
| 12 | $A_{jk} D_j U1_k$ | $\text{div}(A u)$ |
| 13 | $D_i A U1$ | $-\nabla (au)$ |
| 14 | $A D_i U1$ | $a \nabla u$ |
| 15 | $A_i U1$ | au |
| 16 | $D_i A_j D_j U1$ | $-\nabla (a \cdot \nabla u)$ |
| 17 | $D_j A_j D_i U1$ | $-\mathbf{a} \cdot \nabla (\nabla u) - (\nabla u) \cdot \nabla \mathbf{a}$ |
| 18 | $D_j A_{ji} U1$ | $-\nabla \cdot (A u)$ |
| 19 | $A_{ij} D_j U1$ | $A \nabla u$ |
| 20 | $A U1_i$ | au |
| 21 | $A_j D_j U1_i$ | $a \cdot \nabla u$ |
| 22 | $D_j A_j U1_i$ | $-\mathbf{a} \cdot \nabla u - u \text{ div } \mathbf{a}$ |
| 23 | $D_j A D_j U1_i$ | $-\nabla \cdot (a \nabla u)$ |
| 24 | $D_j A_{jk} D_k U1_i$ | $-\nabla \cdot (A \nabla u)$ |
| 25 | $D_i A D_j U1_j$ | $-\nabla (a \nabla \cdot u)$ |
| 26 | $D_i A_j U1_j$ | $-\nabla (a \cdot u)$ |
| 27 | $D_j A D_i U1_j$ | $(\nabla a) \cdot \nabla (\text{div } u) - \nabla (\text{div } au)$ |
| 28 | $A_j D_i U1_j$ | $\mathbf{a} \cdot (\nabla u)$ |
| 29 | $D_j A_i U1_j$ | $-\mathbf{a} (\nabla \cdot u) - u \cdot \nabla \mathbf{a}$ |
| 30 | $A_i D_j U1_j$ | $\mathbf{a} (\nabla \cdot u)$ |
| 31 | $A_{ij} U1_j$ | $A u$ |
| 32 | $D_i A_{jk} D_j U1_k$ | $-\nabla \cdot (A \nabla u)$ |
| 33 | $D_j A_{jk} D_i U1_k$ | |
| 34 | $D_j A_{ik} D_k U1_j$ | |
| 35 | $D_j A_{ij} D_k U1_k$ | |
| 36 | $D_j A_{ik} D_j U1_k$ | |
| 37 | $D_j A_k D_j U1_k$ | $-\text{div } \mathbf{a} \nabla u$ |
| 38 | $D_j A_i D_j U1$ | $-\text{div } \mathbf{a} \nabla u$ |

