# Using FreeFem++ to solve equation of linear elasticity

Igor Shovkun

PGE 383; Instructor: D. N. Espinoza

Department of Petroleum and Geosystems Engineering, The University of Texas at Austin

## 1 Introduction

Equilibrium equation:

$$-\nabla \cdot \sigma = f \quad \text{in } \Omega \tag{1}$$

Generic constitutive model:

$$\sigma_{ij} = C_{ijkl}\varepsilon_{ij} \tag{2}$$

Linear elasticity:

$$\sigma_{ij} = \lambda\delta_{ij}\nabla u + 2G\varepsilon_{ij} \tag{3}$$

Definition of strain:

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{4}$$

The corresponding variationnal form is:

$$\int_\Omega \sigma(u) : \varepsilon(v) dV + \int_{\partial\Omega_n} \tau \cdot n \, dS = \int_\Omega f \cdot v \; dV \tag{5}$$

where : denotes the tensor scalar product, i.e. $a : b = \sum_{i,j} a_{ij}b_{ij}$, $\partial\Omega_n$ is a Neumann boundary, $\tau$ is the boundary force vector, and $n$ is the outward normal vector (perpendicular to the infinitesimal surface $\vec{dS}$).

In the case of plane strain linear elasticity, the variational form look like this:

$$\int_{\Omega} \left[2G\varepsilon_{ij}(u)\varepsilon_{ij}(v) + \lambda\varepsilon_{ii}(u)\varepsilon_{jj}(v)\right] dV + \int_{\partial\Omega_n} \tau \cdot n \, dS = \int_{\Omega} f \cdot v \; dV \tag{6}$$

## 2 Wellbore problem in FreeFem++

One problem of interest is to find stress component around a wellbore. Mathematically, this problem implicates solving Eq. 1 in an infinite domain with a pressurized circular opening with radius $R$ and can be written as follows:

$$\begin{cases} -\nabla\sigma = f \\ \sigma_{xx}(x = \pm\infty, y) = S_1 \\ \sigma_{yy}(x, y = \pm\infty) = S_2 \end{cases} \tag{7}$$

The analytical solution for this problem reads:

$$\begin{aligned} \sigma_{rr} =& \frac{1}{2}(S_1 + S_2 - 2P_w)\left[1 - \left(\frac{R}{r}\right)^2\right] + \frac{1}{2}(S_1 - S_2)\left[1 - \left(\frac{2R}{r}\right)^4 + 3\left(\frac{R}{r}\right)^4\right]\cos(2\theta) + P_w\left(\frac{R}{r}\right)^2 \\ \sigma_{\theta\theta} =& \frac{1}{2}(S_1 + S_2 - 2P_w)\left[1 + \left(\frac{R}{r}\right)^2\right] - \frac{1}{2}(S_1 - S_2)\left[1 + 3\left(\frac{R}{r}\right)^4\right]\cos(2\theta) - P_w\left(\frac{R}{r}\right)^2 \\ \sigma_{r\theta} =& \frac{1}{2}(S_1 - S_2)\left[1 + \left(\frac{2R}{r}\right)^4 - 3\left(\frac{R}{r}\right)^4\right]\sin(2\theta) \end{aligned} \tag{8}$$

Define problem parameters:

```
// Dimensions
real ySize = 10.;  // y-size of the domain
real xSize = 10.;  // x-size of the domain
real R = 0.1; // wellbore radius
//Elastic constants
real E = 1e10 ; // young's modulus
real nu = 0.3;  // poisson's ratio
```

Then we calculate Lame constant and shear modulus that are used in the formulation:

```
real G = E/(2*(1+nu)); // shear modulus
```

```
real lambda = E*nu/((1+nu)*(1-2*nu)); // Lame constant
```

Finally, we specify values for stress boundary conditions:

```
// Stresses
real Sx = 10e6;
real Sy = 10e6;
real Pwell = 0.0;
```

