

3.7 WG FEM $(P_0, P_0; AC_0)$ for Darcy Equation in a General Form

Darcy eqn. general form is considered here:

$$\begin{cases} \nabla \cdot (-\mathbf{K}(\nabla p - \mathbf{f})) \equiv \nabla \cdot \mathbf{u} = s & \text{in } \Omega, \\ p = p_D & \text{on } \Gamma_D, \\ \mathbf{u} \cdot \mathbf{n} = u_N & \text{on } \Gamma_N, \end{cases} \quad (3.24)$$

where \mathbf{f} is a known additional term, e.g., gravity. For convenience, we may write $\mathbf{g} = \mathbf{Kf}$ sometimes.

The **variational form** for (3.24) is

$$\int_{\Omega} (\mathbf{K} \nabla p) \cdot \nabla q = \int_{\Omega} s q - \int_{\Gamma_N} u_N q + \int_{\Omega} (\mathbf{Kf}) \cdot \nabla q. \quad (3.25)$$

WG FE Scheme. We assume Ω is a 2-dim polygonal domain equipped with a quasi-uniform convex quadrilateral mesh \mathcal{E}_h . We use $\text{WG}(P_0, P_0; AC_0)$ finite elements.

We define WG finite element shape functions S_h as ... The WG finite element scheme is

$$\mathcal{A}_h(p_h, q_h) = \mathcal{F}_h(q_h), \quad \forall q_h \in S_h, \quad (3.26)$$

where the bilinear form and linear form are

$$\mathcal{A}_h(p_h, q_h) = \sum_{E \in \mathcal{E}_h} \int_E (\mathbf{K} \nabla_w p_h) \cdot \nabla_w q_h. \quad (3.27)$$

$$\mathcal{F}_h(q_h) = \sum_{E \in \mathcal{E}_h} \int_{E^\circ} s q_h^\circ - \sum_{e \in \Gamma_{N,h}} \int_e u_N q_h^\partial + \sum_{E \in \mathcal{E}_h} \int_E \mathbf{Q}_h(\mathbf{Kf}) \cdot \nabla_w q_h. \quad (3.28)$$

Here \mathbf{Q}_h is the local projection from $L^2(\Omega)^2$ to the broken space $AC_0(\mathcal{E}_h)$.

Post-processing for velocity, flux. After the numerical pressure p_h is solved from (3.26), we compute the elementwise numerical Darcy velocity

$$\mathbf{u}_h = \mathbf{Q}_h(-\mathbf{K}(\nabla_w p_h - \mathbf{f})) = \mathbf{Q}_h(-\mathbf{K}(\nabla_w p_h)) + \mathbf{Q}_h(\mathbf{Kf}). \quad (3.29)$$

Note that $\mathbf{Q}_h(\mathbf{Kf})$ is needed for both assembly of the global RHS and post-processing for the numerical velocity. Thus pre-computation is helpful.

The elementwise edgewise bulk fluxes are calculated *ad hoc* as follows

$$\int_e \mathbf{u}_h \cdot \mathbf{n}_e, \quad \forall e \subset E^\partial \quad \forall E \in \mathcal{E}_h, \quad (3.30)$$

where \mathbf{n}_e is the outward unit normal vector to E^∂ . Actually no quadrature is needed. The mid-point formula suffice, since \mathbf{u}_h is linear on each edge (proof shall be provided later).

Some details about implementation. Note that we are dealing the numerical velocity in the local AC_0 spaces. Note also that

$$\nabla_w p_h = \nabla_w \left(\sum_{i=0}^4 c_i \phi_i \right) = \sum_{i=0}^4 c_i \nabla_w \phi_i = \sum_{i=0}^4 c_i \left(\sum_{j=1}^4 d_{ij} \mathbf{w}_j \right) = \sum_{j=1}^4 \left(\sum_{i=0}^4 c_i d_{ij} \right) \mathbf{w}_j. \quad (3.31)$$

Accordingly,

$$\mathbf{Q}_h(-\mathbf{K} \nabla_w p_h) = \sum_{j=1}^4 \sum_{i=0}^4 c_i d_{ij} \mathbf{Q}_h(-\mathbf{K} \mathbf{w}_j) = \sum_{j=1}^4 \sum_{i=0}^4 c_i d_{ij} \sum_{k=1}^4 \alpha_{jk} \mathbf{w}_k. \quad (3.32)$$

This involves three main formulas.

- (i) Elementwise $p_h|_E$ is a linear combination of 5 basis functions (ϕ_0 for element interior, $\phi_1, \phi_2, \phi_3, \phi_4$ for the 4 edges)

$$p_h|_E = \sum_{i=0}^4 c_i \phi_i. \quad (3.33)$$

The elementwise 5 coefficients $c_i (0 \leq i \leq 5)$ are extracted from the numerical solution for pressure.

- (ii) Elementwise, for the 5 basis functions, the discrete weak gradients are established as linear combinations $\mathbf{w}_j (j = 1, 2, 3, 4)$ as follows

$$\nabla_w \phi_i = \sum_{j=1}^4 d_{ij} \mathbf{w}_j, \quad 0 \leq i \leq 4. \quad (3.34)$$

- (iii) Finally, we deal with the local L_2 -projection:

$$\mathbf{Q}_h(-\mathbf{K} \mathbf{w}_j) = \sum_{k=1}^4 \alpha_{jk} \mathbf{w}_k. \quad (3.35)$$

See Liu, Tavener, Wang, *SIAM J.Sci.Comput.* (2018) p.B1247 (top) on how to do find the projection coefficients.