Example 1(a): flow under a dam

```
% dam-new.prb
P   rho      1000.0      %kg/m^2
P   g 9.81      %m/s^2
P   visc     0.0015     %kg/(m.s)
P   kappa    1.e-8      %m^2
P   konmu    kappa/visc
P   p0       30.0*rho*g  %Pa
P   p1       4.0*rho*g  %Pa

A   Darcy
e   D_j{kappa}D_jU1 = {0.0}
b   2       U1 = {p1}
b   3       U1 = {p0}

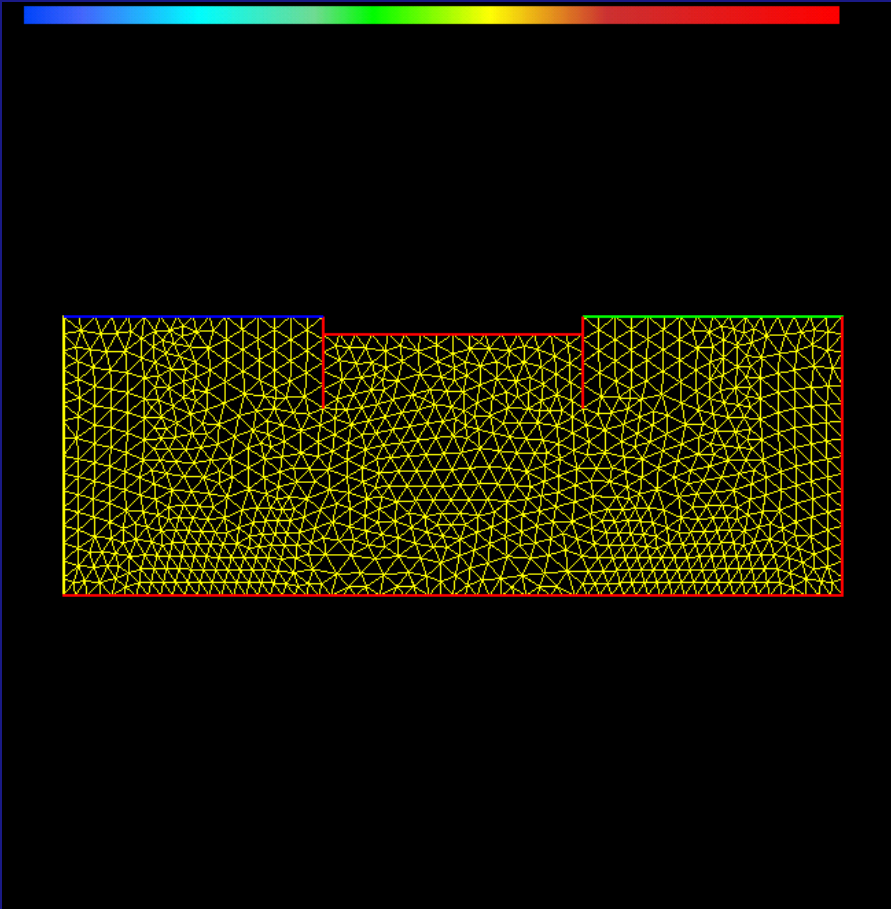
A   calc
e   V000 = [grad] {-konmu*V101}

A   strf
e   D_jD_jU1 = \
D_j{0.0,1.0,-1.0,0.0}_jk {V100}_k
b   4       U1 = {0.0}
```

