

**DarcyLite:  
A Matlab Toolbox for Numerical  
Simulations of Flow and Transport in  
Porous Media**

Jiangguo (James) Liu  
Graham Harper  
Farrah Sadre-Marandi  
Zhuoran (Sophy) Wang  
Department of Mathematics  
Colorado State University  
Fort Collins, CO 80523-1874, USA  
`{liu,harper,wangz}@math.colostate.edu`,  
`sadre.1@mbi.osu.edu`

October 8, 2016

# Contents

<b>1</b>	<b>Flow and Transport in Porous Media</b>	<b>4</b>
1.1	The Darcy Equation . . . . .	4
1.2	Transport Equations . . . . .	5
1.3	Steady-state Equations . . . . .	5
1.4	Two-phase Flow . . . . .	5
1.5	Poroelasticity . . . . .	6
1.6	Other Equations . . . . .	6
1.6.1	Parabolic Equations . . . . .	6
1.6.2	Stokes Equations . . . . .	6
1.6.3	Linear Elasticity . . . . .	6
1.6.4	Fluid Structure Interaction (FSI) . . . . .	7
<b>2</b>	<b>Preliminaries</b>	<b>8</b>
2.1	Some Nice Formulas about Triangles . . . . .	8
2.2	Continuous Finite Elements . . . . .	10
2.3	Discontinuous Finite Elements . . . . .	10
2.4	$H(\text{div})$ Finite Elements . . . . .	10
2.4.1	Edge-based Bases for $RT_0$ and $BDM_1$ . . . . .	11
2.4.2	Normalized Bases for $RT_0, BDM_1, RT_1$ . . . . .	13
2.4.3	Conversion Among the Intrinsic and Natural Bases of $RT_0$ . . . . .	13
2.5	Weak Galerkin Finite Elements . . . . .	14
2.5.1	WG Elements $(P_0, P_0, RT_0)$ on Triangles . . . . .	14
2.5.2	WG Elements $(Q_0, P_0, RT_{[0]})$ on Rectangles . . . . .	16
2.5.3	Higher Order WG Elements . . . . .	17
2.5.4	WG Finite Elements on Polygons . . . . .	18
2.5.5	WG Finite Elements on Quadrilaterals . . . . .	19
2.6	Quadratures . . . . .	20
<b>3</b>	<b>Finite Element Methods for the Darcy Equation</b>	<b>21</b>
3.1	Continuous Galerkin (CG) FEMs . . . . .	21
3.1.1	CG $P_1$ on Triangular Meshes . . . . .	22
3.1.2	CG $Q_1$ on Rectangular Meshes . . . . .	22

3.2	Discontinuous Galerkin (DG) FEMs . . . . .	23
3.2.1	DG $P_k(k = 1, 2)$ for the Darcy Equation . . . . .	25
3.2.2	Postprocessing for DGP1 and DGP2 . . . . .	25
3.3	Enhanced Galerkin (EG) FEMs . . . . .	25
3.4	Weak Galerkin (WG) FEMs . . . . .	25
3.4.1	WG $(P_0, P_0, RT_0)$ on Triangles for the Darcy Equation . . . . .	26
3.4.2	WG $(Q_0, P_0, RT_{[0]})$ on Rectangles for the Darcy Equation . . . . .	26
3.4.3	WG $(Q_0, P_0, RT_{[0]})$ on Quadrilaterals for the Darcy Equation . . . . .	26
3.4.4	WG $(P_1, P_1, P_0^2)$ on Polygons for the Darcy Equation . . . . .	27
3.5	Mixed Finite Element Methods (MFEMs) for the Darcy Equation . . . . .	28
3.5.1	MFEM $(RT_0, P_0)$ on a Triangular Mesh . . . . .	28
3.5.2	MFEM $(RT_{[0]}, Q_0)$ on a Rectangular Mesh . . . . .	31
3.5.3	On the Equivalence between WGFEMs and MFEMs . . . . .	31
3.6	Finite Volume Methods for the Darcy Equation . . . . .	32
<b>4</b>	<b>FEMs for Transient Convection-diffusion-reaction Equations</b>	<b>34</b>
4.1	A Transport Solver Based on Implicit Euler and Weak Galerkin . . . . .	34
<b>5</b>	<b>FEMs for Steady-state Convection-diffusion-reaction Equations</b>	<b>36</b>
5.1	WG for Steady-state CDR Equations . . . . .	36
5.2	FVM for Steady-state CDR Equations . . . . .	37
<b>6</b>	<b>FEMs for Other Types of Equations</b>	<b>38</b>
6.1	FEMs for Parabolic Equations . . . . .	38
6.1.1	CGFEMs for Parabolic Equations . . . . .	38
6.1.2	WGFEMs for Parabolic Equations . . . . .	39
6.2	FEMs for Linear Elasticity . . . . .	39
6.2.1	Classical FEMs for 2-dim Linear Elasticity . . . . .	39
6.2.2	WGFEMs for 2-dim Linear Elasticity . . . . .	39
6.3	FEMs for Linear Poroelasticity (PE) . . . . .	40
6.3.1	WGFEMs for Poroelasticity . . . . .	40
<b>7</b>	<b>Testcases</b>	<b>41</b>
7.1	Testcases for 2-dim Darcy Equation . . . . .	41
7.2	Testcases for 2-dim Transport Problems . . . . .	41
7.3	Testcases for 2-dim Elasticity . . . . .	41
7.3.1	Examples with Known Analytical Solutions . . . . .	41
7.3.2	Cook's Membrane and Cantilever Beams . . . . .	41
7.4	Testcases for 2-dim Poroelasticity . . . . .	42
7.4.1	Mandel's Problem . . . . .	42
7.4.2	Terzaghi's Problem . . . . .	45
7.4.3	Barry and Mercer Point-Source Problem . . . . .	45
7.4.4	Some Benchmarks Used in COSMOL . . . . .	45

<b>8</b>	<b>Design and Implementation Strategies of DarcyLite</b>	<b>46</b>
8.1	Assumptions . . . . .	46
8.2	Mesh Data Structure . . . . .	47
8.3	Implementation of Quadratures . . . . .	48
8.4	Element-level Full Matrices and Mesh-level Sparse Matrices . . . . .	48
8.5	Enforcing Essential Boundary Conditions . . . . .	49
8.6	Interface with Other Software Packages . . . . .	50
8.7	Graphical User Interface (GUI) of DarcyLite . . . . .	51
<b>9</b>	<b>Using DarcyLite</b>	<b>53</b>
9.1	A Quick Start . . . . .	53
9.2	More Details . . . . .	53
<b>10</b>	<b>Extra Stuff To Be Reorganized</b>	<b>55</b>
10.1	Solving Darcy Equation by the Lowest Order WG Elements . . . . .	55
10.2	Working with Quadrilateral Meshes . . . . .	55
10.2.1	Generation of Quadrilateral Meshes . . . . .	55
10.2.2	Solving the Darcy Equation on Quadrilateral Meshes . . . . .	62

# Chapter 1

## Flow and Transport in Porous Media

### 1.1 The Darcy Equation

The Darcy's law is a fundamental equation for modeling flow in porous media. It is usually further coupled with transport equations. Two examples amongst the vast applications in this regard are oil recovery in petroleum reservoirs [10, 18, 29] and drug delivery to tumors.

We consider 2-dim elliptic boundary value problems (Darcy problems) formulated as

$$\begin{cases} \nabla \cdot (-\mathbf{K}\nabla p) \equiv \nabla \cdot \mathbf{u} = f, & \mathbf{x} \in \Omega, \\ p = p_D, & \mathbf{x} \in \Gamma^D, \quad \mathbf{u} \cdot \mathbf{n} = u_N, & \mathbf{x} \in \Gamma^N, \end{cases} \quad (1.1)$$

where  $\Omega \subset \mathbb{R}^2$  is a bounded polygonal domain,  $p$  the primal unknown (pressure),  $\mathbf{K}$  a permeability tensor that is uniformly symmetric positive-definite,  $f$  a source term,  $p_D, u_N$  are respectively Dirichlet and Neumann boundary data,  $\mathbf{n}$  the unit outward normal vector on  $\partial\Omega$ , which has a nonoverlapping decomposition  $\Gamma^D \cup \Gamma^N$ .

Define a subspace and manifold for scalar-valued functions as follows

$$H_{D,0}(\Omega) = \{p \in H^1(\Omega) : p|_{\Gamma^D} = 0\}, \quad H_{D,p_D}(\Omega) = \{p \in H^1(\Omega) : p|_{\Gamma^D} = p_D\}.$$

The variational formulation for the primal variable pressure reads as: Seek  $p \in H_{D,p_D}^1(\Omega)$  such that

$$\int_{\Omega} \mathbf{K}\nabla p \cdot \nabla q = \int_{\Omega} f q - \int_{\Gamma_N} u_N q \quad \forall q \in H_{D,0}^1(\Omega). \quad (1.2)$$

The Darcy equation can also be rewritten as a system of two first-order equations by considering the primal variable (pressure) and flux (velocity  $\mathbf{u} = -\mathbf{K}\nabla p$ ) as follows

$$\mathbf{K}^{-1}\mathbf{u} + \nabla p = \mathbf{0}, \quad \nabla \cdot \mathbf{u} = f. \quad (1.3)$$

Define a subspace and manifold for vector-valued functions as follows

$$\begin{aligned} H_{N,0}(\operatorname{div}, \Omega) &= \{\mathbf{v} \in L^2(\Omega)^2 : \operatorname{div} \mathbf{v} \in L^2(\Omega), \mathbf{v}|_{\Gamma^N} = \mathbf{0}\}, \\ H_{N,u_N}(\operatorname{div}, \Omega) &= \{\mathbf{v} \in L^2(\Omega)^2 : \operatorname{div} \mathbf{v} \in L^2(\Omega), \mathbf{v}|_{\Gamma^N} \cdot \mathbf{n} = u_N\}. \end{aligned}$$

The mixed variational formulation is then: Seek  $\mathbf{u} \in H_{N,u_N}(\operatorname{div}, \Omega)$  and  $p \in L^2(\Omega)$  such that the following hold

$$\begin{cases} \int_{\Omega} (\mathbf{K}^{-1} \mathbf{u}) \cdot \mathbf{v} - \int_{\Omega} p(\nabla \cdot \mathbf{v}) = - \int_{\Gamma_D} p_D \mathbf{v} \cdot \mathbf{n} & \forall \mathbf{v} \in H_{N,0}(\operatorname{div}, \Omega), \\ - \int_{\Omega} (\nabla \cdot \mathbf{u}) q = - \int_{\Omega} f q & \forall q \in L^2(\Omega). \end{cases} \quad (1.4)$$

## 1.2 Transport Equations

## 1.3 Steady-state Equations

## 1.4 Two-phase Flow

In this section, we focus on the flow and transport in a domain  $\Omega$  with heterogeneous permeability, governed by an immiscible two-phase system with a wetting phase and a nonwetting phase (denoted by  $w$  and  $o$  respectively), for example, water and oil. For simplicity of presentation, capillary pressure and gravity are not included in the model. The Darcy's law combined with a statement of conservation of mass are expressed as

$$\nabla \cdot \mathbf{u} = q, \quad \text{where } \mathbf{u} = -\lambda(S)k(\mathbf{x})\nabla p, \quad (1.5)$$

and

$$\frac{\partial S}{\partial t} + \nabla \cdot (f(S)\mathbf{u}) = q_w, \quad (1.6)$$

where  $\mathbf{u}$  is the Darcy velocity,  $S$  is the saturation of the wetting phase, and  $k$  is the permeability coefficient. The total mobility  $\lambda(S)$  and the flux function  $f(S)$  are respectively given by

$$\lambda(S) = \frac{k_{rw}(S)}{\mu_w} + \frac{k_{ro}(S)}{\mu_o}, \quad f(S) = \frac{k_{rw}(S)/\mu_w}{\lambda(S)}, \quad (1.7)$$

where  $k_{r\alpha}(\alpha = w, o)$  is the relative permeability of the phase  $\alpha$ .

See [?].

See [19, 29].

See [17, 24].

See [31].

## 1.5 Poroelasticity

### Linear Poroelasticity

#### Nonlinear Poroelasticity

- Permeability depends on the divergence of displacement of the solid:  $\nabla \cdot \mathbf{u}$
- Permeability depends on the stress of the solid:

## 1.6 Other Equations

### 1.6.1 Parabolic Equations

The parabolic equation usually takes the form

$$c_t = D\Delta c + f(\mathbf{x}, t), \quad \mathbf{x} \in \Omega, t \in (0, T], \quad (1.8)$$

where  $\Omega$  is a Lipschitz domain,  $(0, T]$  a time period,  $D > 0$  is a diffusion constant,  $c(\mathbf{x}, t)$  is the unknown function, e.g., concentration,  $f(\mathbf{x}, t)$  is a known source/sink term. Typical boundary and initial conditions are

$$c(\mathbf{x}, t) = 0, \quad \mathbf{x} \in \partial\Omega, t \in (0, T].$$

### 1.6.2 Stokes Equations

### 1.6.3 Linear Elasticity

There are two ways writing the linear elasticity equation. One follows the physics, the other has more mathematical flavor.

The first approach obeys the physics and formulates the linear elasticity equation as

$$-\nabla \cdot \sigma(\mathbf{u}) = \mathbf{f},$$

where

$$\sigma(\mathbf{u}) = 2\mu\varepsilon(\mathbf{u}) + \lambda\text{tr}(\varepsilon(\mathbf{u}))\mathbf{I}, \quad \varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T).$$

It is clear from Calculus that

$$\text{tr}(\varepsilon(\mathbf{u})) = \nabla \cdot \mathbf{u}.$$

Since  $\lambda, \mu$  are constants, we can write the linear elasticity equation as

$$-\mu\Delta\mathbf{u} - (\mu + \lambda)\nabla(\nabla \cdot \mathbf{u}) = \mathbf{f}. \quad (1.9)$$

Both term involve second order differential operators. The first term is obviously a Laplacian acting on a vector-valued function, although it can also be written as

$$\nabla \cdot \nabla \mathbf{u}.$$

contrary to the differential operator in the the second term

$$\nabla(\nabla \cdot \mathbf{u}).$$

### 1.6.4 Fluid Structure Interaction (FSI)

Here are two basic (important) cases:

- Coupling of the Stokes equation for fluid and the elasticity equation for solid;
- Coupling of the Navier-Stokes equation for fluid and the elasticity equation for solid.

#### **Case 1: Coupling of the Stokes and elasticity equations.**

A simple model problem for Case 1: See also `FreeFem++`. A body of viscous fluid in a cavity that has an elastic solid lid. The lid deforms due to gravity and fluid stress, which impose a Neumann boundary condition for the solid.

Another model problem for Case 1: See also `deal.II`.



# Chapter 2

## Preliminaries

### 2.1 Some Nice Formulas about Triangles

**Barycentric coordinates** are enjoyed by practitioners of FEMs. Let  $T = \Delta P_1 P_2 P_3$  be a triangle oriented counterclockwise and  $|T|$  be its area. For any point  $P(x, y)$  on the triangle, let  $|T_i| (i = 1, 2, 3)$  be the areas of the small triangles when  $P_i(x_i, y_i)$  are respectively replaced by  $P$ . Then  $\lambda_i = |T_i|/|T|, i = 1, 2, 3$  are the barycentric coordinates. Clearly  $0 \leq \lambda_i \leq 1$  and  $\lambda_1 + \lambda_2 + \lambda_3 = 1$ .

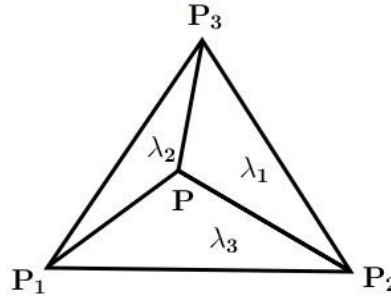


Figure 2.1: Barycentric coordinates  $\lambda_i (i = 1, 2, 3)$  for a generic triangle.

It is clear that  $\lambda_i(x, y) (i = 1, 2, 3)$  are also the Lagrangian  $P_1$  basis functions, whose gradients are

$$\begin{aligned} \nabla \lambda_1 &= \langle y_2 - y_3, x_3 - x_2 \rangle / (2|T|), \\ \nabla \lambda_2 &= \langle y_3 - y_1, x_1 - x_3 \rangle / (2|T|), \\ \nabla \lambda_3 &= \langle y_1 - y_2, x_2 - x_1 \rangle / (2|T|). \end{aligned} \tag{2.1}$$

These can also be expressed using two skew-symmetric matrices

$$\begin{bmatrix} \nabla \lambda_1 \\ \nabla \lambda_2 \\ \nabla \lambda_3 \end{bmatrix} = \frac{1}{2|T|} \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{2.2}$$

Their integrals are nicely expressed in the following lemma.