Implementionwise, for (i), (ii), and (iii), we deal with three arrays `cof`, `CDWGB`, and `ProjCof` and their composition.

Bilinear mapping and Piola matrix. See Liu, Tavener, Wang, *J. Comput. Phys.* (2016) also. Let $E = P_1P_2P_3P_4$ be a convex quadrilateral with four vertices $P_i = (x_i, y_i)$, $i = 1, 2, 3, 4$ that are oriented counterclockwise. The bilinear mapping from the reference element $\hat{E} = [0, 1]^2$ to the quadrilateral E can be expressed as

$$\begin{cases} x = a_1 + a_2\hat{x} + a_3\hat{y} + a_4\hat{x}\hat{y}, \\ y = b_1 + b_2\hat{x} + b_3\hat{y} + b_4\hat{x}\hat{y}, \end{cases} \quad (3.36)$$

where $\hat{\mathbf{x}} = (\hat{x}, \hat{y}) \in \hat{E}$ and $\mathbf{x} = (x, y) \in E$. The eight coefficients $a_i, b_i (i = 1, 2, 3, 4)$ can be directly calculated using the vertex coordinates. The Jacobian matrix of the bilinear mapping is

$$\mathbf{J} = \begin{bmatrix} a_2 + a_4\hat{y} & a_3 + a_4\hat{x} \\ b_2 + b_4\hat{y} & b_3 + b_4\hat{x} \end{bmatrix}. \quad (3.37)$$

The Jacobian determinant is a linear polynomial in \hat{x}, \hat{y} :

$$J = \det(\mathbf{J}) = \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} + \begin{vmatrix} a_2 & a_4 \\ b_2 & b_4 \end{vmatrix} \hat{x} + \begin{vmatrix} a_4 & a_3 \\ b_4 & b_3 \end{vmatrix} \hat{y}. \quad (3.38)$$

The Piola matrix is simply the Jacobian matrix divided by its determinant.

$$\mathbf{P}_E = \mathbf{P}(\hat{\mathbf{x}}) = \frac{\mathbf{J}(\hat{\mathbf{x}})}{J(\hat{\mathbf{x}})}. \quad (3.39)$$

The Piola transformation maps a vector field defined on \hat{E} to a vector field defined on the quadrilateral E (Matrix-vector multiplication):

$$\mathbf{v}(\mathbf{x}) = \mathbf{P}_E \hat{\mathbf{v}}(\hat{\mathbf{x}}). \quad (3.40)$$

Local, broken, and global AC spaces.

Numerical velocity in local and global AC spaces. For each quadrilateral element E , $\mathbf{u}_h|_E$ is established in the local $AC_0(E)$ space. But the bulk flux normal continuity implies that the numerical velocity, after being glued together over the whole mesh, is also in the global AC space $AC_0(\mathcal{E}_h)$.

By post-processing, we obtain the elementwise 4 coefficients for expressing the numerical velocity as a linear combination of the local basis (using the normalized coordinates and Piola transformation)

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{w}_3 = \begin{bmatrix} X \\ Y \end{bmatrix}, \quad \mathbf{w}_4 = \mathbf{P}_E \begin{bmatrix} \hat{x} \\ -\hat{y} \end{bmatrix}, \quad (3.41)$$

where \mathbf{P}_E is the Piola matrix, (\hat{x}, \hat{y}) reference coordinates, (x_c, y_c) element center, $X = x - x_c, Y = y - y_c$.

We shall consider another basis (**FluxBas**), $\{\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4\}$, based on the edge bulk normal flux, as preparation for transition to a global basis. There is no doubt that

$$\mathbf{f}_j = \sum_{k=1}^4 c_{jk} \mathbf{w}_k, \quad j = 1, 2, 3, 4. \quad (3.42)$$

It is expected that for $1 \leq i, j \leq 4$,

$$\delta_{ij} = \int_{e_i} \mathbf{f}_j = \int_{e_i} \sum_{k=1}^4 c_{jk} \mathbf{w}_k = \sum_{k=1}^4 c_{jk} \int_{e_i} \mathbf{w}_k = \sum_{k=1}^4 b_{ik} c_{jk}.$$

Here $b_{ik} = \int_{e_i} \mathbf{w}_k$ is recorded for convenience. This implies that elementwise

$$C = (B^{-1})^T, \quad C = [c_{jk}]_{4 \times 4}, \quad B = [b_{ik}]_{4 \times 4}. \quad (3.43)$$

In summary, for each quadrilateral element,

- We compute edgewise 4 bulk normal fluxes for the normalized and Piola-transformed basis functions in (3.41);
- Form the above results as a 4×4 matrix;
- **BIT**: Matrix B inverted and transposed, and then it is called C ;
- The entries of C will be used for linear combination of the normalized basis functions to form the new edgewise flux-oriented basis functions.

Over the whole quadrilateral mesh, we glue together these flux-based new basis functions to get a (global) basis for the global AC_0 space with some adjustments regarding normal vector directions. This is done so because a function in the global AC_0 space is completely determined by the normal fluxes across the edges.

Example. An example from Girault, Vassilev, Yotov, *Numer. Math.* (2014) has been tested.

Code packet. Here is a self-contained Matlab code packet:

DarcyMINI_DarcyGen_WG_QuadriPkPkACK.zip