deal.II Implementation of a Weak Galerkin Finite Element Solver for Darcy Flow *

Zhuoran Wang[1], Graham Harper[1], Patrick O'Leary[2], Jiangguo Liu[1], and Simon Tavener[1]

[1] Colorado State University, Fort Collins, CO 80523, USA, {wangz,harper,liu,tavener}@math.colostate.edu
[2] Kitware, Inc., Santa Fe, NM 87507, USA, patrick.oleary@kitware.com

Abstract. This paper presents a weak Galerkin (WG) finite element solver for Darcy flow and its implementation on the deal.II platform. The solver works for quadrilateral and hexahedral meshes in a unified way. It approximates pressure by Q-type degree $k(\geq 0)$ polynomials separately defined in element interiors and on edges/faces. Numerical velocity is obtained in the unmapped Raviart-Thomas space $RT_{[k]}$ via postprocessing based on the novel concepts of discrete weak gradients. The solver is locally mass-conservative and produces continuous normal fluxes. The implementation in deal.II allows polynomial degrees up to 5. Numerical experiments show that our new WG solver performs better than the classical mixed finite element methods.

Keywords: Darcy flow · deal.II · finite element methods · hexahedral meshes · quadrilateral meshes · weak Galerkin

1 Introduction

The Darcy equation, although simple, plays an important role for modeling flow in porous media. The equation usually takes the following form

$$\begin{cases} \nabla \cdot (-\mathbf{K}\nabla p) + c\,p = f, \quad \mathbf{x} \in \Omega, \\ p|_{\Gamma^D} = p_D, \quad ((-\mathbf{K}\nabla p)\cdot \mathbf{n})|_{\Gamma^N} = u_N, \end{cases} \tag{1}$$

where Ω is a 2-dim or 3-dim bounded domain, p is the unknown pressure, $\mathbf{K}$ is a conductivity matrix that is uniformly symmetric positive definite (SPD), c is a known function, f is a known source term, p_D is a Dirichlet boundary condition, u_N is a Neumann boundary condition, and $\mathbf{n}$ is the outward unit normal vector on $\partial\Omega$, which has a nonoverlapping decomposition $\Gamma^D \cup \Gamma^N$.

The elliptic boundary value problem (1) can be solved by many types of finite element methods. But in the context of Darcy flow, *local mass conservation* and *normal flux continuity* are two most important properties to be respected by finite element solvers.

* Harper, Liu, and Wang were partially supported by US National Science Foundation grant DMS-1819252. We thank Dr. Wolfgang Bangerth for the computing resources.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

- The continuous Galerkin (CG) methods [5] use the least degrees of freedom but do not possess these two properties and hence cannot be used directly. Several post-processing procedures have been developed [7, 8].
- Discontinuous Galerkin (DG) methods are locally conservative by design and gain normal flux continuity after post-processing [4].
- The enhanced Galerkin (EG) methods [19] possess both properties but need to handle some minor issues in implementation.
- The mixed finite element methods (MFEMs) [2, 6] have both properties by design but result in indefinite discrete linear systems, for which hybridization needs to be employed to convert them into definite linear systems.
- The weak Galerkin (WG) methods [11, 13, 15–17, 20] have both properties and result in SPD linear systems that are easier to solve.

In this paper, we investigate efficient implementation of WG Darcy solvers in `deal.II`, a popular finite element package [3], with the intention to make WG finite element methods practically useful for large-scale scientific computation.

2 A WG Finite Element Solver for Darcy Flow

WG solvers can be developed for Darcy flow on simplicial, quadrilateral or hexahedral, and more general polygonal or polyhedral meshes. These finite element schemes may or may not contain a stabilization term, depending on choices of the approximating polynomials for pressure in element interiors and on edges/faces. Through integration by parts, these polynomial basis functions are used for computing discrete weak gradients, which are used to approximate the classical gradient in the variational form for the Darcy equation. Discrete weak gradients can be established in a general vector polynomial space [18] or a specific one like the Raviart-Thomas space [11, 17] that has desired approximation properties.