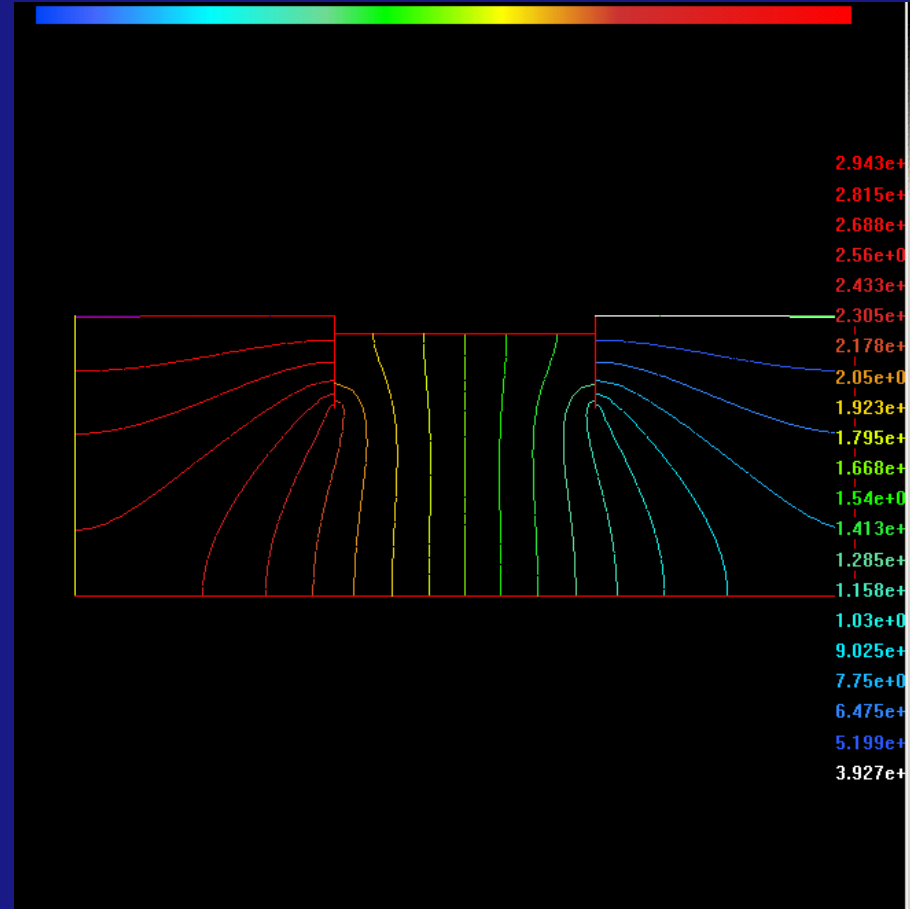
```
< run1
    Darcy
    solve
    black
    contour
>

< run2
    calc
    black
    arrow
>

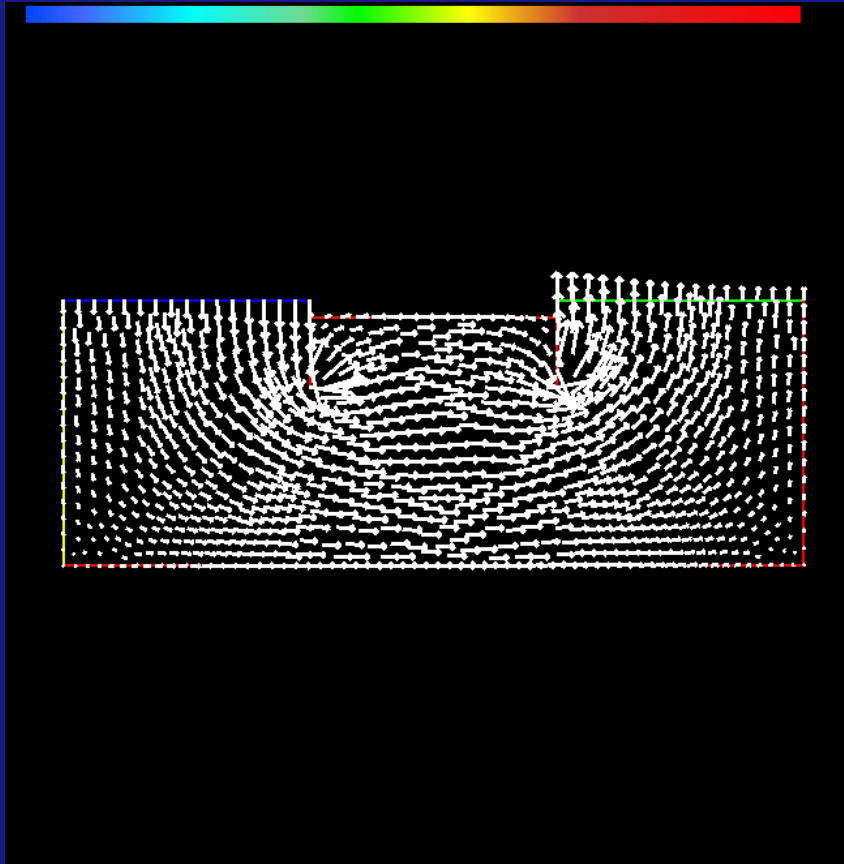
< run3
    strf
    solve
    black
    contour
>
```