**Lemma** Let  $\alpha, \beta, \gamma$  be nonnegative integers. Then

$$\int_T \lambda_1^\alpha \lambda_2^\beta \lambda_3^\gamma dT = \frac{2|T|\alpha!\beta!\gamma!}{(\alpha + \beta + \gamma + 2)!}. \quad (2.3)$$

For convenience, we use  $(x_c, y_c)$  to denote the center of a triangle

$$x_c = \frac{1}{3}(x_1 + x_2 + x_3), \quad y_c = \frac{1}{3}(y_1 + y_2 + y_3). \quad (2.4)$$

Furthermore, denote

$$X = x - x_c, \quad Y = y - y_c. \quad (2.5)$$

Here are some interesting formulas. First,

$$\int_T x^2 = \int_T \left( \sum_{i=1}^3 \lambda_i x_i \right)^2 = [x_1 \ x_2 \ x_3] \frac{|T|}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. \quad (2.6)$$

Similarly,

$$\int_T y^2 = [y_1 \ y_2 \ y_3] \frac{|T|}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}. \quad (2.7)$$

For the mixed product, there holds

$$\int_T xy = \sum_{i=1}^3 \sum_{j=1}^3 \left( \int_T \lambda_i \lambda_j \right) x_i y_j = [x_1 \ x_2 \ x_3] \frac{|T|}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}. \quad (2.8)$$

For the normalized coordinates, it is clearly that

$$\int_T X = 0, \quad \int_T Y = 0. \quad (2.9)$$

Using the fact

$$\int_T X^2 = \int_T (x - x_c)^2 = \int_T x^2 - 2x_c \int_T (X + x_c) + \int_T x_c^2 = \int_T x^2 - x_c^2 |T|$$

and

$$x_c^2 |T| = \frac{|T|}{9} (x_1 + x_2 + x_3)^2 = [x_1 \ x_2 \ x_3] \frac{|T|}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

along with formula (?), we obtain

$$\int_T X^2 = [x_1 \ x_2 \ x_3] \frac{|T|}{36} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. \quad (2.10)$$

Similarly,

$$\int_T Y^2 = [y_1 \ y_2 \ y_3] \frac{|T|}{36} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}. \quad (2.11)$$

Using the fact

$$\int_T XY = \int_T xy - x_c y_c |T|,$$

we obtain

$$\int_T XY = [x_1 \ x_2 \ x_3] \frac{|T|}{36} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}. \quad (2.12)$$

Sometimes we prefer these formulas without using matrix and vector notations:

$$\int_T X^2 = \frac{|T|}{36} \left( (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_1)^2 \right), \quad (2.13)$$

$$\int_T Y^2 = \frac{|T|}{36} \left( (y_1 - y_2)^2 + (y_2 - y_3)^2 + (y_3 - y_1)^2 \right). \quad (2.14)$$

## 2.2 Continuous Finite Elements

Probably continuous  $P_1$  finite elements on a conforming triangular mesh (without hanging nodes) are most commonly used. It is interesting to note that in this case the Lagrange  $P_1$  basis functions on a triangle  $T$  are actually the barycentric coordinates. Therefore the Gram matrix or mass matrix is

$$GM = \frac{|T|}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}. \quad (2.15)$$

## 2.3 Discontinuous Finite Elements

## 2.4 $H(\text{div})$ Finite Elements

Raviart-Thomas (RT) elements and Brezzi-Douglas-Marini (BDM) elements are among the most frequently used [5, 6, 14, 11, 18, 32]. In this section, we briefly discuss some interesting properties of the basis functions for these elements.

### 2.4.1 Edge-based Bases for $RT_0$ and $BDM_1$

Geometric information of triangles is used to construct local and global basis functions for  $RT_0, RT_1, BDM_1$  elements.

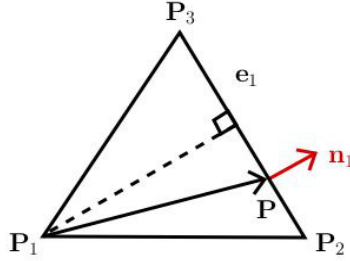


Figure 2.2:  $RT_0$  edge-based local basis functions.

**$RT_0$  edge-based basis functions** use areas and edge lengths of triangles. Let  $e_i (i = 1, 2, 3)$  be the edge opposite to vertex  $P_i$  and

$$\phi_i = \frac{|e_i|}{2|T|}(P - P_i), \quad i = 1, 2, 3. \quad (2.16)$$

One can easily prove that [5]

$$\phi_i \cdot \mathbf{n}_j = \delta_{i,j}, \quad (2.17)$$

$$\nabla \cdot \phi_i = |e_i|/|T|, \quad i = 1, 2, 3, \quad (2.18)$$

where  $\mathbf{n}_j$  is the unit outer normal on edge  $e_j$  and  $\delta_{ij}$  is the Kronecker symbol.

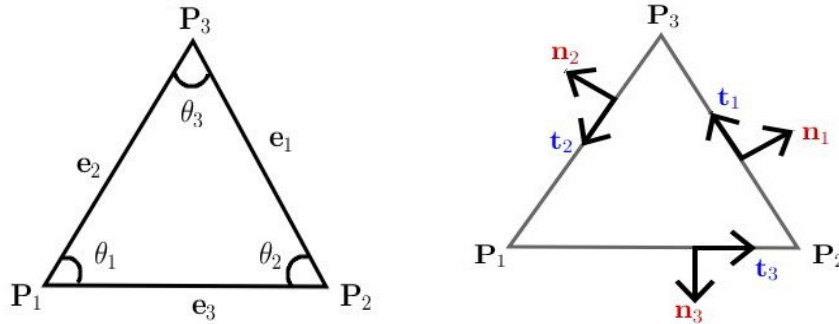


Figure 2.3: Triangle geometric information used for  $BDM_1$  edge-based local basis functions

**BDM<sub>1</sub> edge-based basis functions** use internal angles, tangential vectors, and bary coordinates of triangles. Let  $\theta_i$  be the internal angle at vertex  $P_i$  and  $\mathbf{t}_i$  be the unit tangential vector on edge  $e_i$  following the counterclockwise orientation. Clearly,  $\mathbf{n}_i$

can be obtained by a  $90^\circ$  clockwise rotation of  $\mathbf{t}_i$ . We have six edge-based basis local functions for  $BDM_1$ :

$$\begin{aligned}\Phi_{1,s} &= \frac{\lambda_2 \mathbf{t}_3}{\sin(\theta_2)}, & \Phi_{1,e} &= -\frac{\lambda_3 \mathbf{t}_2}{\sin(\theta_3)}, \\ \Phi_{2,s} &= \frac{\lambda_3 \mathbf{t}_1}{\sin(\theta_3)}, & \Phi_{2,e} &= -\frac{\lambda_1 \mathbf{t}_3}{\sin(\theta_1)}, \\ \Phi_{3,s} &= \frac{\lambda_1 \mathbf{t}_2}{\sin(\theta_1)}, & \Phi_{3,e} &= -\frac{\lambda_2 \mathbf{t}_1}{\sin(\theta_2)}.\end{aligned}\tag{2.19}$$

It is clear that

- (i)  $\Phi_{i,s} \cdot \mathbf{n}_j = 0$  and  $\Phi_{i,e} \cdot \mathbf{n}_j = 0$  for  $i \neq j$ .
- (ii)  $\Phi_{i,s} \cdot \mathbf{n}_i$  and  $\Phi_{i,e} \cdot \mathbf{n}_i$  are linear on edge  $e_i$ .
- (iii)  $\nabla \cdot \Phi_{i,s}, \nabla \cdot \Phi_{i,e}$  can be calculated using (2.1) and (2.19).

Items (i) and (ii) are due to either orthogonality of the normal and tangential vectors or vanishing of a barycentric coordinate on the opposite edge.

The **hierarchy** among  $RT_0, RT_1, BDM_1$  elements is well known [14]:

$$RT_0 \subset BDM_1 \subset RT_1 \subset \dots$$

This is actually reflected in the edge-based local basis functions. In particular, the following holds

$$\phi_i^{RT_0} = \frac{1}{2} (\Phi_{i,s}^{BDM_1} + \Phi_{i,e}^{BDM_1}), \quad i = 1, 2, 3.\tag{2.20}$$

*Proof.* We consider edge  $e_1$ . It is clear that

$$\frac{|e_1|}{\sin(\theta_1)} = \frac{|e_2|}{\sin(\theta_2)} = \frac{|e_3|}{\sin(\theta_3)}.$$

So we have  $|e_2| \sin(\theta_3) = |e_3| \sin(\theta_2) = c$ , where for convenience  $c$  is used to name the same constant. Note that  $|T| = |e_3| |e_1| \sin(\theta_2) = |e_1| |e_2| \sin(\theta_3)$ . Taking coefficients  $\frac{1}{2} = c_s = c_e = \frac{|e_1|}{2|T|} c$  and using the facts that  $|e_3| \mathbf{t}_3 = P_2 - P_1$ ,  $-|e_2| \mathbf{t}_2 = P_3 - P_1$ , we have

$$\begin{aligned}c_s \Phi_{1,s}^{BDM_1} + c_e \Phi_{1,e}^{BDM_1} &= \frac{|e_1|}{2|T|} (\lambda_2 P_2 + \lambda_3 P_3 - \lambda_2 P_1 - \lambda_3 P_1) \\ &= \frac{|e_1|}{2|T|} (\lambda_2 P_2 + \lambda_3 P_3 + \lambda_1 P_1 - P_1) = \frac{|e_1|}{2|T|} (P - P_1) = \phi_1^{RT_0},\end{aligned}$$

where the facts  $\lambda_1 + \lambda_2 + \lambda_3 = 1$  and  $\lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3 = P$  have been used.

### 2.4.2 Normalized Bases for $RT_0$ , $BDM_1$ , $RT_1$

Let  $(x_c, y_c)$  be the element center and  $X = x - x_c, Y = y - y_c$ . The normalized local basis for triangular  $RT_0$  elements refers to [18, 20]

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} X \\ Y \end{bmatrix}. \quad (2.21)$$

The normalized local basis functions for triangular  $BDM_1$  elements are

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} X \\ 0 \end{bmatrix}, \begin{bmatrix} Y \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ X \end{bmatrix}, \begin{bmatrix} 0 \\ Y \end{bmatrix}. \quad (2.22)$$

Similarly, the normalized local basis for triangular  $RT_1$  elements is

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} X \\ 0 \end{bmatrix}, \begin{bmatrix} Y \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ X \end{bmatrix}, \begin{bmatrix} 0 \\ Y \end{bmatrix}, \begin{bmatrix} X^2 \\ XY \end{bmatrix}, \begin{bmatrix} XY \\ Y^2 \end{bmatrix}. \quad (2.23)$$

The hierarchy among these basis functions are trivial. It has been observed in [18] that the edge-based functions are convenient for implementing MFEMs, whereas the normalized basis functions are convenient for implementing WGFEMs.

It is clear that the Gram matrix of  $RT_1$  normalized basis takes the following form

$$GM = \begin{bmatrix} S1 & 0 & 0 & 0 & 0 & 0 & SX2 & SXY \\ 0 & SX2 & SXY & 0 & 0 & 0 & SX3 & SX2Y \\ 0 & SXY & SY2 & 0 & 0 & 0 & SX2Y & SXY2 \\ 0 & 0 & 0 & S1 & 0 & 0 & SXY & SY2 \\ 0 & 0 & 0 & 0 & SX2 & SXY & SX2Y & SXY2 \\ 0 & 0 & 0 & 0 & SXY & SY2 & SXY2 & SY3 \\ SX2 & SX3 & SX2Y & SXY & SX2Y & SXY2 & SX4 + SX2Y2 & SX3Y + SXY3 \\ SXY & SX2Y & SXY2 & SY2 & SXY2 & SY3 & SX3Y + SXY3 & SX2Y2 + SY4 \end{bmatrix},$$

where

$$S1 = |T|, \quad SX2 = \int_T X^2, \quad SXY = \int_T XY, \quad SX2Y2 = \int_T X^2 Y^2$$

and so on so forth. Note that  $\int_T X = 0, \int_T Y = 0$  have been used implicitly.

Hierarchy among the basis functions of the  $H(\text{div})$  finite elements is clear. This should be helpful for  $p$ -adaptivity of WGFEMs.

### 2.4.3 Conversion Among the Intrinsic and Natural Bases of $RT_0$

Conversion between the natural and normalized bases is easy:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - x_c \begin{bmatrix} 1 \\ 0 \end{bmatrix} - y_c \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Conversion from the intrinsic basis to the natural basis can be established based on the fact that

$$\frac{2|T|}{|e_i|}\phi_i = \begin{bmatrix} x \\ y \end{bmatrix} - x_i \begin{bmatrix} 1 \\ 0 \end{bmatrix} - y_i \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad i = 1, 2, 3.$$

This means that

$$\begin{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & -x_1 & -y_1 \\ 1 & -x_2 & -y_2 \\ 1 & -x_3 & -y_3 \end{bmatrix}^{-1} \begin{bmatrix} \frac{2|T|}{|e_1|}\phi_1 \\ \frac{2|T|}{|e_2|}\phi_2 \\ \frac{2|T|}{|e_3|}\phi_3 \end{bmatrix}.$$

## 2.5 Weak Galerkin Finite Elements

Let  $E$  be a triangular or rectangular element with interior  $E^\circ$  and boundary  $\partial E$ .

Let  $l, m$  be nonnegative integers,  $P^l(E^\circ)$  be the space of polynomials on  $E^\circ$  with degree  $\leq l$ , and  $P^m(\partial E)$  be the space of polynomials on  $\partial E$  with degree  $\leq m$ . A **discrete weak function** is a pair of scale-valued functions  $v = \{v^\circ, v^\partial\}$  such that  $v^\circ \in P^l(E^\circ)$  and  $v^\partial \in P^m(\partial E)$ . A discrete weak function space on  $E$  is defined as

$$W(E, l, m) = \{v = \{v^\circ, v^\partial\} : v^\circ \in P^l(E^\circ), v^\partial \in P^m(\partial E)\}. \quad (2.24)$$

Let  $n \geq 0$  be any integer,  $P^n(E)^2$  be the space of vector-valued polynomials with degree  $\leq n$ . Let  $V(E, n)$  be a subspace of  $P^n(E)^2$ . For any  $v \in W(E, l, m)$ , its **discrete weak gradient**  $\nabla_{w,d}v \in V(E, n)$  is so defined that it is the unique solution of

$$\int_E \nabla_{w,d}v \cdot \mathbf{w} dE = \int_{\partial E} v^\partial(\mathbf{w} \cdot \mathbf{n}) ds - \int_E v^\circ(\nabla \cdot \mathbf{w}) dE, \quad \forall \mathbf{w} \in V(E, n). \quad (2.25)$$

There could be many choices for  $P^l(E^\circ), P^m(\partial E), V(E, n)$ . But certain admissible conditions or properties shall be satisfied so that the discrete weak gradient operator  $\nabla_{w,d}$  offers a Galerkin-type approximation of the classical gradient operator [32]. Furthermore, a projection operator  $Q_h = (Q_h^\circ, Q_h^\partial)$  into a discrete weak function space can be defined, where  $Q_h^\circ$  is the  $L^2$ -projection into the polynomial space  $P^l(E^\circ)$  and  $Q_h^\partial$  is the  $L^2$ -projection into the polynomial space on  $P^m(\partial E)$ . Similarly,  $R_h$  is defined as the local (elementwise)  $L^2$  projection from  $L^2(E)^2$  to  $V(E, n)$ .

### 2.5.1 WG Elements $(P_0, P_0, RT_0)$ on Triangles

Let  $T$  be a triangular element with vertices  $P_i(x_i, y_i), i = 1, 2, 3$  oriented counterclockwise. Let  $|T|$  be the triangle area,  $(x_c, y_c)$  the center of the element,  $e_i (i = 1, 2, 3)$  the edge opposite to vertex  $P_i$ , and  $|e_i|$  edge length.

There is only one WG basis function for  $P_0(T^\circ)$ , which takes constant value 1 in the triangle interior, expressed as

$$\phi^\circ|_{T^\circ} = 1.$$

There are three WG basis functions for  $P_0(\partial T)$  that takes constant value 1 on one edge and 0 on the other two edges. In other words, we have

$$\phi_i^\partial|_{e_j} = \delta_{ij}, \quad 1 \leq i, j \leq 3,$$

where  $\delta_{ij}$  is the Kronecker symbol.

With the previously defined local WG basis functions, the global WG basis functions will be simply the “gluing-together” of the local basis functions.

Before the discrete weak gradients of the above 4 WG basis functions can be calculated, a set of basis functions for  $RT_0(T)$  need to be specified. These could be the edge-oriented basis functions [5, 23] or normalized basis functions [20]:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} X \\ Y \end{bmatrix},$$

where  $X = x - x_c, Y = y - y_c$ . For convenience, we name them as  $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$ . As can be seen later, one advantage of the latter is that the Gram matrix will be diagonal and hence trivially inverted.

The Gram matrix of the above normalized basis is simply diagonal:

$$GM = \text{diag}(|T|, |T|, S),$$

where

$$S = \frac{|T|}{36} ((x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_1)^2 + (y_1 - y_2)^2 + (y_2 - y_3)^2 + (y_3 - y_1)^2).$$

Next, the definition formula (2.25) is employed to calculate the discrete weak gradients of  $\phi^\circ, \phi_i^\partial (i = 1, 2, 3)$ . Since the Gram matrix of the above normalized basis is diagonal, one can easily obtain these discrete weak gradients

$$\begin{aligned} \nabla_{w,d}\phi^\circ &= \frac{|T|}{S} (0\mathbf{w}_1 + 0\mathbf{w}_2 + (-2)\mathbf{w}_3), \\ \nabla_{w,d}\phi_1^\partial &= \frac{y_3 - y_2}{|T|}\mathbf{w}_1 + \frac{x_2 - x_3}{|T|}\mathbf{w}_2 + \frac{2|T|}{3S}\mathbf{w}_3, \\ \nabla_{w,d}\phi_2^\partial &= \frac{y_1 - y_3}{|T|}\mathbf{w}_1 + \frac{x_3 - x_1}{|T|}\mathbf{w}_2 + \frac{2|T|}{3S}\mathbf{w}_3, \\ \nabla_{w,d}\phi_3^\partial &= \frac{y_2 - y_1}{|T|}\mathbf{w}_1 + \frac{x_1 - x_2}{|T|}\mathbf{w}_2 + \frac{2|T|}{3S}\mathbf{w}_3. \end{aligned}$$



### 2.5.2 WG Elements $(Q_0, P_0, RT_{[0]})$ on Rectangles

As discussed in [18], there are a variety of choices for weak Galerkin (WG) finite elements that can be used for solving the Darcy equation. However, we concentrate on the lowest order WG elements for the Darcy equation in subsurface flow. This is mainly because the pressure solution usually possesses low regularity in this situation, and thus higher order WG finite elements do not result in additional gains. In this subsection, we further focus on the lowest order WG elements on rectangular meshes.