This paper focuses on quadrilateral and hexahedral meshes, in which faces are or very close to being flat. We use $Q_k(k \geq 0)$-type polynomials in element interiors and on edges/faces for approximating the primal variable pressure. Their discrete weak gradients are established in local unmapped Raviart-Thomas $RT_{[k]}(k \geq 0)$ spaces, for which we do not use the Piola transformation. We use the same form of polynomials as that for rectangles and bricks in the classical MFEMs [6].

To illustrate these new ideas, we consider a quadrilateral E centered at (x_c, y_c). We define the local unmapped Raviart-Thomas space $RT_{[0]}(E)$ as

$$RT_{[0]}(E) = \text{Span}(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4), \tag{2}$$

where

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{w}_3 = \begin{bmatrix} X \\ 0 \end{bmatrix}, \quad \mathbf{w}_4 = \begin{bmatrix} 0 \\ Y \end{bmatrix}, \tag{3}$$

and $X = x - x_c$, $Y = y - y_c$ are the normalized coordinates.

Now we introduce a new concept of 5 discrete weak functions $\phi_i(0 \leq i \leq 4)$.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

– ϕ_0 is for element interior: It takes value 1 in the interior E° but 0 on the boundary E^∂ (all 4 edges);
– $\phi_1, \phi_3, \phi_3, \phi_4$ are for the four sides respectively: $\phi_i (1 \le i \le 4)$ takes value 1 on the i-th edge but 0 on all other three edges and in the interior.

Any such function ϕ has two independent parts: ϕ° is defined in E°, whereas ϕ^∂ is defined on E^∂, together written as $\phi = \{\phi^\circ, \phi^\partial\}$. Its discrete weak gradient $\nabla_w \phi$ can be specified in $RT_{[0]}(E)$ via integration by parts [20]:

$$\int_E (\nabla_w \phi) \cdot \mathbf{w} = \int_{E^\partial} \phi^\partial (\mathbf{w} \cdot \mathbf{n}) - \int_{E^\circ} \phi^\circ (\nabla \cdot \mathbf{w}), \qquad \forall \mathbf{w} \in RT_{[0]}(E). \tag{4}$$

This attributes to solving a size-4 SPD linear system. Note that

(i) For a quadrilateral, ϕ° or ϕ^∂ each can also be a degree $k \ge 1$ polynomial and the discrete weak gradient $\nabla_w \phi$ is then established in the local unmapped Raviart-Thomas space $RT_{[k]} (k \ge 1)$.
(ii) For a hexahedron with nonflat faces, we can use the averaged normal vectors in (4). The Jacobian determinant is still used in computation of the integrals.

For a rectangle $E = [x_1, x_2] \times [y_1, y_2]$ ($\Delta x = x_2 - x_1, \Delta y = y_2 - y_1$), we have

$$\begin{cases} \nabla_w \phi_0 = \quad 0\mathbf{w}_1 \; + \quad 0\mathbf{w}_2 \; + \frac{-12}{(\Delta x)^2}\mathbf{w}_3 \; + \frac{-12}{(\Delta y)^2}\mathbf{w}_4, \\ \nabla_w \phi_1 = \frac{-1}{\Delta x}\mathbf{w}_1 \; + \quad 0\mathbf{w}_2 \; + \frac{6}{(\Delta x)^2}\mathbf{w}_3 \; + \quad 0\mathbf{w}_4, \\ \nabla_w \phi_2 = \frac{1}{\Delta x}\mathbf{w}_1 \; + \quad 0\mathbf{w}_2 \; + \frac{6}{(\Delta x)^2}\mathbf{w}_3 \; + \quad 0\mathbf{w}_4, \\ \nabla_w \phi_3 = \quad 0\mathbf{w}_1 \; + \frac{-1}{\Delta y}\mathbf{w}_2 \; + \quad 0\mathbf{w}_3 \; + \frac{6}{(\Delta y)^2}\mathbf{w}_4, \\ \nabla_w \phi_4 = \quad 0\mathbf{w}_1 \; + \frac{1}{\Delta y}\mathbf{w}_2 \; + \quad 0\mathbf{w}_3 \; + \frac{6}{(\Delta y)^2}\mathbf{w}_4. \end{cases} \tag{5}$$