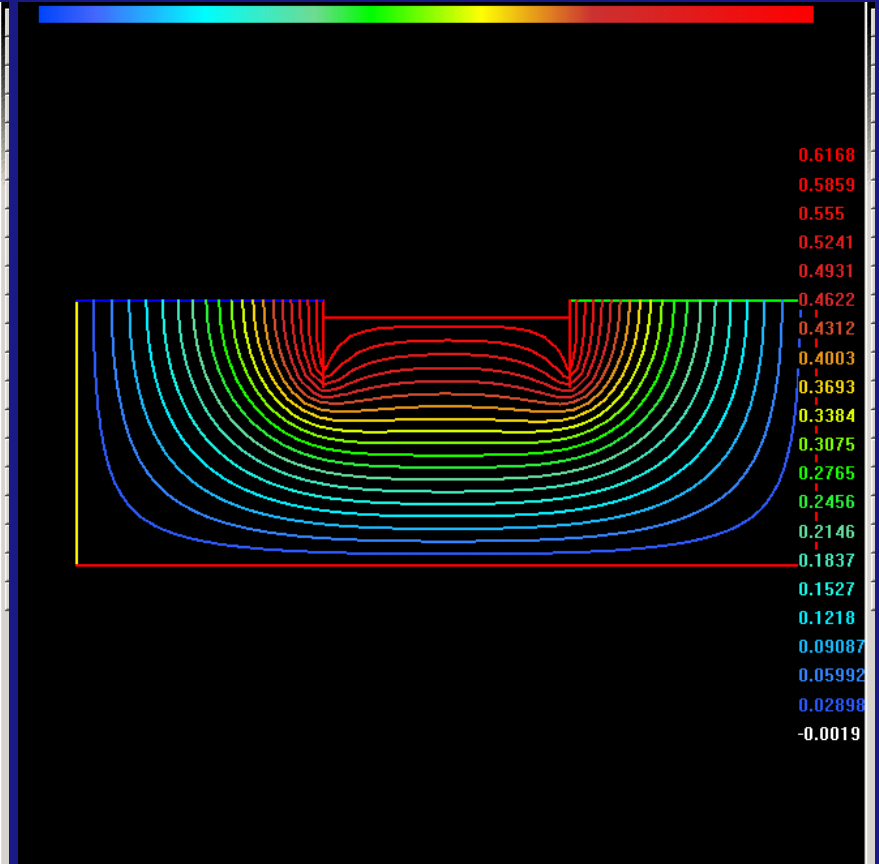
mesh



run1 – potential contours



run2 – arrow
plot of velocity



run3 – streamlines

Example 1(b): effect of stress under dam

- include calculation for stress caused by weight of dam
- apply Kozeny-Carmen equation for permeability
- can then compare flow features, with and without stress
- further complications can be included, especially deformation of the porous material by the fluid stress
- for details, see [dam-new-stress](#)

Example 2: effect of buoyancy

- in the stress balance equation, include buoyancy term, e.g. caused by temperature of fluid
- need to solve an additional equation for the temperature
- use penalty method for timestepping
- for details, see

conv-lam (laminar natural convection)

conv-porous (buoyancy in porous media)

```

% conv-porous.prb
% transient simulation of natural convection for Darcy flow
% dimensional formulation
% timestepping based on the penalty method
% large cavity filled with porous material saturated with water
% wall on right is 50K above ambient
% wall on left is 50K below ambient

```

```

P      rho      1000.0      %kg/m^2
P      g        9.81       %m/s^2
P      visc     0.01       %kg/(m.s)
P      kappa    1.e-10     %m^2
P      cp       4.2e3      %J/(kg.K)
P      delt     1e7.
P      endT     1e8
P      diffu    0.6       %W/(m.s.K)
P      Tcoeff   0.0002
P      Niter    2

P      muonk    visc/kappa
P      dondt    rho/delt      %kg/(m^3.s)
P      rhocp    rho*cp
P      rcondt   rhocp/delt
P      grho     g*rho
P      alpha    g*rho*Tcoeff
P      Pen      1          e11

D      1        all
D      2        all
D      3        all
D      4        all

```

```

A momentum
e {dondt}U1_i - {dondt}{V200}_i + {muonk}U1_i \
  = - D_i{V301} + D_i{Pen}D_jU1_j + D_j{visc}D_jU1_i \
  + {0,alpha*V401}_i
b all U1 = {0.,0.}

```

```

A heat
e {rcondt}U1 - {rcondt}{V401} + {rhocp*V200}_jD_jU1 \
  = D_j{diffu}D_jU1
b 2 U1 = {50}
b 4 U1 = {-50}

```

```


< run
  init
  while t<endT
    t = t + delt
    show t
    onestep
  endwhile
>