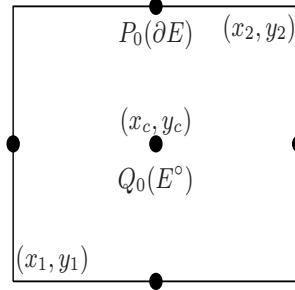


Figure 2.4: Five basis functions for a WG  $(Q_0, P_0, RT_{[0]})$  rectangular element: one constant basis function in the interior of the rectangle and one constant basis function for each of the four edges. Their discrete weak gradients are defined in  $RT_{[0]}$ .

Let  $E = [x_1, x_2] \times [y_1, y_2]$  be a typical rectangular element and  $(x_c, y_c)$  be its center. The divergence form of the Darcy equation (??) suggests that  $V(E, n)$  should be chosen as  $H(\text{div})$ -conforming. Therefore, the lowest order Raviart-Thomas space  $RT_{[0]}(E)$  is undoubtedly a good candidate. It is known that  $\dim(RT_{[0]}(E)) = 4$ . We choose normalized basis functions [20] for  $RT_{[0]}(E)$  as follows

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{w}_3 = \begin{bmatrix} X \\ 0 \end{bmatrix}, \quad \mathbf{w}_4 = \begin{bmatrix} 0 \\ Y \end{bmatrix},$$

where we have set  $X = x - x_c, Y = y - y_c$  for convenience. An obvious advantage of using the normalized basis is that the Gram matrix is diagonal

$$GM = \text{diag} \left( |E|, |E|, \frac{1}{12}(x_2 - x_1)^2|E|, \frac{1}{12}(y_2 - y_1)^2|E| \right),$$

where  $|E|$  is the area of the element. It is then trivial to invert the Gram matrix, which is needed for calculating the discrete weak gradients of the WG basis functions.

For WG basis functions, we choose  $l = 0, m = 0$  respectively for  $P^l(E^\circ), P^m(\partial E)$ . So we have one constant basis function  $\phi^\circ$  in the element interior  $E^\circ$  and one constant basis function  $\phi_i^\partial (i = 1, 2, 3, 4)$  on each of the four edges (together forming  $\partial E$ ) of the rectangle, as shown in Figure 2.4. Their discrete weak gradients, actually the

coefficients in the above  $RT_{[0]}(E)$  basis functions, can be calculated by employing the definition formula (2.25). The results are as follows

$$\left\{ \begin{array}{l} \nabla_{w,d}\phi^\circ = 0\mathbf{w}_1 + 0\mathbf{w}_2 + \frac{-12}{(x_2-x_1)^2}\mathbf{w}_3 + \frac{-12}{(y_2-y_1)^2}\mathbf{w}_4, \\ \nabla_{w,d}\phi_1^\partial = 0\mathbf{w}_1 + \frac{-1}{y_2-y_1}\mathbf{w}_2 + 0\mathbf{w}_3 + \frac{6}{(y_2-y_1)^2}\mathbf{w}_4, \\ \nabla_{w,d}\phi_2^\partial = \frac{1}{x_2-x_1}\mathbf{w}_1 + 0\mathbf{w}_2 + \frac{6}{(x_2-x_1)^2}\mathbf{w}_3 + 0\mathbf{w}_4, \\ \nabla_{w,d}\phi_3^\partial = 0\mathbf{w}_1 + \frac{1}{y_2-y_1}\mathbf{w}_2 + 0\mathbf{w}_3 + \frac{6}{(y_2-y_1)^2}\mathbf{w}_4, \\ \nabla_{w,d}\phi_4^\partial = \frac{-1}{x_2-x_1}\mathbf{w}_1 + 0\mathbf{w}_2 + \frac{6}{(x_2-x_1)^2}\mathbf{w}_3 + 0\mathbf{w}_4. \end{array} \right.$$

In summary, we have WG shape functions that are in  $Q_0$  inside rectangular elements (interiors) and  $P_0$  on edges (element interfaces) and their discrete weak gradients are in  $RT_{[0]}$ . This type of WG elements are referred as  $(Q_0, P_0, RT_{[0]})$ .

### 2.5.3 Higher Order WG Elements

Obviously, higher order WG finite elements can be established on triangles, rectangles, and quadrilaterals.

Here we briefly explain how WG  $(P_1, P_1, RT_1)$  finite elements are constructed on a typical triangle  $T$ .

First note that there are three WG basis function for  $T^\circ$ , they can be chosen as

$$1, X, Y,$$

where  $X = x - x_c$ ,  $Y = y - y_c$  and  $(x_c, y_c)$  is the center of  $T$ . For each of the three edges of this triangle, there are two basis functions. To continue the practice of normalized basis functions, we choose them as

$$1, Z,$$

where  $Z$  is the variable that parametrize the line segment represented by the edge and has range  $[-\frac{L}{2}, \frac{L}{2}]$ , assuming  $L$  is the length of the edge. Each edge has an orientation that agrees the counterclockwise orientation of the triangle. So  $-L/2$  corresponds to the starting node of the edge, whereas  $L/2$  corresponds to the ending node of the edge. In implementation, we can set

$$Z = \text{GAUSSQUADLINE}(\mathbf{k}, 2) - 0.5;$$

assuming `GAUSSQUADLINE` is a three-column 2-dim array consisting of the barycentric coordinates and weights of a chosen Gaussian quadrature for line segments.

Once again, we choose the 8 normalized basis functions for  $RT_1$ .

Next, we apply the definition formula (?)

$$\int_T \nabla_{w,d} \phi w = \int_{\partial T} \phi^\partial \mathbf{w} \cdot \mathbf{n} - \int_T \phi^\circ (\nabla \cdot \mathbf{w})$$

to the above chosen WG and RT basis functions to obtain the linear combination coefficients for the discrete weak gradients of the above 9 WG basis functions in the RT basis.

Note that the classical divergence of the 8 RT basis functions are

$$0, \quad 1, \quad 0, \quad 0, \quad 0, \quad 1, \quad 3X, \quad 3Y.$$

These are needed for finding the discrete weak gradients of the 3 WG basis functions for triangle interior. Specifically, integrals involving  $1, X, Y$  against the above divergence will be computed on the triangle. The highest-order terms are thus  $X^2, XY, Y^2$ . A 3-point Gaussian quadrature for triangles will be enough to get exactness of the integrals of these bivariate polynomials.

To find the discrete weak gradients of the 6 WG basis functions  $\phi$  on the triangle edges, we could ignore  $\nabla \cdot \mathbf{w}$  and  $\phi^\circ$ , since the latter vanish on  $T^\circ$ . We just focus on  $\mathbf{w} \cdot \mathbf{n}$ . Here  $\mathbf{n}$  is the unit outward normal vector on each, and its orientation needs to be settled correctly when we carry out computation simultaneously for the whole triangular mesh. The integrals involve univariate polynomials with highest degree 3, e.g.,

$$\int_{e_1} Z \begin{bmatrix} XY \\ Y^2 \end{bmatrix} \cdot \begin{bmatrix} n_1 \\ n_2 \end{bmatrix},$$

where  $X, Y$  are implicit linear functions of  $Z$ . A 2-point Gaussian quadrature for line segments will be enough to get exactness of the integrals of these univariate polynomials.

## 2.5.4 WG Finite Elements on Polygons

It is also possible / straightforward to develop WG finite elements on (2-dim) polygons and (3-dim) polyhedra with **flat** faces. Let  $E$  be a *convex* polygon with  $n \geq 3$  vertices  $P_i(x_i, y_i)$  ( $1 \leq i \leq n$ ) oriented counterclockwise. Edge  $e_i$  ( $1 \leq i \leq n$ ) connects vertex  $P_i$  with vertex  $P_{i+1}$ , where  $P_{n+1} = P_1$  by assumption for convenience. For any integer  $k \geq 1$ , we could establish a WG element  $(P_k, P_k, P_{k-1}^2)$ . So the lowest-order WG element is  $(P_1, P_1, P_0^2)$ . This means that we could choose  $1, X = x - x_c, Y = y - y_c$  (again  $(x_c, y_c)$  is the element center) as WG basis functions for  $E^\circ$ . For each edge  $e_i$  on  $E^\partial$ , we choose  $1, Z$  as WG basis functions, here  $Z \in [-l, l]/2$  and  $l = |e_i|$  is the edge length. Naturally,  $w_1 = [1, 0]^T, w_2 = [0, 1]^T$  form the simplest basis for the discrete weak gradients.

It can be easily verified that the discrete weak gradients of  $1, X, Y$  are the zero vector, so it the discrete weak gradient of  $Z$  on each edge. On the other hand,

$$\nabla_{w,0}(1|_{e_i}) = \frac{|e_i|}{|E|} \mathbf{n}_i = \frac{1}{|E|} \begin{bmatrix} y_{i+1} - y_i \\ x_i - x_{i+1} \end{bmatrix},$$

where  $|E|$  is the polygon area and  $\mathbf{n}_i$  is the outward unit normal vector on  $e_i$ .

### 2.5.5 WG Finite Elements on Quadrilaterals

Quadrilateral finite elements are needed in certain applications, e.g., monitoring of groundwater contamination.

We deal with convex quadrilaterals, which can be treated as bilinear mapping images of the unit square  $[0, 1]^2$ . Let  $Q$  be a convex quadrilateral with four vertices  $P_i(x_i, y_i)$  ( $i = 1, 2, 3, 4$ ) that are oriented counterclockwise. Let  $(\hat{x}, \hat{y}) \in [0, 1]^2$  and

$$\begin{cases} x = a_1 + a_2\hat{x} + a_3\hat{y} + a_4\hat{x}\hat{y}, \\ y = b_1 + b_2\hat{x} + b_3\hat{y} + b_4\hat{x}\hat{y}, \end{cases} \quad (2.26)$$

and the four corners of the unit square are mapped to the four corners of  $Q$ , then

$$\begin{cases} a_1 = x_1 \\ b_1 = y_1 \end{cases} \begin{cases} a_2 = x_2 - x_1 \\ b_2 = y_2 - y_1 \end{cases} \begin{cases} a_3 = x_4 - x_1 \\ b_3 = y_4 - y_1 \end{cases} \begin{cases} a_4 = (x_1 + x_3) - (x_2 + x_4) \\ b_4 = (y_1 + y_3) - (y_2 + y_4) \end{cases} \quad (2.27)$$

It is easy to find out that the Jacobian matrix and Jacobian determinant are respectively

$$\mathbf{J} = \begin{bmatrix} a_2 + a_4\hat{y} & a_3 + a_4\hat{x} \\ b_2 + b_4\hat{y} & b_3 + b_4\hat{x} \end{bmatrix} \quad (2.28)$$

$$\begin{aligned} J = \det(\mathbf{J}) &= (a_2b_3 - a_3b_2) + (a_2b_4 - a_4b_2)\hat{x} + (a_4b_3 - a_3b_4)\hat{y} \\ &= \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} + \begin{vmatrix} a_2 & a_4 \\ b_2 & b_4 \end{vmatrix} \hat{x} + \begin{vmatrix} a_3 & a_4 \\ b_3 & b_4 \end{vmatrix} \hat{y}. \end{aligned} \quad (2.29)$$

The above results indicate that the Jacobian determinant is a linear polynomial of  $\hat{x}, \hat{y}$ .

If we use vertex  $P_1, P_2, P_4$  to make a parallelogram, then the third vertex  $\tilde{P}_3$  will have coordinates  $\tilde{x}_3 = x_2 + x_4 - x_1, \tilde{y}_3 = y_2 + y_4 - y_1$ . It is interesting to see that

$$\begin{cases} a_4 = x_3 - \tilde{x}_3 = (x_1 + x_3) - (x_2 + x_4), \\ b_4 = y_3 - \tilde{y}_3 = (y_1 + y_3) - (y_2 + y_4). \end{cases}$$

So  $(a_4, b_4)$  is exactly the difference vector  $\overline{\tilde{P}_3 P_3}$ . In other words, the highest order coefficients in the bilinear mapping form a vector that describes exactly the departure of quadrilateral from parallelogram.

Along this line, we note that the midpoint  $P_{13}$  of the diagonal 13 is  $((x_1 + x_3)/2, (y_1 + y_3)/2)$ , whereas the midpoint  $P_{24}$  of the diagonal 24 is  $((x_2 + x_4)/2, (y_2 + y_4)/2)$ . The discrepancy of the two diagonal midpoints is

$$\overline{P_{24} P_{13}} = \frac{1}{2} (a_4, b_4).$$

The midway of these two midpoints is  $(x_c, y_c)$ , where  $x_c = (x_1 + x_2 + x_3 + x_4)/4, y_c = (y_1 + y_2 + y_3 + y_4)/4$ . Let the centroid of the unit square  $(\frac{1}{2}, \frac{1}{2})$  be mapped to  $(\tilde{x}, \tilde{y})$ . The difference between the midway and the centroid image is

## 2.6 Quadratures

Although several different types of quadratures (formulas for numerical integration) could be used, but we use only Gaussian quadratures, since they have the highest order possible exactness. Gaussian quadratures for line segments, triangles, rectangles, tetrahedra, and bricks (3-dim rectangles) are used in **DarcyLite**.

# Chapter 3

## Finite Element Methods for the Darcy Equation

There exists many finite element solvers for the Darcy equation, or more generally second order elliptic boundary value problems, e.g., continuous Galerkin finite element methods [11], discontinuous Galerkin finite element methods [6], the enhanced Galerkin (EG) finite element methods [29], the mixed finite element methods [13], and the newly developed weak Galerkin finite element methods [18, 32].

For any finite element solver, two important desired properties are (assuming  $\mathbf{u}_h$  is the numerical velocity)

- **Local mass conservation:** For any element  $E$  with  $\mathbf{n}$  being the outer normal vector on its boundary  $\partial E$ ,

$$\int_{\partial E} \mathbf{u}_h \cdot \mathbf{n} = \int_E f. \quad (3.1)$$

- **Normal flux continuity (in the integral form):** For an interior edge  $\gamma$  shared by two elements  $E_i (i = 1, 2)$  that have respectively unit outward normal vectors  $\mathbf{n}_i (i = 1, 2)$ :

$$\int_{\gamma} \mathbf{u}_h|_{E_1} \cdot \mathbf{n}_1 + \int_{\gamma} \mathbf{u}_h|_{E_2} \cdot \mathbf{n}_2 = 0. \quad (3.2)$$

### 3.1 Continuous Galerkin (CG) FEMs

CG solvers for the Darcy equation are based on discretizations of the primal variational formulation (1.2). CG  $P_1$  on triangular meshes and CG  $Q_1$  on rectangular meshes are most frequently used. For a triangular mesh  $\mathcal{T}_h$  with no hanging nodes, let  $V_h$  be the space of all continuous piecewise linear polynomials on  $\mathcal{T}_h$ . Let  $V_0^h = V_h \cap H_{D,0}(\Omega)$  and  $V_D^h = V_h \cap H_{D,p_D}(\Omega)$  (after Lagrangian interpolation of Dirichlet boundary data being

performed). A finite element scheme using CG  $P_1$  for pressure is established as: Seek  $p_h \in V_D^h$  such that

$$\sum_{T \in \mathcal{T}_h} \int_E \mathbf{K} \nabla p_h \cdot \nabla q = \sum_{T \in \mathcal{T}_h} \int_T f q - \sum_{\gamma \in \Gamma_h^N} \int_{\gamma} u_N q, \quad \forall q \in V_0^h. \quad (3.3)$$

A similar scheme can also be set up for CG  $Q_1$ . Then numerical velocity and normal fluxes are calculated *ad hoc*.

CG  $P_1$  basis functions are actually barycentric coordinates. Computation and assembly of element stiffness matrices are relatively easy. CG  $P_1$  or  $Q_1$  schemes have the least numbers of unknowns. However, CG schemes are neither locally mass-conservative nor have continuous normal flux across edges [15]. Post-processing is needed [11] to render numerical velocity the above two properties.

### 3.1.1 CG $P_1$ on Triangular Meshes

### 3.1.2 CG $Q_1$ on Rectangular Meshes

Any rectangular element  $[x_1, x_2] \times [y_1, y_2]$  can be mapped back to the reference element  $[0, 1]^2$ . Let  $(\hat{x}, \hat{y})$  be the coordinates in the reference element. Let the four vertices of either the reference element or a generic element be labeled counterclockwise, starting with the lower-left corner, that is,

$$P_1(x_1, y_1), P_2(x_2, y_1), P_3(x_2, y_2), P_4(x_1, y_2).$$

For convenience, we denote

$$\Delta x = x_2 - x_1, \quad \Delta y = y_2 - y_1.$$

The four Lagrangian  $Q_1$  basis functions are

$$\phi_1 = (1 - \hat{x})(1 - \hat{y}) = \frac{x_2 - x}{x_2 - x_1} \frac{y_2 - y}{y_2 - y_1}, \quad \phi_2 = \hat{x}(1 - \hat{y}), \quad \phi_3 = \hat{x}\hat{y}, \quad \phi_4 = (1 - \hat{x})\hat{y}.$$

It can be easily derived that

$$\nabla \phi_1 = \begin{bmatrix} \frac{-1}{\Delta x} \frac{y_2 - y}{y_2 - y_1} \\ \frac{-1}{\Delta y} \frac{x_2 - x}{x_2 - x_1} \end{bmatrix}$$

Finally we obtain the element stiffness (grad-grad) matrix as

$$\frac{\Delta y}{\Delta x} \frac{1}{6} \begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} + \frac{\Delta x}{\Delta y} \frac{1}{6} \begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix}.$$

Clearly this two-part structure stems from the Product Rule of Differentiation.

*Note a more reasonable or neat labeling and variables should be*

- $x_0, x_1, y_0, y_1$  as the left, right, bottom, top coordinates;
- $\Delta x = x_1 - x_0, \Delta y = y_1 - y_0$
- Labeling of four vertices follows the tensor-product order

$$P_{00}(x_0, y_0), P_{10}(x_1, y_0), P_{01}(x_0, y_1), P_{11}(x_1, y_1).$$

So the decimal values of the binary numbers are exactly 0, 1, 2, 3.

