Reminders for NAPDE

Andrea Bonifacio

June 10, 2023

Reminders on calculus

$$\int_\Omega -\Delta u v = \int_\Omega \nabla u \cdot \nabla v - \underbrace{\int_{\Gamma_D} \nabla u \cdot \mathbf{n} v}_{=0 \text{ if } v|_{\Gamma_D}=0}$$

$$\int_\Omega \operatorname{div} u = \int_{\partial\Omega} u \cdot \mathbf{n}$$

Weak Formulations

Elliptic equations

$$\begin{cases} -\operatorname{div}(\mu \nabla u) + \mathbf{b} \cdot \nabla u + \sigma u = f & \text{in } \Omega \quad g \in L^2(\Gamma_N) \\ u = 0 & \text{on } \Gamma_D \quad \partial\Omega = \Gamma_D \cup \Gamma_N \\ \mu \nabla u \cdot \mathbf{n} = g & \text{on } \Gamma_N \quad {\Gamma_D}^o \cap {\Gamma_N}^o = \varnothing \end{cases}$$

$$\Downarrow$$

$$\underbrace{\int_\Omega \mu \nabla u \cdot \nabla v + \int_\Omega \mathbf{b} \cdot \nabla u v + \int_\Omega \sigma u v}_{=:a(u,v)} = \int_\Omega f v + \underbrace{\int_{\Gamma_D} \mu \nabla u \cdot \mathbf{n} v}_{=0 \text{ if } v|_{\Gamma_D}=0} + \int_{\Gamma_N} \underbrace{\mu \nabla u \cdot \mathbf{n}}_{=g} v$$

$$\Downarrow$$

$$\begin{cases} \text{find } u \in V & V = \left\{ v \in H^1(\Omega), v|_{\Gamma_D} = 0 \right\} =: H^1_{\Gamma_D}(\Omega) \\ a(u,v) = \langle F, v \rangle & \forall\, v \in V \end{cases}$$

Parabolic equations

$$\begin{cases} \frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} = f & 0 < x < d, t > 0 \\ u(x,0) = u_0(x) & 0 < x < d \\ u(0,t) = u(d,t) = 0 & t > 0 \end{cases}$$

$$\Downarrow$$

$$\int_\Omega \frac{\partial u(t)}{\partial t} v \, d\Omega + a(u(t), v) = \int_\Omega f(t) v \, d\Omega \quad \forall\, v \in V$$

$$\Downarrow$$

for each $t > 0$, we need to find $u_h(t) \in V_h$ s.t.

$$\int_\Omega \frac{\partial u_h(t)}{\partial t} v_h \, d\Omega + a(u_h(t), v_h) = \int_\Omega f(t) v_h \, d\Omega \quad \forall\, v_h \in V_h$$

Code implementation

CG-FEM

- Matrix A;
$$A_{ij} = \int_\Omega \nabla\varphi_j \nabla\varphi_i$$
Loop on all the elements and compute locally (elements with $\hat{}$ are computed on the reference element):
$$A_{loc_{ij}} = \det(\mathbf{B}_\mathcal{K}) \int_{\hat{\mathcal{K}}} \hat{\nabla}\hat{\varphi}_j^T \mathbf{B}_\mathcal{K}^{-1} \mathbf{B}_\mathcal{K}^{-1} \hat{\nabla}\hat{\varphi}_i = \frac{\det(\mathbf{B})}{2} \hat{\nabla}\hat{\varphi}_j^T \mathbf{B}_\mathcal{K}^{-1} \mathbf{B}_\mathcal{K}^{-T} \hat{\nabla}\hat{\varphi}_i$$
Can be implemented as

```
function [K_loc]=C_lap_loc(Grad,w_2D,nln,BJ)
K_loc=zeros(nln,nln);
for i=1:nln
    for j=1:nln
        for k=1:length(w_2D)
            Binv = inv(BJ(:,:,k));   % inverse
            Jdet = det(BJ(:,:,k));   % determinant
            K_loc(i,j) = K_loc(i,j) + (Jdet.*w_2D(k)) .* ( (Grad(k,:,i)
                             * Binv) * (Grad(k,:,j) * Binv )');
        end
    end
end
```

- Mass matrix M:
$$M_{ij} = \int_\Omega \varphi_j, \varphi_i$$
Loop on all the elements and calculate the local mass matrix
$$M_{loc_{ij}} = \det(\mathbf{B}_\mathcal{K}) \int_{\hat{\mathcal{K}}} \hat{\varphi}_j^T \mathbf{B}_\mathcal{K}^{-1} \mathbf{B}_\mathcal{K}^{-1} \hat{\varphi}_i$$
Can be implemented as

```
function [M_loc]=C_mass_loc(dphiq,w_2D,nln,BJ)
M_loc=zeros(nln,nln);
for i=1:nln
    for j=1:nln
        for k=1:length(w_2D)
            Binv = inv(BJ(:,:,k));      % inverse
            Jdet = det(BJ(:,:,k));      % determinant
            M_loc(i,j) = M_loc(i,j) + (Jdet.*w_2D(k))
                             .* dphiq(1,k,i).* dphiq(1,k,j);
        end
    end
end
```

- Transport matrix T

  Can be implemented as

```
function [ADV_loc]=C_adv_loc(Grad,dphiq,beta,w_2D,nln,BJ)
ADV_loc=sparse(nln,nln);
for i=1:nln
    for j=1:nln
        for k=1:length(w_2D)
            Binv=inv(BJ(:,:,k));    % inverse
            Jdet=det(BJ(:,:,k));    % determinant
            ADV_loc(i,j) = ADV_loc(i,j)+(Jdet.*w_2D(k)).* dphiq(1,k,i)
                           *( (beta)*(Grad(k,:,j) * Binv )');
        end
    end
end
```

- Right hand side **b**:

$$b_i = \int_\Omega f\varphi_i$$

which is computed

```
function [f]=C_loc_rhs2D(force,dphiq,BJ,w_2D,pphys_2D,nln,mu)
f = zeros(nln,1);
x = pphys_2D(:,1);
y = pphys_2D(:,2);
F = eval(force);
for s = 1:nln
    for k = 1:length(w_2D)
        Jdet = det(BJ(:,:,k));  % determinant
        f(s) = f(s) + w_2D(k)*Jdet*F(k)*dphiq(1,k,s);
    end
end
```