```

```

< init
  nostack
  t = 0
  V200_i = {0.,0.}
  V301 = {0}
  V401 = {0}
>

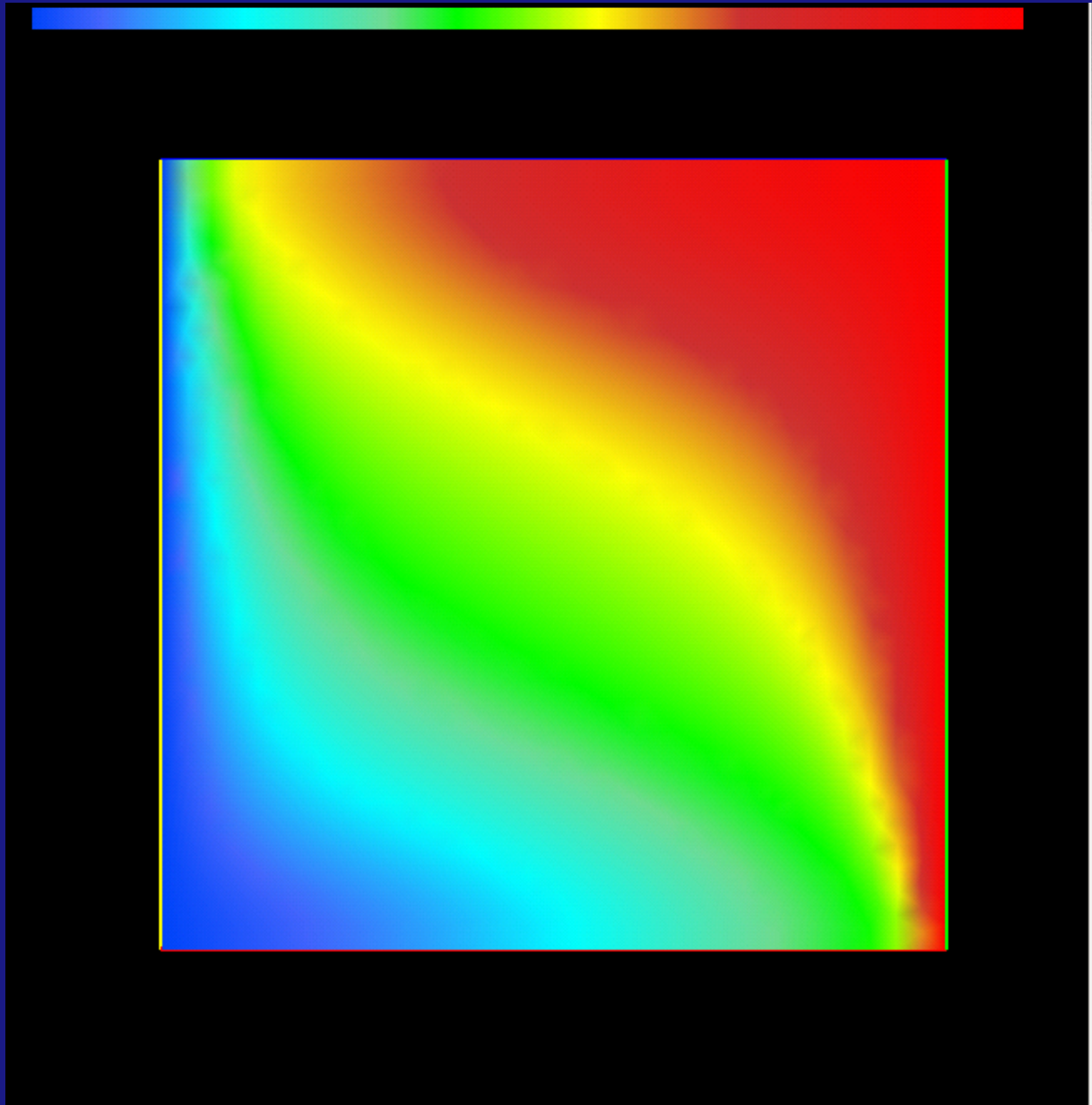
```



```

< onestep
  iter = 0
  heat 1 200 V200 401 V401
  solve
  black
  V401 = V101
  shade 401
  momentum 1 200 V200 301 V301 401 V401
  solve
  black
  V200 = V100
  arrow 200
>