## 3.2 Discontinuous Galerkin (DG) FEMs

Weak formulation elementwise. Integration against a test function yields

$$\sum_{E \in \mathcal{E}_h} \int_E \nabla \cdot (-\mathbf{K} \nabla p) q = \sum_{E \in \mathcal{E}_h} \int_E f q.$$

Integration by parts on each element gives

$$\int_E \nabla \cdot (-\mathbf{K} \nabla p) q = \int_{\partial E} (-\mathbf{K} \nabla p \cdot \mathbf{n}) q + \int_E \mathbf{K} \nabla p \cdot \nabla q.$$

Suppose  $\gamma$  is an edge shared by two element  $E_1, E_2$ . Due to the discontinuity of the functions, we have to deal integrals on the edge as follows

$$\int_{\gamma} (-\mathbf{K}_1 \nabla p_1 \cdot \mathbf{n}_1 q_1 - \mathbf{K}_2 \nabla p_2 \cdot \mathbf{n}_2 q_2) =: \int_{\gamma} \mathcal{G}.$$

Manipulating the integrand, we rewrite it as

$$\begin{aligned} 2\mathcal{G} &= (-\mathbf{K}_1 \nabla p_1 \cdot \mathbf{n}_1 q_1 - \mathbf{K}_1 \nabla p_1 \cdot \mathbf{n}_1 q_2) \\ &\quad + (\mathbf{K}_1 \nabla p_1 \cdot \mathbf{n}_1 q_2 - \mathbf{K}_2 \nabla p_2 \cdot \mathbf{n}_2 q_2) \\ &\quad + (-\mathbf{K}_1 \nabla p_1 \cdot \mathbf{n}_1 q_1 + \mathbf{K}_2 \nabla p_2 \cdot \mathbf{n}_2 q_1) \\ &\quad + (-\mathbf{K}_2 \nabla p_2 \cdot \mathbf{n}_2 q_1 - \mathbf{K}_2 \nabla p_2 \cdot \mathbf{n}_2 q_2). \end{aligned}$$

Noting that  $\mathbf{n}_1 = -\mathbf{n}_2$ , we get

$$\begin{aligned} 2\mathcal{G} &= -\mathbf{K}_1 \nabla p_1 \cdot \mathbf{n}_1 (q_1 + q_2) + (-\mathbf{K}_1 \nabla p_1 - \mathbf{K}_2 \nabla p_2) \cdot (\mathbf{n}_2 q_2) \\ &\quad + (-\mathbf{K}_1 \nabla p_1 - \mathbf{K}_2 \nabla p_2) \cdot (\mathbf{n}_1 q_1) - \mathbf{K}_2 \nabla p_2 \cdot \mathbf{n}_2 (q_1 + q_2). \end{aligned}$$

So we have

$$-\mathbf{K}_1 \nabla p_1 \cdot \mathbf{n}_1 q_1 - \mathbf{K}_2 \nabla p_2 \cdot \mathbf{n}_2 q_2 = [[-\mathbf{K} \nabla p]] \{q\} + \{-\mathbf{K} \nabla p\} [[q]] \quad (3.4)$$



by introducing notations for averages and jumps:

$$\begin{aligned}
\{-\mathbf{K}\nabla p\} &= (-\mathbf{K}_1\nabla p_1 - \mathbf{K}_2\nabla p_2)/2, \\
[[-\mathbf{K}\nabla p]] &= (-\mathbf{K}_1\nabla p_1) \cdot \mathbf{n}_1 + (-\mathbf{K}_2\nabla p_2) \cdot \mathbf{n}_2, \\
\{q\} &= (q_1 + q_2)/2, \\
[[q]] &= q_1\mathbf{n}_1 + q_2\mathbf{n}_2.
\end{aligned} \tag{3.5}$$

It is interesting to notice that the jump of a scalar is a vector whereas the jump of a vector is a scalar.

Let  $\mathcal{E}_h$  be a rectangular or triangular mesh, define  $V_h^k$  as the space of (discontinuous) piecewise polynomials with a total degree  $\leq k$  on  $\mathcal{E}_h$ . For the boundary value problem (1.1), a DG scheme is to seek  $p_h \in V_h^k$  such that [29]

$$\mathcal{A}_h(p_h, q) = \mathcal{F}(q), \quad \forall q \in V_h^k, \tag{3.6}$$

where

$$\begin{aligned}
\mathcal{A}_h(p_h, q) &= \sum_{E \in \mathcal{E}_h} \int_E \mathbf{K} \nabla p_h \cdot \nabla q dE - \sum_{\gamma \in \Gamma_h^I \cup \Gamma_h^D} \int_{\gamma} \{\mathbf{K} \nabla p_h \cdot \mathbf{n}\} [q] ds, \\
&+ \beta \sum_{\gamma \in \Gamma_h^I \cup \Gamma_h^D} \int_{\gamma} \{\mathbf{K} \nabla q \cdot \mathbf{n}\} [p_h] ds + \sum_{\gamma \in \Gamma_h^I \cup \Gamma_h^D} \frac{\alpha_{\gamma}}{h_{\gamma}} \int_{\gamma} [p_h] [q] ds
\end{aligned} \tag{3.7}$$

$$\begin{aligned}
\mathcal{F}(q) &= \sum_{E \in \mathcal{E}_h} \int_E f q dE - \sum_{\gamma \in \Gamma_h^N} \int_{\gamma} u_N q ds \\
&+ \beta \sum_{\gamma \in \Gamma_h^D} \int_{\gamma} \mathbf{K} \nabla q \cdot \mathbf{n} p_D ds + \sum_{\gamma \in \Gamma_h^D} \frac{\alpha_{\gamma}}{h_{\gamma}} \int_{\gamma} p_D q ds.
\end{aligned} \tag{3.8}$$

Here  $\alpha_{\gamma} > 0$  is a penalty factor for any edge  $\gamma \in \Gamma_h$  and  $\beta$  is a formulation parameter [29]. Depending on the choice of  $\beta$ , one ends up with the symmetric interior penalty Galerkin (SIPG) for  $\beta = -1$ , the nonsymmetric interior penalty Galerkin (NIPG) for  $\beta = 1$ , and the incomplete interior penalty Galerkin (IIPG) for  $\beta = 0$ .

Stability of the above DG scheme for numerical pressure relies on the penalty factor. DG numerical velocity can be calculated *ad hoc*. The DG velocity is locally mass-conservative, but its normal component is not continuous across element boundaries. As pointed out in [6], this leads to nonphysical oscillations if the DG velocity is coupled to a DG transport solver in a straightforward manner. This could make particle tracking difficult or impossible if the DG velocity is used in a characteristic-based transport solver.

### 3.2.1 DG $P_k(k = 1, 2)$ for the Darcy Equation

### 3.2.2 Postprocessing for DGP1 and DGP2

## 3.3 Enhanced Galerkin (EG) FEMs

JL Remark: This section will be updated once the re-implementation of EG is done.

## 3.4 Weak Galerkin (WG) FEMs

WG finite element methods are developed based on the innovative concepts of weak gradients and discrete weak gradients [32]. Instead of using the *ad hoc* gradient of shape functions, discrete weak gradients are established at the element level to provide nice approximations of the classical gradient in PDEs.

The WG approach considers a discrete shape function in two pieces:  $v = \{v^\circ, v^\partial\}$ , the interior part  $v^\circ$  could be a polynomial of degree  $l \geq 0$  in the interior  $E^\circ$  of an element, the boundary part  $v^\partial$  on  $E^\partial$  could be a polynomial of degree  $m \geq 0$ , its discrete weak gradient  $\nabla_{w,n}v$  is specified in  $V(E, n) \subseteq P_n(E)^2, n \geq 0$  via integration by parts

$$\int_E (\nabla_{w,n}v) \cdot \mathbf{w} = \int_{\partial E} v^\partial (\mathbf{w} \cdot \mathbf{n}) - \int_E v^\circ (\nabla \cdot \mathbf{w}), \quad \forall w \in V(E, n).$$

For example, for a triangle  $T$ ,  $WG(P_0, P_0, RT_0)$  uses a constant basis function on  $T^\circ$ , a constant basis function on each edge on  $\partial T$ , and specifies the discrete weak gradients of these 4 basis functions in  $RT_0$ . Normalized basis functions for  $RT_0$  facilitates computation of these discrete weak gradients [18, 19, 15], although small-size linear systems need to be solved in general.

WG schemes for the Darcy equation rely on WG discretizations of the primal variation formulation (1.2). Let  $\mathcal{E}_h$  be a triangular or rectangular mesh,  $l, m, n$  nonnegative integers,  $S_h(l, m)$  the space of discrete shape functions on  $\mathcal{E}_h$  that have polynomial degree  $l$  in element interior and degree  $m$  on edges,  $S_h^0(l, m)$  the subspace of functions in  $S_h(l, m)$  that vanish on  $\Gamma^D$ . Seek  $p_h = \{p_h^\circ, p_h^\partial\} \in S_h(l, m)$  such that  $p_h^\partial|_{\Gamma^D} = Q_h^\partial p_D$  (projection of Dirichlet boundary data) and

$$\mathcal{A}_h(p_h, q) = \mathcal{F}(q), \quad \forall q = \{q^\circ, q^\partial\} \in S_h^0(l, m), \quad (3.9)$$

where

$$\mathcal{A}_h(p_h, q) := \sum_{E \in \mathcal{E}_h} \int_E \mathbf{K} \nabla_{w,n} p_h \cdot \nabla_{w,n} q, \quad (3.10)$$

$$\mathcal{F}(q) := \sum_{E \in \mathcal{E}_h} \int_E f q^\circ - \sum_{\gamma \in \Gamma_h^N} \int_\gamma u_N q. \quad (3.11)$$

After the numerical pressure  $p_h$  is obtained from solving (3.9), one computes the WG numerical velocity as follows

$$\mathbf{u}_h = R_h(-\mathbf{K}\nabla_{w,n}p_h), \quad (3.12)$$

where  $R_h$  is the local  $L^2$ -projection onto  $V(E, n)$ . But it can be skipped when  $\mathbf{K}$  is a constant scalar on the element. Normal fluxes are computed accordingly. It is shown in [18, 19, 32] that WG schemes are locally conservative and produce continuous normal fluxes (in the integral form).

#### 3.4.1 WG $(P_0, P_0, RT_0)$ on Triangles for the Darcy Equation

#### 3.4.2 WG $(Q_0, P_0, RT_{[0]})$ on Rectangles for the Darcy Equation

#### 3.4.3 WG $(Q_0, P_0, RT_{[0]})$ on Quadrilaterals for the Darcy Equation

For a general quadrilateral mesh, we could still apply WG  $(Q_0, P_0, RT_{[0]})$  to solve the Darcy equation. In this approach, we avoid the Piola transformation. Of course, this type of WG finite elements do not produce discrete weak gradients in a  $H(\text{div})$ -subspace. But we still have normal flux continuity for the WG finite element solution of the Darcy equation. The asymptotic  $h^2$ -assumption for quadrilateral meshes will be a sufficient condition for good approximation, but this assumption is not unusual.

Similar to a rectangle, we specify for

$$RT_{[0]}(Q) = \text{Span}(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4),$$

where

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{w}_3 = \begin{bmatrix} X \\ 0 \end{bmatrix}, \quad \mathbf{w}_4 = \begin{bmatrix} 0 \\ Y \end{bmatrix},$$

and  $X = x - x_c, Y = y - y_c$ ,  $(x_c, y_c)$  is the quadrilateral barycenter. Again the Gram matrix of this basis is

$$GM = \begin{bmatrix} |E| & 0 & \int_E X & 0 \\ 0 & |E| & 0 & \int_E Y \\ \int_E X & 0 & \int_E X^2 & 0 \\ 0 & \int_E Y & 0 & \int_E Y^2 \end{bmatrix},$$

where  $|E|$  is the quadrilateral element area. This can be treated as the sum of the area of the two consisting triangles. Other integrals can still be evaluated exactly using a sufficiently higher order Gaussian quadrature on the reference rectangle  $[0, 1]^2$ , since  $X, Y, X^2, XY, Y^2$  along with the Jacobian of the bilinear transformation are just bivariate polynomials.

Note that generally speaking, the following vanishing momentum property

$$\int_E X = 0, \quad \int_E Y = 0,$$

no longer holds on a quadrilateral.

With a general (elementwise constant) permeability tensor

$$\mathbf{K} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix},$$

we need to deal with the projection of  $\mathbf{K}\mathbf{w}_i$  ( $i = 1, 2, 3, 4$ ) back into the subspace  $RT_{[0]}(Q)$ . Specifically,

$$\mathbf{K}\mathbf{w}_1 = K_{11}\mathbf{w}_1 + K_{21}\mathbf{w}_2,$$

$$\mathbf{K}\mathbf{w}_2 = K_{12}\mathbf{w}_1 + K_{22}\mathbf{w}_2.$$

But for  $\text{Proj}\mathbf{K}\mathbf{w}_i$ , ( $i = 3, 4$ ), we need solve a small-size SPD linear system. The coefficient matrix is obviously the Gram matrix, whereas the right-hand sides for  $\text{Proj}(\mathbf{K}\mathbf{w}_i)$ ,  $i = 3, 4$  are respectively

$$\begin{bmatrix} K_{11} \int_E X \\ K_{21} \int_E X \\ K_{11} \int_E X^2 \\ K_{21} \int_E XY \end{bmatrix}, \quad \begin{bmatrix} K_{12} \int_E Y \\ K_{22} \int_E Y \\ K_{12} \int_E XY \\ K_{22} \int_E Y^2 \end{bmatrix}.$$

### 3.4.4 WG $(P_1, P_1, P_0^2)$ on Polygons for the Darcy Equation

For Dirichlet boundary data, we use  $L_2$ -projection. For each edge in the polygonal mesh, WG  $P_1$  basis functions are assigned. So we could choose  $1, Z$  as the basis functions, where

$$Z = \text{GAUSSQUAD.LINE}(k, 2) - 0.5.$$

It can be verified that the Gram matrix for this basis is an order-2 diagonal matrix

$$\text{diag}(|e|, |e|/12),$$

where  $|e|$  is the edge length. More specifically, the projection coefficients form a 2-vector as below

$$\begin{bmatrix} (1/|e|) \int_e p_D \\ (12/|e|) \int_e p_D Z \end{bmatrix}.$$

### 3.5 Mixed Finite Element Methods (MFEMs) for the Darcy Equation

Mixed finite element schemes for the Darcy equation are based on discretizations of the mixed variational form (1.4). A pair of finite element spaces  $(V_h, W_h)$  are chosen such that  $V_h \subset H(\text{div}, \Omega)$  for approximating velocity and  $W_h \subset L^2(\Omega)$  for approximating pressure and together they satisfy the inc-sup condition. Among the popular choices are  $(RT_0, P_0)$ ,  $(BDM_1, P_0)$  for triangular meshes and  $(RT_{[0]}, Q_0)$  for rectangular meshes [18, 19].

For a rectangular or triangular mesh  $\mathcal{E}_h$ , denote  $U_h = V_h \cap H_{u_N, N}(\text{div}; \Omega)$  and  $V_h^0 = V_h \cap H_{0, N}(\text{div}; \Omega)$ . A mixed finite element scheme can be stated as: Seek  $\mathbf{u}_h \in U_h$  and  $p_h \in W_h$  such that the following hold

$$\begin{cases} \sum_{E \in \mathcal{E}_h} \int_E \mathbf{K}^{-1} \mathbf{u}_h \cdot \mathbf{v} - \sum_{E \in \mathcal{E}_h} \int_E p_h (\nabla \cdot \mathbf{v}) = - \sum_{\gamma \in \Gamma_h^D} \int_{\gamma} p_D (\mathbf{v} \cdot \mathbf{n}) & \forall \mathbf{v} \in V_h^0, \\ - \sum_{E \in \mathcal{E}_h} \int_E (\nabla \cdot \mathbf{u}_h) q = - \sum_{E \in \mathcal{E}_h} \int_E f q & \forall q \in W_h. \end{cases} \quad (3.13)$$

For implementation of  $(RT_0, P_0)$ , the edge-based basis functions for  $V_h$  are usually used in computation and assembly of element matrices [18, 19]. Then a symmetric indefinite linear system needs to be solved. This is taken care by Matlab backslash, although it is nontrivial behind the scene.

An obvious advantage of the MFEM schemes is the automatic local mass conservation (obtained by taking test function  $q = 1$  in (3.13) 2nd equation) and normal flux continuity (built in the construction of  $H(\text{div})$  finite element spaces). It is interesting to observe certain equivalence between MFEM schemes and WG finite element schemes [18].

See [14].

#### 3.5.1 MFEM $(RT_0, P_0)$ on a Triangular Mesh

It is known that the  $(RT_0, P_0)$  element pair on a triangular mesh does satisfy the inf-sup condition.

Let  $N_e$  be the number of elements in a given triangular mesh  $\mathcal{E}_h$ ,  $N_1$  be the number of the interior and Dirichlet edges, and  $N_2$  be the number of the Neumann edges.

For the entire space  $RT_0(\mathcal{E}_h)$ , there are mainly two groups of global vector basis functions

- (1)  $\Phi_{i_1}, 1 \leq i_1 \leq N_1$  associated with the interior and Dirichlet edges;
- (2)  $\Phi_{i_2}, 1 \leq i_2 \leq N_2$  associated with the Neumann edges.

The first two groups of basis functions actually span the subspace  $\mathbf{V}_h^0 = RT_0(\mathcal{E}_h) \cap H_{0,N}(\text{div}, \Omega)$ .