Next we proceed to the definition of the domain. We first define all the domain boundaries and then create mesh:

```
// First define boundaries
border Right(t=-ySize/2,ySize/2) {x = xSize/2; y = t;}
border Top(t=xSize/2,-xSize/2) {x = t; y = ySize/2;}
border Left(t=ySize/2,-ySize/2) {x = -xSize/2; y = t;}
border Bottom(t=-xSize/2,xSize/2) {x = t; y = -ySize/2;}
border Well(t = 0, -2*pi) {x = R*cos(t); y = R*sin(t);}
// Create mesh
int n = 20; // number of mesh nodes on the outer borders
int nwell = 50; // number of mesh nodes on wellobre
mesh Omega = buildmesh(Right(n)+Top(n)+Left(n)+Bottom(n)+Well(nwell));
```

The next step is to define Finite Element spaces. These object designate the type of shape functions used to solve the problem. In this example we will be using "P1" elements for displacement: continuous Galerkin polynomials of the first degree defined on triangles. To approximate stresses we will use "P0" elements - piecewise constants.

```
// FE spaces
fespace Displacement(Omega, P1); // linear shape functions
fespace Stress(Omega, P0); // piecewise constants
```

Next, we will tell the interpreter to allocate several objects to store displacement components (objects of type Displacement) and stresse components (objects of type Stress). $u1$ and $u2$ denote

x and y components of the displacement vector. $v1$ and $v2$ are the components of the virtual displacement vector; we do not need those vectors per se, but we use these objects to code up the system of equations. We also define objects *sigmaxx*, *sigmaxy*, and *sigmayy* to store stress components. Note that we only need 3 components of the stress tensor because it is symmetric.

```
Displacement u1, u2, v1, v2;
Stress sigmaxx, sigmayy, sigmaxy;
```

Next we define macros. Macros are special compiler instructions that are inserted into the part of code where they are called during compilation (as opposed to runtime). In this code we use macros in order to make the system of equation look neat later on. The following code defines two macros for strain and stress "vectors".

```
// definition of 2 macros:
// macro for strain
macro e(u1,u2)
  [
    dx(u1),
    (dy(u1)+dx(u2))/2,
    (dx(u2)+dy(u1))/2,
    dy(u2)
  ]
  // eps_xx, eps_xy, eps_yx, eps_yy

// macro for stress
macro sigma(u1,u2)
  [
    (lambda+2.*G)*e(u1,u2)[0] + lambda*e(u1,u2)[3],
    2.*G*e(u1,u2)[1],
    2.*G*e(u1,u2)[2],
    lambda*e(u1,u2)[0] + (lambda+2.*G)*e(u1,u2)[3]
  ] // stress sxx,sxy,syx,syy
```

Now, we can finally define the system to be solved corresponding to the Formulation (ref).

```
// Define system of equations
problem Elasticity([u1,u2], [v1,v2]) =
    int2d(Omega) (   sigma(u1,u2)'*e(v1,v2)   )
  // Boundary conditions
  + int1d(Omega, Right) (Sx*v1)
  - int1d(Omega, Left) (Sx*v1)
  + int1d(Omega, Top) (Sy*v2)
  - int1d(Omega, Bottom) (Sy*v2)
  + int1d(Omega, Well) (Pwell*(N.x*v1 + N.y*v2))
  ;
```

In order to solve the problem, we simply need to call this subroutine:

```
// Solve system
Elasticity;
```

Finally, in order to calculate stresses, we use previously define macros again:

```
// Stresses
sigmaxx = sigma(u1, u2)[0];
sigmayy = sigma(u1, u2)[3];
sigmaxy = sigma(u1, u2)[1]; // we could use [2] as well
```

## 3 Stresses around a fracture

The changes in the code are minor. Instead of a wellbore we need to define a fracture:

```
real xf = 40; // fracture half-length
real fw = 0.1; // fracture half-width
```

```
border Frac(t = 0, -2*pi) {x = fw*cos(t); y = xf*sin(t);}
```

The bondary condition on the fracture boundary are also a little bit different:

```
// Define system of equations
problem Elasticity([u1,u2], [v1,v2]) =
    int2d(Omega) (   sigma(u1,u2)'*e(v1,v2)   )
  // Boundary conditions
  + int1d(Omega, Right) (Sx*v1)
  - int1d(Omega, Left) (Sx*v1)
  + int1d(Omega, Top) (Sy*v2)
  - int1d(Omega, Bottom) (Sy*v2)
  // condition only on one component
  + int1d(Omega, Frac) (Pfrac*(N.x*v1))
  ;
```