```



conv-porous

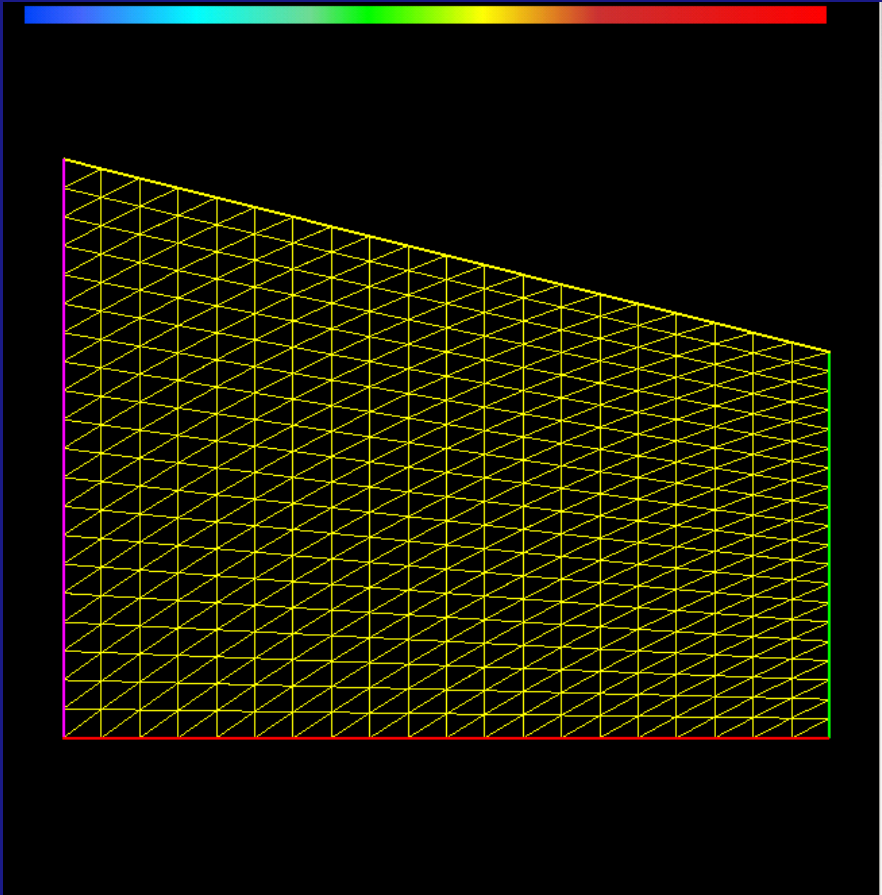
temperature
contours after 7
timesteps

Example 3: free surface between dry material and saturated flow

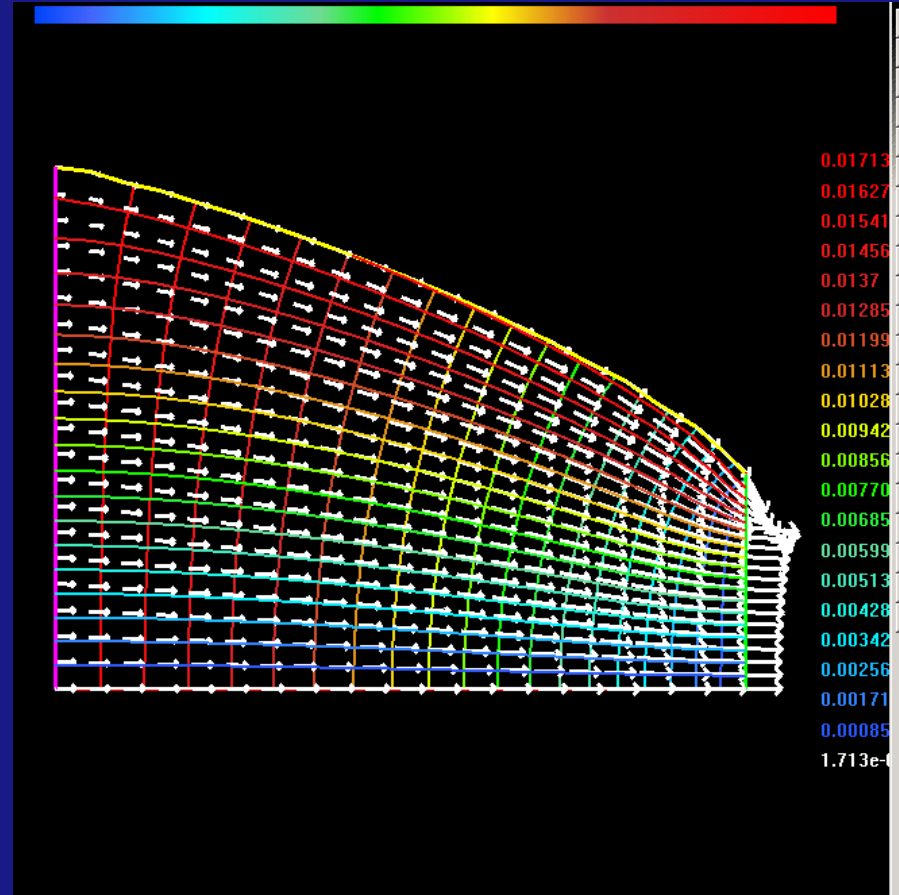
- here we solve the “underground dam problem”, that is calculate the phreatic surface between dry material and saturated flow
- the calculation has the main components
 - calculate Darcy flow
 - calculate flux at the top of the saturated region
 - use this flux as the forcing term in a calculation to deform the mesh to the saturated region