To establish an approximation from  $RT_0(\mathcal{E}_h)$  to the subset  $H_{u_N,N}(\text{div}, \Omega)$ , a projection of the boundary Neumann boundary condition is needed. We define

$$\bar{u}_{N,\gamma} = \frac{1}{|\gamma|} \int_{\gamma} u_N d\gamma \quad \forall \gamma \in \Gamma_h^N, \quad (3.14)$$

and

$$\bar{\mathbf{u}}_N = \sum_{\gamma \in \Gamma_h^N} \bar{u}_{N,\gamma} \Phi_{\gamma}. \quad (3.15)$$

It is now clear that

$$\mathbf{V}_h = \mathbf{V}_h^0 + \bar{\mathbf{u}}_N. \quad (3.16)$$

The space of piecewise constant functions on a given triangular mesh  $\mathcal{E}_h$  is defined as

$$P_1(\mathcal{E}_h) = \{q \in L_2(\Omega) : q|_E = \text{const} \quad \forall E \in \mathcal{E}_h\}, \quad (3.17)$$

whose basis functions can be taken as

$$\psi_E = 1 \quad \text{on } E^\circ \in \mathcal{E}_h \quad \text{and } 0 \quad \text{elsewhere.} \quad (3.18)$$

**MFEM ( $RT_0, P_0$ ) for Darcy Equation:** Seek  $\mathbf{u}_h \in \mathbf{U}_h$  and  $p_h \in W_h$  such that for any  $\mathbf{v} \in \mathbf{V}_h$ ,  $q \in W_h$

$$\begin{cases} (\mathbf{K}^{-1} \mathbf{u}_h, \mathbf{v})_{\mathcal{E}_h} - (p_h, \nabla \cdot \mathbf{v})_{\mathcal{E}_h} = - (p_D, \mathbf{v} \cdot \mathbf{n})_{\Gamma_h^D}, \\ - (\nabla \cdot \mathbf{u}_h, q)_{\mathcal{E}_h} = - (f, q)_{\mathcal{E}_h}. \end{cases} \quad (3.19)$$

Clearly, the unknowns are the normal fluxes on the interior and Dirichlet edges and the pressures on all elements.

Let  $\Phi_{\gamma_j}$  be the global vector basis function associated with edge  $\gamma_j$ ,  $1 \leq j \leq N_1 + N_2$  and  $\psi_l$  the global scalar basis function associated with element  $E_l$ ,  $1 \leq l \leq N_e$ . Let

$$\mathbf{u}_h = \sum_{j=1}^{N_1} u_k \Phi_{\gamma_j} + \sum_{j=N_1+1}^{N_1+N_2} \bar{u}_{N,\gamma_j} \Phi_{\gamma_j}, \quad p_h = \sum_{l=1}^{N_e} p_{E_l} \psi_{E_l}.$$

The term  $(\mathbf{K}^{-1} \mathbf{u}_h, \mathbf{v})_{\mathcal{E}_h}$  is assembled over all elements. At the element level,  $(\mathbf{K}^{-1} \mathbf{u}_h, \mathbf{v})_E$  involves the three edges of a typical element  $E$ . So there could be interaction among interior, Dirichlet, and Neumann edges. If we simply treat the knowns associated with the Neumann edges as unknowns and allow the Neumann edge basis functions in assembly, then we end up with a global coefficient matrix as follows

$$\begin{bmatrix} A_{11} & A_{12} & B^T \\ A_{21} & A_{22} & C^T \\ B & C & 0 \end{bmatrix}$$

There are choices for enforcing the Neumann boundary condition. A straightforward ("brutal") way is to modify the second row blocks. One obtains

$$\begin{bmatrix} A_{11} & A_{12} & B^T \\ 0 & I & 0 \\ B & C & 0 \end{bmatrix} \begin{bmatrix} U \\ \bar{U}_N \\ P \end{bmatrix} = \begin{bmatrix} D \\ \bar{U}_N \\ F \end{bmatrix},$$

where  $I$  is the order  $N_2$  identity matrix. But this results in a global coefficient matrix that is neither symmetric nor definite.

Another approach is to separate the Neumann edges (basis functions) from the interior and Dirichlet edges, then one has

$$\begin{bmatrix} A_{11} & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} U \\ P \end{bmatrix} = \begin{bmatrix} D - A_{12}\bar{U}_N \\ F - C\bar{U}_N \end{bmatrix},$$

as shown by formula (2.7) and (2.8) in [5].

Notice that the right-hand side in the above linear system can be obtained from

$$\begin{bmatrix} D \\ 0 \\ F \end{bmatrix} - \begin{bmatrix} A_{11} & A_{12} & B^T \\ A_{21} & A_{22} & C^T \\ B & C & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \bar{U}_N \\ 0 \end{bmatrix}$$

by ignoring the second row block in the final result. This motivates an easier ("gentle") and symmetric way for (assembling the global linear system and) enforcing Neumann boundary conditions:

- (i) Set DOFs to number of all edges plus number of all elements;
- (ii) Assemble the two global stiffness matrices by allowing participation of all edge-based basis functions and all element-based functions;
- (iii) Assemble the two terms in the global right-hand side: the Dirichlet boundary integral and the source integral;
- (iv) Set the Neumann boundary values as knowns and modify the global RHS;
- (v) Set the flux unknowns on the interior and Dirichlet edges and the pressure unknowns on all elements together as "free unknowns", solve a smaller linear system (submatrix, sub-RHS).

For item (iii)-(v), see [5] subsection 6.1, 6.2.1, and 6.3.1 and the Matlab code on pp. 348–350.

Anyway we deal with a discrete linear system

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} U \\ P \end{bmatrix} = \begin{bmatrix} D \\ F \end{bmatrix}. \quad (3.20)$$

This is a saddle-point problem.

### Uzawa Algorithm

Several issues:

- Submatrix  $A$ : SPD, largest and smallest eigenvalues;
- Submatrix  $B$ : singular value decomposition;
- Matrix  $BA^{-1}B^T$ : condition number, sparsity;

### 3.5.2 MFEM ( $RT_{[0]}, Q_0$ ) on a Rectangular Mesh

### 3.5.3 On the Equivalence between WGFEMs and MFEMs

See [18, 19]. It has been observed in our numerical experiments that WGFEMs and MFEMs produce very close numerical results in pressure and flux, when permeability is piecewise constant, even though these two types of methods are established on totally different concepts. There actually exists some “equivalence” between WGFEMs and MFEMs, which can be established via hybridized MFEMs.

We examine the equivalence for a simple case  $\mathbf{K} = \mathbf{I}_2$ :

$$\nabla \cdot (-\nabla p) = f \text{ on } \Omega, \quad p|_{\partial\Omega} = 0.$$

Let  $\mathcal{E}_h$  be a triangular mesh. Note that the MFEM using  $V_h = RT_0(\mathcal{E}_h) \subset H(\text{div}, \Omega)$  and  $W_h = P_0(\mathcal{E}_h) \subset L^2(\Omega)$  seeks  $\mathbf{U}_h \in V_h, P_h \in W_h$  such that

$$\begin{cases} (\mathbf{U}_h, \mathbf{v}) - (P_h, \nabla \cdot \mathbf{v}) = 0, & \forall \mathbf{v} \in V_h, \\ -(\nabla \cdot \mathbf{U}_h, w) = -(f, w), & \forall w \in W_h. \end{cases} \quad (3.21)$$

The normal flux continuity built in the definition of  $V_h = RT_0(\mathcal{E}_h)$  can be relaxed through hybridization, i.e., using a space of Lagrange multipliers  $M_h = P_0(\Gamma_h)$  on the edges and a new space  $\tilde{V}_h = \Pi_{E \in \mathcal{E}_h} RT_0(E) \supset V_h$ . This leads to the hybridized MFEM: seek  $\tilde{\mathbf{U}}_h \in \tilde{V}_h, P_h \in W_h, \lambda_h \in M_h$  such that

$$\begin{cases} (\tilde{\mathbf{U}}_h, \mathbf{v})_{\mathcal{E}_h} - (P_h, \nabla \cdot \mathbf{v})_{\mathcal{E}_h} + (\lambda_h, \mathbf{v} \cdot \mathbf{n})_{\Gamma_h} = 0, & \forall \mathbf{v} \in \tilde{V}_h, \\ -(\nabla \cdot \tilde{\mathbf{U}}_h, w) = -(f, w), & \forall w \in W_h, \\ (\tilde{\mathbf{U}}_h \cdot \mathbf{n}, \mu) = 0, & \forall \mu \in M_h. \end{cases} \quad (3.22)$$

The first equation in (3.22) can be rewritten elementwise as

$$(-\tilde{\mathbf{U}}_h, \mathbf{v})_E = (\lambda_h, \mathbf{v} \cdot \mathbf{n})_{\partial E} - (P_h, \nabla \cdot \mathbf{v})_E, \quad \forall E \in \mathcal{E}_h.$$

If we interpret  $\{P_h, \lambda_h\} =: \{p_h^\circ, p_h^\partial\} =: p_h$  as a discrete weak function, then the right-hand side of the above equation becomes  $(\nabla_{w,d} p_h, \mathbf{v})_E$ , according to the definition formula (2.25) for discrete weak gradients. So  $-\tilde{\mathbf{U}}_h = \nabla_{w,d} p_h$ , or  $\tilde{\mathbf{U}}_h = -\mathbf{K} \nabla_{w,d} p_h$ , since  $\mathbf{K} = \mathbf{I}_2$  in this case.



This explains the closeness of the numerical solutions of MFEM and WGFEM observed in Example ?. If the space for discrete weak gradients in the WGFEM is chosen as the same vector function space for velocity in the MFEM, then the WGFEM and MFEM have the same degrees of freedom, but the WGFEM has less degrees of freedom than that of the hybridized MFEM. Notice that the degrees of freedom for the WGFEM are all for the pressure, whereas the degrees of freedom for the MFEM are for both velocity and pressure.

Of course, this “equivalence” is not a full equivalence. When the source and boundary conditions are nonzero, or the permeability varies spatially, the difference between the WGFEM solution (pressure, flux) and the MFEM solution will be higher order of the mesh size. We shall investigate this further theoretically and numerically.

### 3.6 Finite Volume Methods for the Darcy Equation

For convenience, we consider a (2-dim) polygonal domain  $\Omega$  that is equipped with a triangular mesh  $\mathcal{T}_h$ .

A dual mesh is established by connecting edge midpoints and triangle bary-centers. For each triangle, there are three small/short edges that connect the edge midpoints to the bary center. Assume the three vertices of a triangle  $T$  are oriented counterclockwise and labeled as  $P_1, P_2, P_3$ . For convenience, denote the edge midpoint opposite the vertex  $P_i (i = 1, 2, 3)$  as  $Q_i (i = 1, 2, 3)$  and the triangle center as  $Q$ . The three small/short edges going from  $Q_i$  to  $Q$  are denoted as  $\gamma_i (i = 1, 2, 3)$ . We take the tangential vector of such a directed edge as itself, without considering normalization (unit length). By rotating the tangential vector clockwise by  $90^\circ$ , we get a normal vector that has the length of the edge. This laziness is actually useful in the integrals for the lower order finite volume methods.

For (Matlab) implementation, we shall use mainly the info of the primal mesh and the above minimal info of the dual mesh. So we don’t really formulate the dual mesh as shown in most mathematical papers. That formulation might need to arrange the dual nodes (edge midpoints and triangle center) in the counterclockwise orientation. This is doable in 2-dim by calculating the angles of the line segments that connect those points to a specified node in the primal mesh. In 3-dim, this will be more complicated.

There is an interesting 3-by-3 matrix (see Table ?) that characterize the necessary sign changes.

Short edge	1	2	3
Vertex #1	0	-1	1
#2	1	0	-1
#3	-1	1	0

Note that for the node-oriented finite volume method for the Darcy equation, the

unknown pressure unknown is approximated by a continuous piecewise linear polynomial on the triangular mesh. The DOFs are set at the nodes. So the trial functions are the same as those for CG  $P_1$ . A control volume is constructed around each node by connecting the midpoints of the adjacent edges and the bary centers of the adjacent triangles. The test function is defined on the control value and is actually the constant function 1.

Starting from the standard form of the Darcy equation, one has

$$\int_{\partial V} (-\mathbf{K}\nabla p) \cdot \mathbf{n} = \int_V f \quad (3.23)$$

For efficient implementation, we actually do not construct the construct completely. We still use the small/short edges connecting the edge midpoints and the triangle center, but we work in the primary triangle.

Of course, the control volume  $V$  consists of many small dual triangles. Gaussian quadrature is applied to each dual triangle when computing the source term  $\int_V f$ .

# Chapter 4

## FEMs for Transient Convection-diffusion-reaction Equations

There are now three mechanisms/processes: convection, diffusion, and reaction. Which mechanism is dominant? When the three mechanisms are at comparable scales, all numerical methods work. When convection or reaction dominates, there could be non-physical oscillations, undershoots and overshoots [12]. Numerical methods need to be carefully designed to solve the problems accurately and robustly.

### 4.1 A Transport Solver Based on Implicit Euler and Weak Galerkin

DarcyLite contains also finite element solvers for transport problems in 2-dim prototyped as

$$\begin{cases} c_t + \nabla \cdot (\mathbf{v}c - D\nabla c) + Rc = f(x, y, t), & (x, y) \in \Omega, t \in (0, T), \\ c(x, y, t) = 0, & (x, y) \in \partial\Omega, t \in (0, T), \\ c(x, y, 0) = c_0(x, y), & (x, y) \in \Omega. \end{cases} \quad (4.1)$$

Here  $c(x, y, t)$  is the unknown (solute) concentration,  $\mathbf{v}$  the Darcy velocity,  $D > 0$  a diffusion constant,  $f$  a source/sink term. There exist various types of finite element methods for transport problems. Here we briefly discuss a numerical scheme that utilizes the implicit Euler for time-marching and weak Galerkin for spatial approximation.

For simplicity, we assume  $\Omega$  is a rectangular domain equipped with a rectangular mesh  $\mathcal{E}_h$  and a numerical velocity  $\mathbf{u}_h$  has been obtained from applying a finite element solver that is locally mass-conservative and has continuous normal fluxes. The unknown concentration is approximated using the lowest order finite element space

$WG(Q_0, P_0, RT_{[0]})$ . Let  $C^{(n)} (n \geq 1)$  be such an approximation at discrete time  $t_n$ , then there holds for  $n \geq 1$ ,

$$\begin{aligned}
& \sum_{E \in \mathcal{E}_h} (C^{(n)}, w)_E - \sum_{E \in \mathcal{E}_h} (C^{(n)}, \mathbf{u}_h \cdot \nabla_{w,d} w)_E \\
& \quad + \Delta t D \sum_{E \in \mathcal{E}_h} (\nabla_{w,d} C^{(n)}, \nabla_{w,d} w)_E \\
& = \sum_{E \in \mathcal{E}_h} (C^{(n-1)}, w)_E + \Delta t \sum_{E \in \mathcal{E}_h} (f, w)_E.
\end{aligned} \tag{4.2}$$

An initial approximant  $C^{(0)}$  can be obtained via local  $L_2$ -projection of  $c_0(x, y)$  into the WG finite element space.

The above “Implicit Euler + Weak Galerkin” scheme has two nice properties as stated below.

- (i)  $C^{(n)}|_{E^\circ}$  represents intuitively the *cell average of concentration* on any element  $E$ .
- (ii) The scheme is *locally conservative* and hence globally conservative. This can be verified by taking a test function  $w$  that has value 1 in one element interior but 0 in all others and on all edges.

The 2nd term on the left side of the above equation characterizes interaction of the flow (Darcy velocity) and the concentration (discrete weak) gradient. The 3rd term is a symmetric term similar to that in any elliptic problem. The 1st term on the right side represents the mass at the previous time moment, whereas the last term depicts the source/sink contribution during the time period  $[t_{n-1}, t_n]$ .

# Chapter 5

## FEMs for Steady-state Convection-diffusion-reaction Equations

Steady-state convection-diffusion-reaction equations are also important and very useful for real applications. It can be regarded as the limit case of transient convection-diffusion-reaction equation when the concentration profile  $c(x, y, t)$  does not change with time anymore, that is,  $c_t = 0$ .

The reaction term could be linear or nonlinear. But we investigate linear reaction first, which is usually modeled as  $Rc$ . Here  $R$  could be generally a function of space and term. The simplest case is that  $R = \text{const}$ .

Actually a more important issue is about the scale (size) of  $R$ . There are now three mechanisms/processes: convection, diffusion, and reaction. Which mechanism is dominant? When the three mechanisms are at comparable scales, all numerical methods work. When convection or reaction dominates, there could be non-physical oscillations, undershoots and overshoots. Numerical methods need to be carefully designed to solve the problems accurately and robustly.

Accordingly, the steady-state (“transport”) convection-diffusion-reaction-reaction (CDR) equation is formulated as (in the conservative form)

$$\begin{cases} \nabla \cdot (\mathbf{v}c - D\nabla c) + Rc = f(x, y), & (x, y) \in \Omega \\ c(x, y) = 0, & (x, y) \in \partial\Omega \end{cases} \quad (5.1)$$

### 5.1 WG for Steady-state CDR Equations

Here is WGFEM for steady-state convection-diffusion-reaction (CDR) equation

$$- \sum_{E \in \mathcal{E}_h} (C_h, \mathbf{u}_h \cdot \nabla_{w,d} w)_E + \sum_{E \in \mathcal{E}_h} (D \nabla_{w,d} C_h, \nabla_{w,d} w)_E + \sum_{E \in \mathcal{E}_h} (R C_h, w)_E = \sum_{E \in \mathcal{E}_h} (f, w)_E. \quad (5.2)$$

## 5.2 FVM for Steady-state CDR Equations

Here we consider the node-oriented finite volume methods for steady-state CD equations. Let  $c(x, y)$  be the unknown concentration. Let  $z$  be a typical node and  $V_z$  be the control volume containing  $z$ . It is obvious that we have

$$\int_{\partial V_z} (\mathbf{v} \cdot \mathbf{n}) c + \int_{\partial V_z} (-D \nabla c) \cdot \mathbf{n} = \int_{V_z} f. \quad (5.3)$$

The first term on the left-side will be treated thru upwinding, the second term on the left-side and the source term on the right-side are treated in a way similar to those for the Darcy equation in Section ?