Let $\mathcal{E}_h$ be a shape-regular quadrilateral mesh. Let Γ_h^D be the set of all edges on the Dirichlet boundary Γ^D and Γ_h^N be the set of all edges on the Neumann boundary Γ^N. Let S_h be the space of discrete shape functions on $\mathcal{E}_h$ that are degree k polynomials in element interiors and also degree k polynomials on edges. Let S_h^0 be the subspace of functions in S_h that vanish on Γ_h^D. For (1), we seek $p_h = \{p_h^\circ, p_h^\partial\} \in S_h$ such that $p_h^\partial|_{\Gamma_h^D} = Q_h^\partial(p_D)$ (the L^2-projection of Dirichlet boundary data into the space of degree k polynomials on Γ_h^D) and

$$\mathcal{A}_h(p_h, q) = \mathcal{F}(q), \qquad \forall q = \{q^\circ, q^\partial\} \in S_h^0, \tag{6}$$

where

$$\mathcal{A}_h(p_h, q) = \sum_{E \in \mathcal{E}_h} \int_E \mathbf{K} \nabla_w p_h \cdot \nabla_w q + \sum_{E \in \mathcal{E}_h} \int_E c\, p\, q, \tag{7}$$

$$\mathcal{F}(q) = \sum_{E \in \mathcal{E}_h} \int_E f q^\circ - \sum_{\gamma \in \Gamma_h^N} \int_\gamma u_N q^\partial. \tag{8}$$

This results in a symmetric positive-definite discrete linear system [17].

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

Note $\nabla_w p_h$ is in the local Raviart-Thomas space, but $-\mathbf{K}\nabla_w p_h$ may not be. A local L^2-projection $\mathbf{Q}_h$ is needed [11, 13, 17] to get it back into the RT space:

$$\mathbf{u}_h = \mathbf{Q}_h(-\mathbf{K}\nabla_w p_h). \tag{9}$$

This is the numerical Darcy velocity for subsequent applications, e.g., transport simulations. Clearly, this process is readily parallelizable for large-scale computation. This numerical velocity is locally mass-conservative and the corresponding normal flux is continuous across edges or faces, as proved in [11, 17].

As shown in [17], this Darcy solver is easy to be implemented and results in a symmetric positive-definite system that can be easily solved by a conjugate-gradient type linear solver. The WG methodology has connections to but is indeed different than the classical mixed finite element methods, especially the hybridized MFEMs [13, 14].

3 `deal.II` Implementation of WG Solver for Darcy Flow

`deal.II` is a popular C++ finite element package [3]. It uses quadrilateral and hexahedral meshes instead of simplicial meshes. The former may involve less degrees of freedom than the latter. The resulting linear systems may have smaller sizes, although the setup time for these linear systems may be longer. The setup time is spent on bilinear/trilinear mappings from the reference square/cube to general quadrilaterals/hexahedra and computation of various integrals.

3.1 Quadrilateral and Hexahedral Meshes

`deal.II` handles meshes by the **GridGenerator** class. All mesh information, such as the number of active cells, degrees of freedom, are stored in this class. For any integer $k \geq 0$, our WG$(Q_k, Q_k; RT_{[k]})$ solver is locally mass-conservative and produces continuous normal fluxes regardless of the quality of quadrilateral and hexahedral meshes. In order to obtain the desired order k convergence rate in pressure, velocity, and normal fluxes, we require meshes to be asymptotically parallelogram or parallelopiped [11, 17].