Example 3 (continued)

- iterate between Darcy and mesh deformation
- mesh deformation is based on deformation of elastic region, with artificial deformation force
- for details, see [phreatic](#)



phreatic – mesh
after first mapping



arrow plot of velocity;
pressure; streamlines

Example 4: porous media flow fed by distribution system

- this example requires solution of the full stress balance equation, with regionally dependent terms
- timestepping by Augmented Lagrangian method
- there is a major change in flow character from medium Reynolds number flow with small pressure changes and important nonlinearities, to Darcy flow with big pressure changes and no nonlinearity
- for details, see [freeDarcy](#)

```

% freeDarcy.prb

P   velin   0.3           %m/s
P   kappa   1.e-10        %m^2
P   mu       1.e-3        %kg/(m.s)
P   rho      1000.        %kg/m^3
P   Nstep    50
P   Niter    5
P   delT     0.01         %s
P   dondt    rho/delT

D   1        inlet
D   2        outlet
D   5        wall
D   6        wall
D   3        wall
D   4        wall
D   7        join

P   1        mubykap     0.
P   2        mubykap     mu/kappa
P   1        convpar     rho
P   2        convpar     0.
P   1        Pen         4000.
P   2        Pen         4000.

```

```

A   augment
e   {dondt}U1_i - {dondt}{V500}_i \
    + {convpar*V200}_jD_jU1_i \
    + {mubykap}U1_i \
= - D_i{V301} + D_i{Pen}D_jU1_j \
    + D_j{mu}D_jU1_i + D_j{mu}D_iU1_j
b   wall     U1={0.0,0.0}
b   inlet    U1={velin,0.}