# Chapter 6

## FEMs for Other Types of Equations

### 6.1 FEMs for Parabolic Equations

**Local mass conservation.** Let  $C^{(n)}(\mathbf{x})$  be the approximate concentration at time  $t_n$ . Then it is expected that

$$\int_E C_h^{(n)}(x) - \int_E C_h^{(n-1)}(x) + \int_{t_{n-1}}^{t_n} \int_{\partial E} (-D \nabla C_h^{(n)}) \cdot \mathbf{n} = \int_{t_{n-1}}^{t_n} \int_E f(\mathbf{x}, t) \quad \forall E \in \mathcal{E}_h \quad (6.1)$$

The difference of the first and second terms on the left-hand side expresses the net change of the mass over a fixed element at two time moments. The third term on the left-hand side expresses the overall outward normal diffusive (bulk) flux on the boundary of this element. The term on the right-hand side is the overall mass contribution by all sources (sinks) in the the element.

#### 6.1.1 CGFEMs for Parabolic Equations

The diffusion in the parabolic equation actually brings smoothness to the solution. So there is certain continuity in the exact solution of the parabolic equation. Thus it is natural to apply the continuous Galerkin finite element methods to the parabolic BIVP, especially the lowest order CGP1 on triangular meshes.

Let  $\mathcal{T}_h$  be a triangular mesh on  $\Omega$ ,  $0 = t_0 < t_1 < \dots < t_{n-1} < t_n < \dots < t_N = T$  be a partition of  $[0, T]$ ,  $C_n^{(n)}$  be the numerical solution at time  $t_n$ . Then CGP1 takes the form

$$\begin{aligned} & \sum_{T \in \mathcal{T}_h} C^{(n)}(x, y) \psi(x, y) \, dx dy + \Delta t \sum_{T \in \mathcal{T}_h} D \nabla C^{(n)} \cdot \nabla \psi \, dx dy \\ &= \sum_{T \in \mathcal{T}_h} C^{(n-1)}(x, y) \psi(x, y) \, dx dy + \Delta t \sum_{T \in \mathcal{T}_h} f(x, y, t) \psi(x, y) \, dx dy. \end{aligned} \quad (6.2)$$

Unfortunately CGP1 for the parabolic problem is generally *not* locally mass-conservative, even though the residual could be small.

### 6.1.2 WGFEMs for Parabolic Equations

Let  $\mathcal{T}_h$  be a family of conforming quasi... triangular mesh on  $\Omega$ . We choose WG  $(P_0, P_0, RT_0)$  finite elements for approximation.

**Theorem** (Local mass conservation for WGFEMs) For WG  $(P_0, P_0, RT_0)$  finite element approximation combined with the implicit Euler for time-marching, then the method is locally mass-conservative.

$$C_h^n \Big|_{E^\circ} |E| - C_h^{(n-1)} \Big|_{E^\circ} |E| + \int_{t_{n-1}}^{t_n} \int_{\partial E} (-D \nabla_{w,d} C_h^{(n)}) \cdot \mathbf{n} = \int_{t_{n-1}}^{t_n} \int_E f(\mathbf{x}, t). \quad (6.3)$$

*Proof.*

## 6.2 FEMs for Linear Elasticity

Note that our format for Neumann (traction) condition is

$$-\sigma(\mathbf{u}) \mathbf{n} = \mathbf{t}_N.$$

It carries a negative sign, to be consistent in writing with the Neumann condition for the Darcy equation.

### 6.2.1 Classical FEMs for 2-dim Linear Elasticity

**CGP1<sup>2</sup> for 2d Elasticity.** See [1, 28].

**CGQ1<sup>2</sup> for 2d Elasticity.** See [1]. But notice that the so-called "quadrilateral elements" in [1] are actually parallelogram elements.

### 6.2.2 WGFEMs for 2-dim Linear Elasticity

**WG FE Scheme.**

$$\mathcal{A}_s(\mathbf{u}_h, \mathbf{v}) = \mathcal{F}(\mathbf{v}), \quad \forall \mathbf{v} \in \quad (6.4)$$

where the  $\mathcal{F}$  is defined

$$\mathcal{A}_s(\mathbf{u}_h, \mathbf{v}) = \sum_{E \in \mathcal{E}_h} 2\mu(\varepsilon_w(\mathbf{u}_h), \varepsilon_w(\mathbf{v}))_E + \sum_{E \in \mathcal{E}_h} \lambda(\nabla_w \cdot \mathbf{u}_h, \nabla_w \cdot \mathbf{v})_E \quad (6.5)$$

and the stabilizer is defined as

$$\mathcal{S}(\mathbf{u}_h, \mathbf{v}) = \sum_{E \in \mathcal{E}_h} h_E^{-1} (Q_h^\partial \mathbf{u}_h^\circ - \mathbf{u}_h^\partial, Q_h^\partial \mathbf{v}^\circ - \mathbf{v}^\partial)_{E^\partial} \quad (6.6)$$



Here the weak strain tensor is defined as

$$\varepsilon_w(\mathbf{v}) = \frac{1}{2} (\nabla_w \mathbf{v} + (\nabla_w \mathbf{v})^T), \quad (6.7)$$

and the weak stress tensor is defined as

$$\sigma_w(\mathbf{v}) = 2\mu\varepsilon_w(\mathbf{v}) + \lambda(\nabla_w \cdot \mathbf{v})\mathbf{I}. \quad (6.8)$$

## 6.3 FEMs for Linear Poroelasticity (PE)

### 6.3.1 WGFEMs for Poroelasticity

# Chapter 7

## Testcases

### 7.1 Testcases for 2-dim Darcy Equation

### 7.2 Testcases for 2-dim Transport Problems

### 7.3 Testcases for 2-dim Elasticity

#### 7.3.1 Examples with Known Analytical Solutions

See [1].

**L-shape domain.** For this example, the domain  $\Omega$  is an L-shaped domain that has vertices (in the counterclockwise orientation)  $(2, 0), (0, 2), (-1, 1), (0, 0), (-1, -1), (0, -2)$ . It is a bit unusual but natural for using polar coordinates. The known exact solution has the form

$$\begin{cases} u_1(r, \theta) = \frac{1}{2\mu} r^a \left( -(1+a) \cos((1+a)\theta) + (C_2 - 1 - a)C_1 \cos((1-a)\theta) \right), \\ u_2(r, \theta) = \frac{1}{2\mu} r^a \left( (1+a) \sin((1+a)\theta) - (C_2 - 1 + a)C_1 \sin((1-a)\theta) \right), \end{cases} \quad (7.1)$$

where the critical exponent  $a \approx 0.544483737$  is actually the solution of an algebraic equation  $\sin(2\omega)a + \sin(2\omega a) = 0$ , with  $\omega = 3\pi/4$ ,  $C_1 = -\cos((1+a)\omega)/\cos((1-a)\omega)$ ,  $C_2 = 2(\lambda + 2\mu)/(\lambda + \mu)$ . The body force is zero. Furthermore,  $E = 1e6$ ,  $\nu = 0.3$ .

In this example, the origin is a re-entry corner, the exact solution has singularity at the origin.

#### 7.3.2 Cook's Membrane and Cantilever Beams

**Cook's membrane.** For this problem,  $\Omega$  is a trapezoidal domain generated by  $(0, 0), (48, 44), (48, 60), (0, 44)$ . The material is plexiglass with  $E = 2900$ ,  $\nu = 0.4$ . It shows bending-dominated elastic response. Usually body force is set to zero  $\mathbf{f} = (0, 0)$ .

In [8], a mixed boundary problem is posed as follows: Dirichlet boundary conditions: on the left side, homogeneous  $\mathbf{u}_D = (0, 0)$ . Neumann boundary conditions: on the other three sides. On the right side, traction  $(0, 1)$ . On the top and bottom sides, traction zero. Note that the jump in the boundary conditions at the domain corner  $(0, 44)$  [8] (JL Remark:  $(0, 0)$  also?) affects the solution regularity so that  $\mathbf{u} \notin H^2(\Omega)^2$ .

Brenner-Sung nonconforming  $P_1^2$  finite element method [7] cannot be applied directly because of the Neumann boundary.

Triangular meshes [8] or quadrilateral meshes [16] can be used.

**Cantilever beam.** See [?, 16].

Cantilever beams exist widely, either naturally (like a tree branch), or manmade (like a hanging beam for traffic lights).

A rectangular beam with a horizontal width  $A$  and a vertical height  $B$  is pinned at the lower-left corner. This means both the horizontal and vertical displacement at this point are zero. Furthermore, the horizontal displacement and vertical (surface) traction along the left end are both zero. The right end (called free end) is subject to a linearly varying force with a maximal value  $f$ . As usual,  $E$  denotes the Young's modulus and  $\nu$  is the Poisson ratio.

The known analytical solution for the displacement  $\mathbf{u} = (u_1, u_2)$  is

$$\begin{cases} u_1 = \frac{1 - \nu^2}{E} \frac{2}{B} x \left( \frac{B}{2} - y \right) f, \\ u_2 = \frac{1 - \nu^2}{E} \frac{1}{B} \left( x^2 - \frac{\nu}{1 - \nu} y(B - y) \right) f. \end{cases} \quad (7.2)$$

Typical test values are  $E = 1500$ ,  $\nu = 0.4999$  (or another value close to  $\frac{1}{2}$ ),  $f = 3000$ ,  $A = 10$ ,  $B = 2$ , see [16].

## 7.4 Testcases for 2-dim Poroelasticity

### 7.4.1 Mandel's Problem

See [26], Mandel [?], Abousleiman [?] also. This problem has a known analytical solution, although it involves infinite series and is complicated.

The Mandel's problem involves a poroelastic rectangular slab with extent  $2a$  in the  $x$ -direction and extent  $2b$  in the  $y$ -direction being sandwiched by two rigid plates at the top and the bottom. Two forces of magnitude  $2F$ , pointing to the slab, are applied respectively at the top and bottom plates. Due to the rigidity of the plates, the slab remains in contact with the two plates. This implies that the vertical displacement at the top and bottom sides are uniform.

Due to the symmetry in the problem, we choose the center of the slab as the origin and consider the upper-right quadrant. Mathematically, the Mandel problem is thus formulated on domain  $\Omega = (0, a) \times (0, b)$  for a certain time period  $(0, T)$ . The unknown functions, the solid displacement  $\mathbf{u}(x, y, t) = (u_1, u_2)$  and the fluid pressure  $p(x, y, t)$ , satisfy the coupled equations (?) and (?). The boundary conditions for the solid displacement are

- (Dirichlet)  $u_1 = 0$  for  $x = 0$ ; (symmetry)
- (Dirichlet)  $u_2 = 0$  for  $y = 0$ ; (symmetry)
- (Dirichlet)  $u_2 = \text{time-dependent constant}$  for  $y = b$ . (**rigid-plate-constraint**)
- (Neumann, traction)  $\tau \cdot \mathbf{n} = (0, -2F)$  for  $y = b$ :
- (Neumann, traction)  $\tau \cdot \mathbf{n} = \mathbf{0}$  otherwise.

The boundary conditions for the fluid pressure are

- (Dirichlet)  $p = 0$  for  $x = a$ ; (drained)
- (Neumann, no-flux)  $(-\mathbf{K}\nabla p) \cdot \mathbf{n} = 0$  for  $x = 0, y = 0, y = b$ .

The initial conditions are just zeros:

- $\mathbf{u}(x, y, 0) = \mathbf{0}$ ;
- $p(x, y, 0) = 0$ .

**Some details about the new physical parameters.** See [26]. Starting from Lamé constants  $\lambda, \mu$ . The skeleton bulk modulus  $K$ , Young's modulus  $E$ , Poisson's coefficient  $\nu$  are expressed as

$$K = \lambda + \frac{2}{3}\mu, \quad E = \frac{9K}{3K + \mu} \mu, \quad \nu = \frac{3K - 2\mu}{2(3K + \mu)}. \quad (7.3)$$

These three parameters have an "undrained" version also:  $K_u, E_u, \nu_u$ . In particular,

$$K_u = K + \frac{\alpha^2}{c_0}. \quad (7.4)$$

Based on the above formulas, the fluid diffusivity coefficient  $c_f$  and Skempton's coefficient  $B$  are defined as

$$c_f = \frac{\kappa}{c_0} \frac{K + \frac{4}{3}\mu}{K_u + \frac{4}{3}\mu}, \quad B = \frac{\alpha}{c_0 K_u}. \quad (7.5)$$

Note that  $c_f$  is defined only for the case when the permeability is a constant ( $\mathbf{K} = \kappa \mathbf{I}$ ). Skempton's coefficient is further related to Poisson's coefficient and its undrained version

$$B = \frac{3}{\alpha(1-2\nu)} \frac{\nu_u - \nu}{1 + 2\nu_u}. \quad (7.6)$$

Let  $\alpha_n$  be the  $n$ -th positive numerical solution of the nonlinear equation

$$\tan(\alpha_n) = \frac{1-\nu}{\nu_u - \nu} \alpha_n, \quad n = 1, 2, \dots \quad (7.7)$$

For convenience, we introduce normalized variables

$$\tilde{x} = \frac{x}{a}, \quad \tilde{t} = \frac{c_f}{a} t. \quad (7.8)$$

We don't use  $\bar{x}, \bar{t}$ , since these are not for nondimensionalization. Let

$$A_n = \frac{\sin(\alpha_n)}{\alpha_n - \sin(\alpha_n) \cos(\alpha_n)}, \quad (7.9)$$

**Known analytical series solution for displacement.**

$$\frac{u_1}{F/a} = \left( \frac{\nu}{2\mu} - \frac{\nu_u}{\mu} + \sum_{n=1}^{\infty} A_n e^{-\alpha_n \tilde{t}} \cos \alpha_n \right) x + \frac{1}{\mu} \sum_{n=1}^{\infty} B_n e^{-\alpha_n^2 \tilde{t}} \sin(\alpha_n \tilde{x}) \quad (7.10)$$

and

$$\frac{u_2}{F/a} = \left( -\frac{1-\nu}{2\mu} + \frac{1-\nu_u}{\mu} \sum_{n=1}^{\infty} A_n e^{-\alpha_n^2 \tilde{t}} \cos \alpha_n \right) y. \quad (7.11)$$

**Known analytical series solution for solid stress.** In addition to  $\sigma_{xx} = \sigma_{xy} = 0$ , there holds

$$\frac{\sigma_{yy}}{F/a} =: \widetilde{\sigma_{yy}} = -1 + 2 \sum_{n=1}^{\infty} A_n e^{-\alpha_n^2 \tilde{t}} \left( \cos \alpha_n - \frac{B(\nu_u - \nu)}{1 - \nu} \cos(\alpha_n \tilde{x}) \right). \quad (7.12)$$

**Known analytical series solution for fluid pressure.** Note that the pressure is independent of  $y$ :

$$\frac{p(x, y, t)}{\frac{F}{a} \frac{B(1+\nu_u)}{3}} = 2 \sum_{n=1}^{\infty} A_n e^{-\alpha_n^2 \tilde{t}} (\cos(\alpha_n \tilde{x}) - \cos \alpha_n) =: \tilde{p}(\tilde{x}, \tilde{t}). \quad (7.13)$$

In this sense,  $\tilde{p}(\tilde{x}, \tilde{t})$  can be interpreted as a normalized pressure.

### **7.4.2 Terzaghi's Problem**

See [27] also.

This is actually a 1-dim problem, but contains interesting things for regularity and testing convergence rates.

### **7.4.3 Barry and Mercer Point-Source Problem**

See [27] also.

This is a point-source problem on a 2-dim rectangular domain. The problem might not describe any real-world situations, but it demonstrates some interesting mathematical properties. Its known analytical solution involves infinite series also.

### **7.4.4 Some Benchmarks Used in COSMOL**

# Chapter 8

## Design and Implementation Strategies of DarcyLite

The main ideas for design and implementation strategies of **DarcyLite** are explained in [21]. **DarcyLite** is implemented in Matlab, which is a popular interpretative language. Although it is different than the compilation languages like C/C++ or Fortran, Matlab is very efficient in matrix or array operations, since its core was implemented mainly in Fortran. This also means that Matlab arrays are stored columnwise. The integrated development environment and graphics functionalities offered by Matlab are very helpful for educational purpose.

The above facts guide our overall design of **DarcyLite**:

- *Code modules and separation of tasks.*
- *Minimization of function calls.* Function calls invoke stack operations, which could be time-consuming.
- *Columnwise storage and access for full matrices or arrays.*
- *Utilizing Matlab  $(i, j, s)$ -format for sparse matrices.*
- *Performing operations on a mesh for all elements simultaneously, instead of looping through each element.*

In this section, we further elaborate on the above ideas through a list of selected topics. Some data structures and implementation techniques in **DarcyLite** share the same spirit as those in [2, 5, 9].

### 8.1 Assumptions

**DarcyLite** focuses on 2-dimensional flow and transport problems, but few modules and examples are available for 3-dim problems. This is reflected in our naming convention

for functions and files. "2-dim" are standard cases and not shown in names, whereas "3d" are attached to those for 3-dimensional problems and solvers.

For convenience of coding, we assume the permeability in the Darcy equation is a constant scalar or  $2 \times 2$  matrix on each element. This can be realized by taking the average of  $\mathbf{K}$  on each element, if  $\mathbf{K}$  does not change dramatically. Otherwise, multiscale WG might be used. Depending on whether  $\mathbf{K}$  is a scalar or tensor, we have four code modules

```
function PermK = Darcy_SmplnPerm_RectMesh(fxnK,RectMesh,GAUSSQUAD)
function PermK = Darcy_SmplnPerm_TriMesh(fxnK,TriMesh,GAUSSQUAD)
function PermKt = Darcy_SmplnPermTen_RectMesh(fxnKt,RectMesh,GAUSSQUAD)
function PermKt = Darcy_SmplnPermTen_TriMesh(fxnKt,TriMesh,GAUSSQUAD)
```

## 8.2 Mesh Data Structure

Mesh information can be categorized as geometric (node coordinates, position of element center, etc.), topological (an element vs its nodes, adjacent edges of a given edge, etc.), and combinatorial (number of neighboring nodes of a given node, etc.). Some basic mesh data are used by all types of finite element solvers, even though they use some derived mesh info in different ways. Mesh data can be organized as

- (i) *Primary* info such as number of nodes/elements, node coordinates, element vs its nodes  
`NumNds, NumEms, node, elem`
- (ii) *Secondary* info such as edge vs its vertices, edge vs its one or two neighboring elements, element vs its all edges  
`NumEgs, edge, edge2elem, elem2edge, LenEg, area, EmCntr`
- (iii) *Tertiary* info are more specific to particular finite element solvers. For instance, DG needs to know what are the neighboring elements for a give element, MFEM and WG need specific info on whether an edge is the 2nd edge of a given element. The info on the (unit) normal vector of a given edge is also used by MFEM and WG.

