A nonlinear example with FreeFem++

Marco Caliari

June 9, 2014

1 AR equation

Let us consider the advection–reaction nonlinear problem

$$-\mu\Delta u(x,y)+\rho u^2(x,y)=1,\quad (x,y)\in[0,1]^2$$

with homogeneous Dirichlet boundary conditions. The weak formulation is

$$\text{find } u\in H_0^1 \text{ such that } F(u)=\mu\int\nabla u\cdot\nabla v+\rho\int u^2v-\int v=0\quad\forall v\in H_0^1$$

If we choose v as a basis function φ_i, we can consider

$$F_i(u)=\mu\int\nabla u\cdot\nabla\varphi_i+\rho\int u^2v-\int\varphi_i$$

Therefore, we have to compute the zero of F. Let us compute its Jacobian applied to $\delta\in H_0^1$

$$J_F(u)\delta=\mu\int\nabla\delta\cdot\nabla\varphi_i+\rho\int 2u\delta\varphi_i$$

Newton's method writes now

- set $r=0$ and take an initial guess u^r
- solve the linear system $J_F(u^r)\delta^{r+1}=-F(u^r)$ to get δ^{r+1}
- WHILE $\|\delta^{r+1}\|>\text{tol}$

  $u^{r+1}=u^r+\delta^{r+1}$

  $r=r+1$

  solve the linear system $J_F(u^r)\delta^{r+1}=-F(u^{r+1})$ to get δ^{r+1}

END

Given u^r, $J_F(u^r)$ is a symmetric bilinear form $a(\delta^{r+1}, v)$ and $F(u^r)$ a linear functional $\ell(v)$. Therefore, it is possible to solve the linear system in Newton's method with

```
solve AR(delta,v) = int2d(Th)(mu*(dx(delta)*dx(v)+dy(delta)*dy(v))
                              +2*rho*(delta*u*v))
  +int2d(Th)(mu*(dx(u)*dx(v)+dy(u)*dy(v))+rho*(u^2*v)-v)
  +on(1,2,3,4,delta=0);
```

It is also possible to costruct explicitely the matrix and the right hand side of the linear system to solve.

```
varf Fvar(unused,v) = int2d(Th)(-mu*(dx(u)*dx(v)+dy(u)*dy(v))
                                -rho*(u^2*v)+v)
  +on(1,2,3,4,unused=0); // notice the sign
ret = Fvar(0,Xh);
varf JFvar(delta,v) = int2d(Th)(mu*(dx(delta)*dx(v)+dy(delta)*dy(v))
                                +2*rho*(delta*u*v))
  +on(1,2,3,4,delta=0);
matrix A = JFvar(Xh,Xh); // here you can specify the solver
delta[] = A^(-1)*ret;
```

It is also possible to use explicitely an iterative method in FreeFem++. Let us define the functions

```
func real[int] F(real[int] & u)
{
  // F(u)\phi_i
  Xh uloc;
  uloc[] = u;
  varf Fvar(unused,v) = int2d(Th)(-mu*(dx(uloc)*dx(v)+dy(uloc)*dy(v))
  -rho*(uloc^2*v)+v)
    +on(1,2,3,4,unused=0);
  ret = Fvar(0,Xh);
  return ret;
}
```

and

```
func real[int] JF(real[int] & delta)
{
  // JF(u)\phi_i
```

```
  Xh deltaloc;
  deltaloc[] = delta;
  varf JFvar(unused,v) = int2d(Th)(mu*(dx(deltaloc)*dx(v)+dy(deltaloc)*dy(v))
   +2*rho*(deltaloc*u*v))
    +on(1,2,3,4,unused=0);
  ret = JFvar(0,Xh);
  return ret;
}
```

and call the solver

```
rhs = F(u[]);
LinearCG(JF,deltav,rhs,eps=-1.e-30); // ||rhs-JF*deltav||_2^2<-eps
```

1.1 Differential preconditioning

Given an initial guess u^0, it is possible to consider the following linear AD

$$-\mu\Delta u(x,y)+\rho u^0(x,y)u(x,y)=1, \quad (x,y)\in[0,1]^2$$

The Jacobian (applied to δ) associated to the weak form is

$$J_{F^0}(u)\delta=\mu\int\nabla\delta\cdot\nabla\varphi_i+\rho\int u^0\delta\varphi_i$$

The matrix with elements $J_{F^0}(u)\varphi_j$ does not depend on u (only on u^0) and can be used as a differential preconditioner in Newton's iterations. It can be computed in FreeFem++ as

```
varf AR0(delta,v) = int2d(Th)(mu*(dx(delta)*dx(v)+dy(delta)*dy(v))
      +rho*(delta*u0*v))
  +on(1,2,3,4,delta=0);
matrix AR0mat = AR0(Xh,Xh,solver=Cholesky,factorize=1);
```

The preconditioner is implemented in a function

```
func real [int] P(real[int] & w)
{
  ret = AR0mat^-1*w;
  return ret;
}
```

and the Conjugate Gradient method is called by

```
LinearCG(JF,deltav,rhs,precon=P);
```