3.2 Finite Element Spaces

The WG$(Q_k, Q_k; RT_{[k]})$ solver involves three finite element spaces. The first two spaces are for the pressure unknowns, the third one (RT space) is used for discrete weak gradients and numerical velocity. In `deal.II` implementation, the first two are combined as

```
FESystem<dim> fe;
```

The third one (with `dim` being 2 or 3) is

```
FE_RaviartThomas<dim> fe_rt;
```

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

Raviart-Thomas Spaces for Discrete Weak Gradients and Velocity. WG allows use of unmapped RT spaces on quadrilaterals and hexahedra [11, 17]. These spaces use the same polynomials for shape functions as those in the classical RT spaces for 2-dim or 3-dim rectangles [6]. They are respectively,

$$RT_{[k]}(E) = Q_{k+1,k} \times Q_{k,k+1}, \tag{10}$$

$$RT_{[k]}(E) = Q_{k+1,k,k} \times Q_{k,k+1,k} \times Q_{k,k,k+1}. \tag{11}$$

In `deal.II`, we use `degree` for k in equation (10) or (11) and have

```
fe_rt(degree);
```

Two Separate Polynomial Spaces for Pressure. Note that for the WG$(Q_k, Q_k; RT_{[k]})$ finite element solver for Darcy flow, the pressure is approximated separately in element interiors by Q_k-type polynomials and on edges/faces by Q_k-type polynomials also. Note that the 2nd group of Q_k-type polynomials are defined locally on each edge/face. For the `deal.II` implementation, we have

```
fe(FE_DGQ<dim>(degree), 1, FE_FaceQ<dim>(degree), 1);
```

where `degree` is k, that is, the degree of the polynomials, "1" means these two groups of pressure unknowns are just scalars. Note that

- **FE_DGQ** is a finite element class in `deal.II` that has no continuity across faces or vertices, i.e., every shape function lives exactly in one cell. So we use it to approximate the pressure in element interiors.
- **FE_FaceQ** is a finite element class that is defined only on edges/faces.

However, these two different finite element spaces are combined into one finite element system, we split these shape functions as

```
const FEValuesExtractors::Scalar interior(0);
const FEValuesExtractors::Scalar face(1);
```

Here "0" corresponds to the 1st finite element class **FE_DGQ** for the interior pressure; "1" corresponds to the 2nd finite element class **FE_FaceQ** for the face pressure. Later on, we will just use **fe_values[interior].value** and **fe_values[face].value** for assembling the element-level matrices.

3.3 Gaussian Quadratures

Finite element computation involves various types of integrals, which are discretized via quadratures, e.g., Gaussian quadratures. For example, we consider

$$\int_E f \approx \sum_{k=1}^{K} w_k f(x_k, y_k) J_k, \tag{12}$$

where K is the number of quadrature points, (x_k, y_k) is the k-th quadrature point, J_k is the corresponding Jacobian determinant, and w_k is the weight. In `deal.II`, this is handled by the **Quadrature** class. In particular, the Jacobian determinant value and weight for each quadrature point are bundled together as

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

```
fe_values.JxW(q_k);
```

where q_k is the k-th quadrature point.

3.4 Linear Solvers

`deal.II` provides a variety of linear solvers that are inherited from `PETSc`. The global discrete linear systems obtained from the weak Galerkin finite element discretization of the Darcy equation are symmetric positive-definite. Thus we can choose a conjugate-gradient type linear solver for them.

3.5 Graphics Output

In our WG$(Q_k, Q_k; RT_{[k]})$ solver for Darcy flow, the scalar pressures are defined separately in element interiors and on edges/faces of a mesh. These values are output separately in `deal.II`. The interior pressures are handled by **DataOut**, whereas the face pressures are handled by **DataOutFace**. Specifically,

```
data_out.build_patches(fe.degree);
data_out_face.build_patches(fe.degree);
```

are used to subdivide each cell into smaller patches, which provide better visualization if we use higher degree polynomials. The post-processed data are saved as `vtk` files for later visualization in `VisIt`.

4 Code Excerpts with Comments

This section provides some code excerpts with comments. More details can be found in `deal.II` tutorial Step-61 (subject to minor changes) [1].