For example, the geometric info about mesh node coordinates is organized as an array of size `NumNds*2`, the topological info about triangular elements vs their vertices is organized as an array of size `NumEms*3`. Shown below is a code fragment for computing all triangle areas based on the columnwise storage in Matlab.

```
k1 = TriMesh.elem(:,1); k2 = TriMesh.elem(:,2); k3 = TriMesh.elem(:,3);
x1 = TriMesh.node(k1,1); y1 = TriMesh.node(k1,2);
x2 = TriMesh.node(k2,1); y2 = TriMesh.node(k2,2);
x3 = TriMesh.node(k3,1); y3 = TriMesh.node(k3,2);
TriMesh.area = 0.5*abs((x2-x1).*(y3-y1)-(x3-x1).*(y2-y1));
```



Based on these three levels of mesh info, `DarcyLite` organizes mesh data as a structure that has several dynamic fields. The fields can be easily added or removed. Code modules are provided for mesh info enrichment (adding data fields at a higher level), see code modules

```
TriMesh_Enrich1.m, TriMesh_Enrich3.m,
RectMesh_Enrich1.m, RectMesh_Enrich3.m
```

### 8.3 Implementation of Quadratures

We use only Gaussian quadratures.

Integrals on elements and element interfaces comprise a major portion of computation in finite element methods. Although this looks like a simple issue, but code efficiency can be improved when we adopt an unconventional approach. For example, in the  $WG(P_0, P_0, RT_0)$  scheme for the Darcy equation, one needs to compute the integrals  $\int_T f$  over all triangular elements. If a  $K$ -point Gaussian quadrature is applied, the conventional approach would be

$$|T| \sum_{k=1}^K f(\alpha_k P_1 + \beta_k P_2 + \gamma_k P_3) w_k, \quad \forall T \in \mathcal{T}_h,$$

where  $(\alpha_k, \beta_k, \gamma_k), w_k (k = 1, \dots, N)$  are respectively the barycentric coordinates and weights of the quadrature points. The order of summation and loop can be reversed (when elements in the mesh are all triangles), as shown below

```
GlbRHS = zeros(DOFs,1);
NumQuadPts = size(GAUSSQUAD.TRIG,1);
for k=1:NumQuadPts
    qp = GAUSSQUAD.TRIG(k,1) * TriMesh.node(TriMesh.elem(:,1),:)...
        + GAUSSQUAD.TRIG(k,2) * TriMesh.node(TriMesh.elem(:,2),:)...
        + GAUSSQUAD.TRIG(k,3) * TriMesh.node(TriMesh.elem(:,3),:);
    GlbRHS(1:NumEms) = GlbRHS(1:NumEms)...
        + GAUSSQUAD.TRIG(k,4) * EqnBC.fxnf(qp);
end
GlbRHS(1:NumEms) = GlbRHS(1:NumEms) .* TriMesh.area;
```

### 8.4 Element-level Full Matrices and Mesh-level Sparse Matrices

Finite schemes involve computation of element-level small-size full matrices, e.g., mass matrices and stiffness matrices, and mesh-level large-size sparse matrices, e.g., the global

stiffness matrix. In general, we avoid using cell arrays of usual matrices, since they may be inefficient. Instead we use 3-dim arrays. For example, in the  $WG(P_0, P_0, RT_0)$  scheme for the Darcy equation, each triangle involves one WG basis function for element interior and three WG basis functions for the edges. The discrete weak gradient of each of the four WG basis functions is a linear combination of the three  $RT_0$  normalized basis functions [18]. We organize these coefficients as a three-dimensional array

```
CDWGB = zeros(TriMesh.NumEms, 4, 3);
```

Then the interaction of the discrete weak gradients of the three edge WG basis functions results in a 3-dim array

```
ArrayGG = zeros(TriMesh.NumEms, 3, 3);
```

These element-level small-size full matrices will be assembled into the mesh-level sparse global stiffness matrix

```
DOFs = TriMesh.NumEms + TriMesh.NumEgs;
GlbMat = sparse(DOFs, DOFs);
```

Instead of using a loop over all elements, we utilize the  $(i, j, s)$ -structure in Matlab and the mesh topological info:

```
for i=1:3
    II = TriMesh.NumEms + TriMesh.elem2edge(:, i);
    for j=i:3 % Utilizing symmetry
        JJ = TriMesh.NumEms + TriMesh.elem2edge(:, j);
        if (j==i)
            GlbMat = GlbMat + sparse(II, II, ArrayGG(:, i, i), DOFs, DOFs);
        else
            GlbMat = GlbMat + sparse(II, JJ, ArrayGG(:, i, j), DOFs, DOFs);
            GlbMat = GlbMat + sparse(JJ, II, ArrayGG(:, j, i), DOFs, DOFs);
        end
    end
end
end
```

See the following code modules for more details:

```
Darcy_WG_TriPOPORTO_AsmSlv.m, Darcy_WG_RectQOPORTO_AsmSlv.m
```

## 8.5 Enforcing Essential Boundary Conditions

For CG and WG, Dirichlet boundary conditions are essential, but Neumann boundary conditions are natural, see Equations (3.3)(3.9). For MFEM, Neumann conditions are

essential but Dirichlet conditions are natural, see Equation (3.13). For DG, Dirichlet conditions are enforced weakly [29] in the bilinear and linear forms, see Equations (3.6,3.7,3.8). Theoretically, a zero essential boundary condition corresponds to a nice subspace for the numerical solution in the finite element scheme. For a non-zero essential condition, however, we usually use the terminology “manifold” to accommodate the numerical solution. From the implementation viewpoint, to enforce essential boundary conditions, we need to modify the global sparse linear system obtained from a finite element scheme. There are basically two approaches, hard (“brutal”) and soft (“gentle”). We illustrate the latter using a simple linear system.

Let  $\{x_1, x_2\}$  be unknowns that satisfy a linear system as follows

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Assume  $x_2 = c$  is the essential boundary condition. A simple/hard/brutal way is to modify the linear system to

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ c \end{bmatrix}.$$

But this might damage the symmetry in the original linear system. An alternative is to consider solving  $A_{11}x_1 = b_1 - A_{12}c$ , which is a linear system with a smaller size and maintains symmetry of  $A_{11}$  shown in the original system. Note that

$$\begin{bmatrix} b_1 - A_{12}c \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} - \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} 0 \\ c \end{bmatrix}$$

involves a simple matrix-vector multiplication (using the original global coefficient matrix) and we just need to take the first block in the final result vector. See the following code modules for more details:

```
Darcy_CG_RectQ1t_AsmSlv.m,  
Darcy_WG_TriPOPORTO_AsmSlv.m, Darcy_WG_RectQOPORTO_AsmSlv.m
```

## 8.6 Interface with Other Software Packages

**DarcyLite** focuses on solving flow and transport problems in the environment provided by Matlab. It can import triangular meshes generated by other packages, e.g., **PDE Toolbox** and **DistMesh** [25], which also run on Matlab. **DarcyLite** provides functions for triangular mesh conversion from the aforementioned two packages.

**Triangle** is a popular triangular mesh generator developed in C. **DarcyLite** can read in the mesh data files generated by **Triangle** in the so-called flat file transfer (FFT) approach.

## 8.7 Graphical User Interface (GUI) of DarcyLite

A graphical user interface (GUI) is provided with **DarcyLite** for demonstration and education purposes. It was developed using Matlab built-in **guide** (graphical user interface development environment). The GUI is based on the event-driven design principle and some techniques in SENSAT [30] are adopted.

*I. Preparation.* This part allows the user to specify details for a problem to be solved.

(A) A domain needs to be defined, including the endpoints along the x- and y-axis. The default settings suggest a rectangular domain with both axis starting at the origin 0 and ending at 1.

(B) The Darcy Equation is specified by choosing or uploading a permeability  $K$  and a source term  $f$ .

(C) Dirichlet and Neumann boundary conditions are specified. The two built-in options for  $K$ ,  $f$ , and Dirichlet and Neumann boundary conditions are defined according to two popular test examples.

(D) A mesh is to be generated. The user can choose between a rectangular or triangular mesh by specifying the numbers of uniform partitions  $nx, ny$  for the  $x, y$ -directions, respectively. Default settings suggest  $nx = 20, ny = 20$ . The **Show Mesh** button gives the user a chance to check whether a correct mesh has been generated.

(E) Gaussian type quadratures (for edges, rectangles, and triangles) are to be chosen. In most cases, the default choices (5,9,13) are sufficient, although the user can chose to change the orders of quadratures.

*II. Numerical Method.* This part of the GUI lists the four major types of finite element solvers. The user can choose among Continuous Galerkin, Discontinuous Galerkin, Weak Galerkin, or Mixed Finite Element Method. Then the user must click the **Run Code** button. A popup will confirm that the inputs from Part I are correct for the chosen finite element solver. Otherwise, an error message will pop up.

*III. Presentation.* The checkboxes in this part allow the user to display any combination of the numerical pressure and velocity profiles, local mass-conservation residual (LMCR), and flux discrepancy across edges.

### Extra Stuff

For Matlab implementation, we focus on 2-dim problems, although most implementation strategies and techniques can be extended to 3-dim problems.

#### I. Preparation / Setup

1. Domain
2. Equation Coefficients
3. Boundary Conditions
4. Mesh

## 5. Quadratures

### II. Processing / Solving

CG P1

CG Q1

DG P1 with post-processing

DG P2 with post-processing

WG(P0,P0,RT0)

WG(Q0,Q0,RT[0])

MFEM(RT0,P0)

MFEM(RT[0],Q0)

### III. Presentation / Show

Numerical pressure & velocity combined

Local mass conservation residual

Flux discrepancy

# Chapter 9

## Using DarcyLite

### 9.1 A Quick Start

Here is a quick start.

- (1) Download the single-piece zip file `DarcyLite.zip` from J.Liu's webpage:  
`http://www.math.colostate.edu/~liu/code.html`
- (2) Unzip the zip file to get a folder with the default name "DarcyLite";
- (3) Run `setpath.m` in Matlab environment;
- (4) Get into subfolder "project" ;
- (5) Run `run-Darcy-WG-TriPORT0t.m` or a similar file in Matlab environment;  
(JL Remark: Read the hyphen(-) in the above filename as underscore)  
Observe graphics and numerical results in Matlab command window.

### 9.2 More Details

A natural way of using `DarcyLite` is to write a Matlab script file that calls the functions in `DarcyLite`. There are many good examples in the subfolder `project`. One performs computation/simulation task within a project. This echoes the ideas in Microsoft Visual C++ or Apple Xcode.

All the filenames in `DarcyLite` are self-explanatory.

Listed below are typical steps in a FE task.

- Set up a physical problem to be solved:  
Domain, boundary conditions,
- Generate a mesh;

- Set up quadratures (on line segments, rectangles, triangles) to be used
- Sample the permeability for solving Darcy equation;
- Call a FE solver (CG, DGPP, WG, MFEM)
- Perform appropriate post-processing
- Present numerical and graphical results.

# Chapter 10

## Extra Stuff To Be Reorganized

### 10.1 Solving Darcy Equation by the Lowest Order WG Elements

For numerical solutions of the Darcy equation, we prefer the lowest order WG finite elements WG  $(P_0, P_0, RT_0)$  on triangles and WG  $(Q_0, P_0, RT_{[0]})$  on rectangles or quadrilaterals.

### 10.2 Working with Quadrilateral Meshes

It is interesting to notice that

- A triangular mesh can be refined into a quadrilateral mesh by partitioning each triangle into three quadrilaterals. This is done by connecting the centroid (barycentric center) of the triangle with the midpoints of the three edges of the triangle.
- A quadrilateral mesh can be refined into a triangular mesh by simply partitioning each quadrilateral into two triangles. There are two choices for doing so. One could use the diagonal 13 or the diagonal 24. However, we use the shorter diagonal. This avoids slim triangles.

#### 10.2.1 Generation of Quadrilateral Meshes

There are three ways of generating quadrilateral meshes. Perturbation on a given domain, refinement from a given triangular mesh and refinement from a given quadrilateral mesh.

We make perturbation on three types of domain, rectangular, circular sector domain and hydrogeological in practical.



On a given rectangular domain, which the ranges and partitions of x- and y-direction are given, the logically rectangular quadrilateral mesh data is organized as two status. Here, we have the code modulus.

```
function QuadriMesh = RectDom_QuadriMesh_GenLogiRect(
xa,xb,nx,yc,yd,ny,delta,status)
```

Status 1(*Primary* info) includes number of nodes/elements, node coordinate, element vs its nodes.

```
QuadriMesh.NumNds, QuadriMesh.NumEms, QuadriMesh.node, QuadriMesh.elem
```

In order to get a general quadrilateral mesh, after labelling the nodes of rectangular mesh, use an arbitrary value **delta** and **sin**, which is related with partitions **nx** and **ny**, to make a perturbation and then get the coordinates of a general quadrilateral mesh.

```
k = 1;
for i=1:(nx+1)
    for j=1:(ny+1)
        X(k) = X(k) + delta * sin((i-1)/nx*pi*2) * sin((j-1)/ny*pi*2);
        Y(k) = Y(k) + delta * sin((i-1)/nx*pi*2) * sin((j-1)/ny*pi*2);
        k = k + 1;
    end
end
```

Labelling the four vertices of quadrilateral in counterclockwise orientation.

```
QuadriMesh.elem(:,1) = lbl(:); % lower-left
QuadriMesh.elem(:,2) = QuadriMesh.elem(:,1) + (ny+1); % lower-right
QuadriMesh.elem(:,3) = QuadriMesh.elem(:,1) + (ny+2); % upper-right
QuadriMesh.elem(:,4) = QuadriMesh.elem(:,1) + 1; % upper-left
```

Status 2(*Secondary* info) includes edge vs its vertices, element vs its all edges, edge vs its one or two neighboring elements.

```
QuadriMesh.edge, QuadriMesh.elem2edge, QuadriMesh.edge2elem, QuadriMesh.elem,
QuadriMesh.area, QuadriMesh.EmCntr.
```

To label the four vertices of quadrilateral, separate nodes into two groups, vertical edges and horizontal edges.

```
% Vertical edges
lbl = LblNd(1:ny,1:nx+1);
QuadriMesh.edge(1:NumEgsVert,1) = lbl(:);
QuadriMesh.edge(1:NumEgsVert,2) = QuadriMesh.edge(1:NumEgsVert,1) + 1;
% Horizontal edges
```

```

lbl = LblNd(1:ny+1,1:nx);
QuadriMesh.edge(NumEgsVert+1:end,1) = lbl(:);
QuadriMesh.edge(NumEgsVert+1:end,2) = QuadriMesh.edge(NumEgsVert+1:end,1) + (ny+1);

```

To label the four edges of quadrilateral, divide edges into odd and even ones.

```

lbl = LblEgVert(1:ny,1:nx);
QuadriMesh.elem2edge(:,4) = lbl(:);
QuadriMesh.elem2edge(:,2) = QuadriMesh.elem2edge(:,4) + ny;
lbl = LblEgHori(1:ny,1:nx);
QuadriMesh.elem2edge(:,1) = NumEgsVert + lbl(:);
QuadriMesh.elem2edge(:,3) = QuadriMesh.elem2edge(:,1) + 1;

```

To figure out one edge belongs to which one or two elements, use the mesh info about the elem2edge. Do adjustment if the number of the second element is larger than the first element or the first one is 0.

```

CntEmsEg = zeros(QuadriMesh.NumEgs,1);
for ie=1:QuadriMesh.NumEms
    LblEg = QuadriMesh.elem2edge(ie,1:4);
    CntEmsEg(LblEg) = CntEmsEg(LblEg) + 1;
    for k=1:4
        QuadriMesh.edge2elem(LblEg(k),CntEmsEg(LblEg(k))) = ie;
    end
end

```

In order to get the area and the centroid of each element, we introduce vectors  $k1, k2, k3, k4, x1, x2, x3, x4, y1, y2, y3, y4$ , to make the code clear and then use the general formula to calculate the area and the centroid of element.