A   [reduced]   delp
e   U1={Pen}D_j{-V200}_j

A   fluxL
b   inlet      [integrated] {V201}

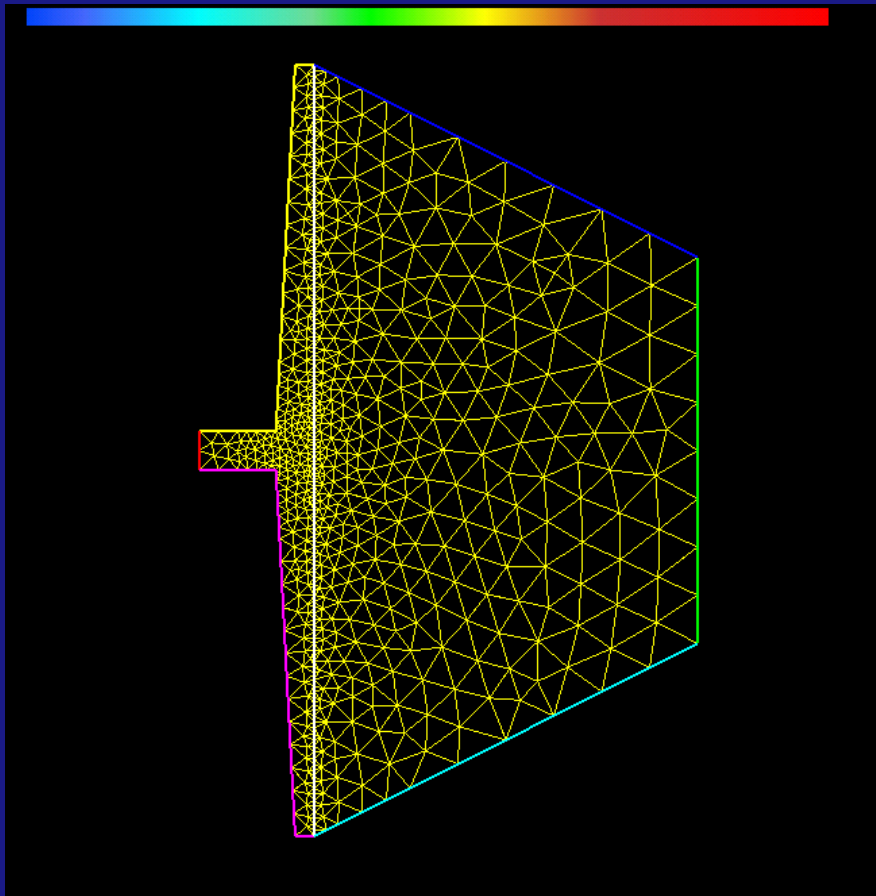
A   fluxR
b   outlet     [integrated] {V201}

A   strf
e   D_jD_jU1=[curlv] {V200}
b   5   U1={0.0}
b   6   U1={0.0}

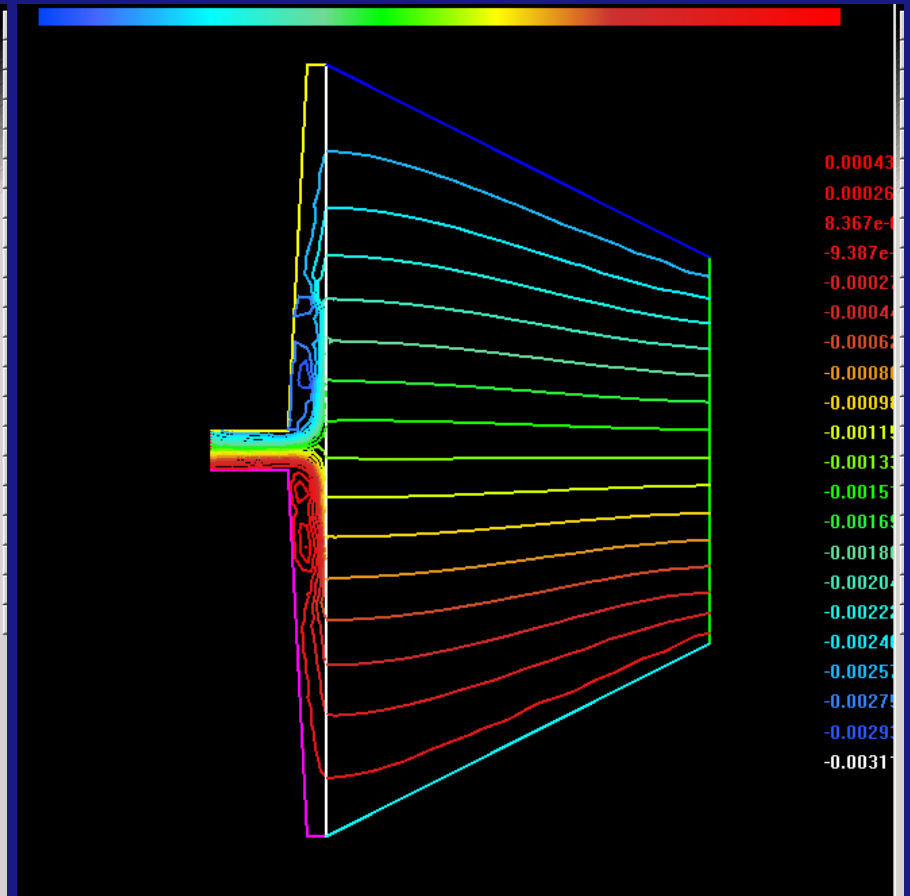
```

```
< run
  nostack
  istep = 0
  V200 = {0., 0.}
  V500 = {0., 0.}
  while istep < Nstep
    istep = istep + 1
  !# *** commencing new timestep
    show istep
    iter = 0
      while iter < Niter
        iter = iter + 1
        augment 1 200 V200 301 V301 500 V500
        solve
        V200 = V100
        delp
        solve
        expand 1
        pressincr = L1 V101
        show pressincr
        V301 = V301 + V101
        black
        arrow 200
        contour 301
      endwhile
    stream
    V600 = V500 - V200
    Vstep = L1 V600
    show Vstep
    V500 = V200
```

```
fluxL
fluxin = Out
fluxR
fluxout = Out
!# check on mass conservation
fluxratio = fluxin/fluxout
show fluxratio
endwhile
>
< stream
  strf
  solve
  black
  contour 101
>
```



freeDarcy - mesh



streamlines

Where to find out more

- www.cmis.csiro.au/Fastflo
- www.compumod.com.au
- www.nag.co.uk
- *Fastflo* Tutorial Guide, Version 3
- *Fastflo* Fluids ToolBox

Summary

- *Fastflo* - features appropriate to porous media problems
- 4 examples
 - dam, and effect of stress on dam solution
 - calculations for buoyancy driven flows
 - calculation of phreatic surface
 - porous media flow fed by liquid distribution systems