4.1 Construction of Finite Element Spaces

Note that `FE_RaviartThomas` is a Raviart-Thomas space for vector-valued functions, `FESystem` defines finite element spaces in the interiors and on edges/faces. Shown below is the code for the lowest order WG finite elements.

```
88  FE_RaviartThomas<dim> fe_rt;
89  DoFHandler<dim> dof_handler_rt;
90  FESystem<dim> fe;
91  DoFHandler<dim> dof_handler;
```

```
227 fe_rt (0);
228 dof_handler_rt (triangulation);
229 fe (FE_DGQ<dim>(0), 1, FE_FaceQ<dim>(0), 1);
230 dof_handler (triangulation);
```

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

4.2 System Setup

The following piece distributes degrees of freedom for finite element spaces.

```
dof_handler_rt.distribute_dofs (fe_rt);
dof_handler.distribute_dofs (fe);
```

The following piece sets up matrices and vectors in the system.

```
DynamicSparsityPattern dsp(dof_handler.n_dofs());
DoFTools::make_sparsity_pattern(dof_handler, dsp, constraints);
sparsity_pattern.copy_from(dsp);
system_matrix.reinit(sparsity_pattern);
solution.reinit(dof_handler.n_dofs());
system_rhs.reinit(dof_handler.n_dofs());
```

4.3 System Assembly

The following piece uses extractors to extract components of finite element shape functions.

```
const FEValuesExtractors::Vector velocities (0);
const FEValuesExtractors::Scalar interior (0);
const FEValuesExtractors::Scalar face (1);
```

The following pieces calculates the Gram matrix for the RT space.

```
for (unsigned int q = 0; q < n_q_points_rt; ++q) {
  for (unsigned int i = 0; i < dofs_per_cell_rt; ++i) {
    const Tensor<1,dim> phi_i_u =
          fe_values_rt[velocities].value(i,q);
    for (unsigned int j = 0; j < dofs_per_cell_rt; ++j) {
      const Tensor<1,dim>  phi_j_u =
            fe_values_rt[velocities].value (j, q);
            cell_matrix_rt(i,j) += phi_i_u * phi_j_u
                                 * fe_values_rt.JxW(q);
} } }
```

The following piece handles construction of WG local matrices.

```
for (unsigned int q = 0; q < n_q_points_rt; ++q) {
  for (unsigned int i = 0; i<dofs_per_cell; ++i) {
    for (unsigned int j = 0; j<dofs_per_cell; ++j) {
     for (unsigned int k = 0; k<dofs_per_cell_rt; ++k) {
```

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

```
        const Tensor<1,dim> phi_k_u =
              fe_values_rt[velocities].value(k,q);
        for (unsigned int l = 0; l < dofs_per_cell_rt; ++l) {
          const Tensor<1,dim> phi_l_u =
                fe_values_rt[velocities].value(l,q);
                local_matrix(i,j) += coefficient_values[q] *
                    cell_matrix_C[i][k] * cell_matrix_C[j][l] *
                    phi_k_u * phi_l_u * fe_values_rt.JxW(q);
} } } } }
```

The following piece calculates the local right-hand side.

```
for (unsigned int q = 0; q < n_q_points; ++q) {
  for (unsigned int i = 0; i < dofs_per_cell; ++i) {
   cell_rhs(i) += (fe_values[interior].value(i, q) *
     right_hand_side.value(fe_values.quadrature_point(q)) *
     fe_values.JxW(q));
} }
```

The following piece distributes entries of local matrices into the system matrix and also incorporates the local right-hand side into the system right-hand side.

```
cell->get_dof_indices(local_dof_indices);
constraints.distribute_local_to_global(
            local_matrix, cell_rhs, local_dof_indices,
            system_matrix, system_rhs);
```

5 Numerical Experiments

This section presents three numerical examples (equation (1) with $c = 0$) to demonstrate accuracy and robustness of our novel WG solver for Darcy flow.