We can get trapezoid mesh, which is a special case of quadrilateral mesh, in a rectangular domain. Here, we have the code modulus,

```

function QuadriMesh = RectDom_QuadriMesh_GenTrapezoid(
xa,xb,nx,yc,yd,ny,alpha,status)

```

There are also two status, which are the same as we mentioned before. The only difference is the coordinates of nodes. Here, we use `alpha` to make perturbation.

```

QuadriMesh.node(k,2) = QuadriMesh.node(k,2) - alpha*hy; % odd nodes
QuadriMesh.node(k,2) = QuadriMesh.node(k,2) + alpha*hy; % even nodes

```

On a given rectangular domain, we can get a random quadrilateral mesh.

```
function QuadriMesh = RectDom_QuadriMesh_GenRandRect(
xa,xb,nx,yc,yd,ny,delta,status)
```

The method of getting two status are the same with generating a logically rectangular quadrilateral mesh. The only difference is the coordinates of nodes. Here, we used `delta` and `randn` to make a random perturbation.

```
X(k) = X(k) + delta * hx * sin(pi*(i/nx)) * randn();
Y(k) = Y(k) + delta * hy * sin(pi*(j/ny)) * randn();
```

Besides on a rectangular domain, we can get quadrilateral mesh on a circular sector domain. The code modulus,

```
function QuadriMesh = CircSecDom_QuadriMesh_GenUnfm(
ra,rb,nr,thetac,thetad,ntheta,status)
```

There are three status. Status 1 includes number of nodes/elements, node coordinates, element vs its nodes. Here, we calculate the node coordinates in polar-coordinate system. `nr` is the number of partitions in radial direction, `ntheta` is the number of partitions in the angular direction. Other ways of labelling are the same with in rectangular domain.

```
QuadriMesh.NumNds, QuadriMesh.NumEms, QuadriMesh.node, QuadriMesh.elem
```

Status 2 includes edge vs its vertices, element vs its all edges, edge vs its one or two neighboring elements. The ways of getting these information are same as generating QuadriMesh on rectangular domain.

```
QuadriMesh.edge, QuadriMesh.elem2edge, QuadriMesh.edge2elem, QuadriMesh.elem,
QuadriMesh.area, QuadriMesh.EmCntr.
```

To label the two vertices of each edge, divide edges into two groups, angular edges and radial edges.

```
% Angular edges
lbl = LblNd(1:ntheta, 1:nr+1);
QuadriMesh.edge(1:NumEgsVert,1) = lbl(:);
QuadriMesh.edge(1:NumEgsVert,2) = QuadriMesh.edge(1:NumEgsVert,1) + 1;
% Radial edges
lbl = LblNd(1:ntheta+1, 1:nr);
QuadriMesh.edge(NumEgsVert+1:end,1) = lbl(:);
QuadriMesh.edge(NumEgsVert+1:end,2) = ...
QuadriMesh.edge(NumEgsVert+1:end,1) + (ntheta+1);
```

To label the four edges of each quadrilateral, divide edges into odd and even ones.

```

QuadriMesh.elem2edge(:,4) = lbl(:);
QuadriMesh.elem2edge(:,2) = QuadriMesh.elem2edge(:,4) + ntheta;
QuadriMesh.elem2edge(:,1) = NumEgsVert + lbl(:);
QuadriMesh.elem2edge(:,3) = QuadriMesh.elem2edge(:,1) + 1;

```

To figure out each edge belongs to which one or two elements, use the mesh info about the elem2edge. Do adjustment if the number of the second element is larger than the first element or the first one is 0.

Status 3 marks the boundary edges. ??

On a circular sector domain, we can make perturbation to the quadrilateral mesh and then change the coordinates of nodes. Here, we have the code

```

function QuadriMesh = CircSecDom_QuadriMesh_GenTrapezoid(
ra,rb,nr,thetac,thetad,ntheta,alpha,status)

```

And the only difference with the former normal QuadriMesh is the coordinates of all nodes. We multiply an arbitrary value `alpha` to pull back odd nodes and push forward even nodes. Then we get a more general mesh.

```

% Odd j-index and even i-index: Pulling back
for j=1:2:(ny+1)
    i = 2:2:nx;
    k = (i-1)*(ny+1) + j;
    X(k) = X(k) - alpha*hx;
end
% Even j-index and even i-index: Pushing forward
for j=2:2:ny
    i = 2:2:nx;
    k = (i-1)*(ny+1) + j;
    X(k) = X(k) + alpha*hx;
end

```

In practical, we can generate quadrilateral meshes on the longitude and latitude domain.

Refinement from triangular mesh. In the code,

```

function QuadriMesh = QuadriMesh_RefinedFrom_TriMesh(TriMesh)

```

we use information of nodes, edges, etc from an arbitrary triangular mesh. Find the centroid of each element,  $xc = (x1+x2+x3)/3$ ,  $yc = (y1+y2+y3)/3$ .

Then connect the centroid with the middle of three edges separately. So, each triangle has been divided into three quadrilaterals. And three more edges in each element have

been added to total number of edges. The midpoints of each edge and centroid nodes are newly added nodes.

```
QuadriMesh.NumNds = NDT + NET + NGT;
QuadriMesh.NumEms = 3*NET;
QuadriMesh.NumEgs = 3*NET + 2*NGT;
```

All nodes in the new quadrilateral mesh are separated into three groups.  
The original nodes of triangular mesh keep the same.

```
QuadriMesh.node(1:NDT,:) = TriMesh.node;
```

The second group is the centroid of original triangles.

```
QuadriMesh.node(NDT+(1:NET),1) = xc;
QuadriMesh.node(NDT+(1:NET),2) = yc;
```

The third group is the midpoints of original edges.

```
QuadriMesh.node(NDT+NET+(1:NGT),:) =
0.5*(TriMesh.node(TriMesh.edge(:,1),:)
+TriMesh.node(TriMesh.edge(:,2),:));
```

Because each original triangle is divided into three quadrilaterals, so we regard one triangle as a group. The first vertex of quadrilateral is the original vertex of triangle. The second vertex is the midpoint of triangle's edge, and it belongs to the third group of all nodes. So we need to add numbers of the first two groups of node to the TriMesh information `TriMesh.elem2edge`. The third vertex is the centroid of triangle and it belongs to the second group of node. The fourth vertex has the same position with the second vertex, but they are just different in the sequence of numbering.

There are two types of edges. The first type is the children of original edges. Since we take the midpoint of the each edge, each original edge has been divided into two edges, left and right one. So the first vertex of left child is the first vertex of original edge, the second vertex of right child is the second vertex of original edge. They share the same node as another vertex, and this shared node belongs to the third group of all nodes. The second type of edge is newly added edges in the triangle. Firstly, we number the the edge opposite to the first original triangle vertex as the first new edge, then number others in counterclockwise direction. The first vertex of the first new edge belongs to the third group of nodes, here, we use `TriMesh.elem2edge`. The second vertex is the centroid of triangle. The ways of numbering the second vertices of all new edges are the same. Therefore, we set the centroids of triangle as the new edges' second vertices. The first vertices of all new edges are the midpoints of all original edges. But they are just different in numbering sequence.

To get **Element vs edges**, we create a sparse matrix `NdNdEg`, which shows the relation between two vertices and the edge. Then use four vertices of new element to find out the four edges.

```

for ie=1:QuadriMesh.NumEms
QuadriMesh.elem2edge(ie,1) = NdNdEg(k1(ie),k2(ie));
QuadriMesh.elem2edge(ie,2) = NdNdEg(k2(ie),k3(ie));
QuadriMesh.elem2edge(ie,3) = NdNdEg(k3(ie),k4(ie));
QuadriMesh.elem2edge(ie,4) = NdNdEg(k4(ie),k1(ie));
end

```

To get Edges vs elements, we use `QuadriMesh.elem2edge`. ???

In *DarcyLite*, we have an example of getting a quadrilateral mesh with nodes, edges, elements labeled. Here is the code modulus.

```
test_QuadriMesh_RefinedFrom_TriMesh
```

And the following two figures are quadrilateral mesh and original triangular mesh when  $n = 2$ .

Refinement from quadrilateral mesh. Here is the code,

```
function QuadriMeshNew = QuadriMesh_RefineReg(QuadriMeshOld)
```

We connect the midpoints of old edges with the centroid of each old element. Therefore, in the new quadrilateral mesh, midpoints of edges and the centroid of elements are newly added nodes. Each old element is divided into four small quadrilaterals and has four new edges. Each old edge is divided into two new edges. So the number nodes, elements and edges are

```

QuadriMeshNew.NumNds = NDO + NEO + NGO;
QuadriMeshNew.NumEms = 4*NEO;
QuadriMeshNew.NumEgs = 4*NEO + 2*NGO;

```

There are three types of nodes. The first type is old nodes. The second is the midpoints of old edges. The third is the centroid of old elements.

To get the mesh information of **Element vs nodes**, the way of labelling is the same as *QuadriMesh Refined From TriMesh*. We deal with four nodes separately.

To get the mesh information of **Edge vs nodes**, the way of labelling the vertices of child of old edges is the same as *QuadriMesh Refined From TriMesh*. And so do the new four edges. The only difference is that we have to deal with another new edge in the old element. If necessary, take adjustment to make sure the label of the first vertex is smaller than the second vertex.

The ways of getting the mesh information of **Elements vs edges** and **Edges vs elements** are the same with *QuadriMesh Refined From TriMesh*, which is stated before.

In *DarcyLite*, we have an example of getting a quadrilateral mesh with nodes, edges, elements labeled. Here is the code modulus and figures of quadrilateral mesh and original quadrilateral mesh in the circular sector domain.

```
test_QuadriMesh_RefineReg
```

## 10.2.2 Solving the Darcy Equation on Quadrilateral Meshes

For the certain applications, we need to solve the Darcy equation on quadrilateral meshes. This is a nontrivial task. Existing efforts can be found in [3, 4, 22], which utilize Piola transformation (mapping) to construct  $H(\text{div})$  finite elements that are used in the mixed finite element setting. This (mapping) approach involves heavy computation. It is worthwhile to try the unmapped approach. Namely, we will construct weak Galerkin finite elements on quadrilaterals directly.

First, we could still construct the localized  $RT_{[0]}$  finite element space on a quadrilateral without using any mapping. Specifically,

$$RT_{[0]}(Q) = \text{Span}(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4),$$

where

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{w}_3 = \begin{bmatrix} X \\ 0 \end{bmatrix}, \mathbf{w}_4 = \begin{bmatrix} 0 \\ Y \end{bmatrix},$$

and  $X = x - x_c, Y = y - y_c, (x_c, y_c)$  is the element center. It is not difficult to compute the Gram matrix of this normalized basis, but generally it does not have the diagonal structure.

**JL Remark:** The Gram matrix is to be finished by ZW.

The lowest WG finite element on a quadrilateral  $Q$  will be  $(Q_0, P_0, RT_{[0]})$ , that is, we have one constant WG basis function for the element interior, and one constant WG basis function for each of the four edges. These 5 WG basis functions have discrete weak gradients in the above (unmapped)  $RT_{[0]}(Q)$  space, defined through integration by parts.

**JL Remark:** Zhuoran, Please write something for this part.

Then we could establish a WG finite element scheme for the Darcy equation on a quadrilateral mesh in a similar way as that shown in Equations (3.9)–(3.11).

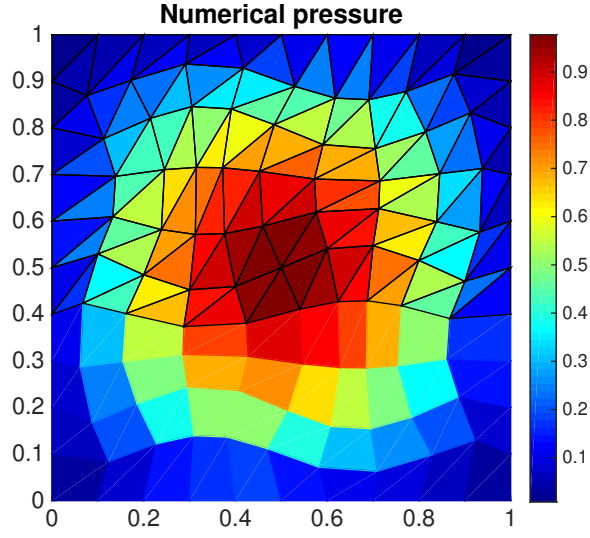
**JL Remark:** Zhuoran, Please read the paragraphs before and after Equation (3.9)–(3.11) and write similar things.

In summary, we have a lowest order weak Galerkin finite element scheme for the Darcy equation on a hybrid mesh that consists of triangles and quadrilaterals. The primal pressure unknowns are simply constants on the triangles, quadrilaterals and edges. If needed, Schur complement could be used to further eliminate the pressure unknown inside elements and keep only those on the edges. Discrete weak gradients are calculated in  $RT_0$  or  $RT_{[0]}$  respectively for triangles and quadrilaterals. Local  $L_2$  projection of  $-\mathbf{K}\nabla_{w,1}p_h$  is conducted to compute elementwise numerical velocity.

**Example ?.** An example for a hybrid/mixed mesh. See Figure ??.

The advantages of this approach are

- Lowest order (just constants);
- Rectangles as a special case of quadrilaterals;



- Mix of quadrilaterals and triangles in a mesh;
- No stabilizer needed.

The limitation is the "asymptotically parallelogram" assumption on mesh quality, although it is a widely accepted assumption. For quadrilateral meshes with lower quality, reduced convergence order should be obtained, but error analysis will be more technical, see the techniques in [?]. To relax the "asymptotical parallelogram" mesh requirement, one could use the WG finite element scheme  $(P_1, P_0, P_0^2)$  in [?], which needs a stabilizer.



# Bibliography

- [1] J. Albery, C. Carstensen, S. A. Funken, and R. Klose. Matlab implementation of the finite element method in elasticity. *Computing*, 69:239–263, 2002.
- [2] Jochen Albery, Carsten Carstensen, and Stefan Funken. Remarks around 50 lines of matlab: short finite element implementation. *Numer. Algor.*, 20:117–137, 1999.
- [3] D.N. Arnold, D. Boffi, and R.S. Falk. Approximation by quadrilateral finite elements. *Math. Comput.*, 71:909–922, 2002.
- [4] D.N. Arnold, D. Boffi, and R.S. Falk. Quadrilateral  $h(\text{div})$  finite elements. *SIAM J. Numer. Anal.*, 42:2429–2451, 2005.
- [5] C. Bahriawati and Carsten Carstensen. Three matlab implementations of the lower-order raviart-thomas mfem with a posteriori error control. *Comput. Meth. Appl. Math.*, 5:333–361, 2005.
- [6] P. Bastian and B. Riviere. Superconvergence and  $h(\text{div})$  projection for discontinuous galerkin methods. *Int. J. Numer. Meth. Fluids*, 42:1043–1057, 2003.
- [7] Susanne Brenner and Li yeng Sung. Linear finite element methods for planar linear elasticity. *Math. Comput.*, 59:321–338, 1992.
- [8] C. Carstensen and M. Schedensack. Medius analysis and comparison results for first-order finite element methods in linear elasticity. *IMA J. Numer. Anal.*, 35:1591–1621, 2015.
- [9] Long Chen. ifem: an integrated finite element methods package in matlab. *Technical Report, University of California at Irvine*, 2009.
- [10] Z. Chen, G. Huan, and Y. Ma. *Computational methods for multiphase flows in porous media*. SIAM, 2006.
- [11] B. Cockburn, J. Gopalakrishnan, and H. Wang. Locally conservative fluxes for the continuous galerkin method. *SIAM J. Numer. Anal.*, 45:1742–1770, 2007.

- [12] R. Codina. comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Comput. Meth. Appl. Mech. Engrg.*, 156:185–210, 1998.
- [13] Louis Durlofsky. Accuracy of mixed and control volume finite element approximations to darcy velocity and related quantities. *Water Resour. Res.*, 30:965–973, 1994.
- [14] F.Brezzi and M.Fortin. *Mixed and hybrid finite element methods*. Springer-Verlag, 1991.
- [15] Victor Ginting, Guang Lin, and Jiangguo Liu. On application of the weak galerkin finite element method to a two-phase model for subsurface flow. *J. Sci. Comput.*, 66:225–239, 2016.
- [16] Bishnu P. Lamichhane. A mixed finite element method for nearly incompressible elasticity and stokes equations using primal and dual meshes with quadrilateral and hexahedral grids. *J. Comput. Appl. Math.*, 260:356–363., 2014.
- [17] Qiaoluan Li and Junping Wang. Weak galerkin finite element methods for parabolic equations. *Numer. Meth. PDEs*, 29:2004–2024, 2013.
- [18] G. Lin, J. Liu, L. Mu, and X. Ye. Weak galerkin finite element methdos for darcy flow: Anistropy and heterogeneity. *J. Comput. Phys.*, 276:422–437, 2014.
- [19] Guang Lin, Jiangguo Liu, and Farrah Sadre-Marandi. A comparative study on the weak galerkin, discontinuous galerkin, and mixed finite element methods. *J. Comput. Appl. Math.*, 273:346–362, 2015.
- [20] Jiangguo Liu and R.Cali. A note on the approximation properties of the locally divergence-free finite elements. *Int. J. Numer. Anal. Model.*, 5:693–703, 2008.
- [21] Jiangguo Liu, Farrah Sadre-Marandi, and Zhuoran Wang. Darcylite: A matlab toolbox for darcy flow computation. *Procedia Computer Science*, 80:1301–1312, 2016.
- [22] M.F.Wheeler and I.Yotov. A multipoint flux mixed finite element method. *SIAM J. Numer. Anal.*, 44:2082–2106, 2006.
- [23] Lin Mu, Junping Wang, Yanqiu Wang, and Xiu Ye. A computational study of the weak galerkin method for second-order elliptic equations. *Numer. Algor.*, 63:753–777, 2012.
- [24] Lin Mu, Junping Wang, and Xiu Ye. Weak galerkin finite element methdos on polytopal meshes. *Int. J. Numer. Anal. Model.*, 12:31–53, 2015.

- [25] Per-Olof Persson and Gilbert Strang. A simple mesh generator in matlab. *SIAM Review*, 46:329–345, 2004.
- [26] Phillip J. Phillips and Mary Wheeler. A coupling of mixed with continuous galerkin finite element methods for poroelasticity i: the continuous in time case. *Comput. Geosci.*, 11:131–144, 2007.
- [27] Phillip J. Phillips and Mary Wheeler. A coupling of mixed with continuous galerkin finite element methods for poroelasticity ii: the-discrete-in-time case. *Comput. Geosci.*, 11:145–158, 2007.
- [28] S.C.Brenner and L.Ridgway Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics*. Springer-Verlag, New York, third edition, 2008.
- [29] Shuyu Sun and Jiangguo Liu. A locally conservative finite element method based on piecewise constant enrichment of the continuous galerkin method. *SIAM J. Sci. Comput.*, 31:2528–2548, 2009.
- [30] Simon Tavener and Michael Mikucki. Sensai: A matlab package for sensitivity analysis. <http://www.math.colostate.edu/~tavener/FEScUE/SENSAI/sensai.shtml>.
- [31] Hong Wang, Helge Dale, Richard Ewing, Magne Espedal, Robert Sharpley, and Shushuang Man. An ellam scheme for advection-diffusion equations in two dimensions. *SIAM J. Sci. Comput.*, 20:2160–2194, 1999.
- [32] Junping Wang and Xiu Ye. A weak galerkin finite element method for second order elliptic problems. *J. Comput. Appl. Math.*, 241:103–115, 2013.