Example 1 (A smooth example for convergence rates). Here we have domain $\Omega = (0,1)^2$, conductivity $\mathbf{K} = \mathbf{I}_2$, and a known solution for the pressure:

$$p(x,y) = \sin(\pi x)\sin(\pi y).$$

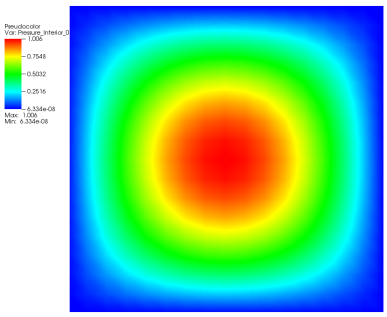
A homogeneous Dirichlet boundary condition is posed on the entire boundary.

The WG$(Q_k, Q_k; RT_{[k]})$ solver is tested on Example 1 for $k = 0, 1, 2$ on a sequence of uniform rectangular meshes. As shown in Table 1, the solver exhibits order k convergence rates for the L^2-norms of the errors in the interior pressure, velocity, and normal flux. Shown in Figure 1 are the profiles of the numerical pressure obtained from applying the WG$(Q_1, Q_1; RT_{[1]})$ solver. In the right panel, the edge pressures are plotted as grey line segments. The graphical

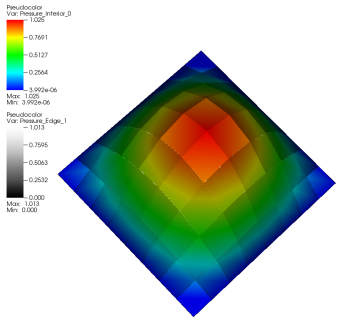
ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

Table 1. Ex.1: Convergence rates of WG($Q_k, Q_k; RT_{[k]}$) solver on rectangular meshes

$1/h$	$\|p - p_h^\circ\|$	Rate	$\|\mathbf{u} - \mathbf{u}_h\|$	Rate	$\|\mathbf{u}\cdot\mathbf{n} - \mathbf{u}_h\cdot\mathbf{n}\|$	Rate
WG($Q_0, Q_0; RT_{[0]}$)						
4	1.5870E-01	—	5.1289E-01	—	7.0500E-01	—
8	7.9980E-02	0.988	2.5309E-01	1.018	3.5523E-01	0.988
16	4.0058E-02	0.997	1.2608E-01	1.005	1.7796E-01	0.997
32	2.0037E-02	0.999	6.2977E-02	1.001	8.9020E-02	0.999
64	1.0020E-02	0.999	3.1481E-02	1.000	4.4516E-02	0.999
128	5.0099E-03	1.000	1.5740E-02	1.000	2.2258E-02	1.000
WG($Q_1, Q_1; RT_{[1]}$)						
4	1.6130E-02	—	5.0989E-02	—	7.1588E-02	—
8	4.0560E-03	1.991	1.2762E-02	1.998	1.8016E-02	1.990
16	1.0155E-03	1.997	3.1915E-03	1.999	4.5113E-03	1.997
32	2.5396E-04	1.999	7.9792E-04	1.999	1.1283E-03	1.999
64	6.3496E-05	1.999	1.9948E-04	1.999	2.8210E-04	1.999
128	1.5874E-05	2.000	4.9871E-05	1.999	7.0528E-05	1.999
WG($Q_2, Q_2; RT_{[2]}$)						
4	1.0719E-03	—	3.3764E-03	—	4.7589E-03	—
8	1.3465E-04	2.992	4.2331E-04	2.995	5.9814E-04	2.992
16	1.6852E-05	2.998	5.2952E-05	2.998	7.4870E-05	2.998
32	2.1072E-06	2.999	6.6203E-06	2.999	9.3620E-06	2.999
64	2.6342E-07	2.999	8.2757E-07	2.999	1.1703E-06	2.999
128	3.2928E-08	2.999	1.0344E-07	2.999	1.46298E-07	2.999



Element interior pressure for $h = \frac{1}{16}$



Interior/edge pressure 3d view ($h = \frac{1}{8}$)

Fig. 1. Ex.1: Numerical pressure by WG($Q_1, Q_1; RT_{[1]}$) solver on rectangular meshes

results in both panels demonstrate nice monotonicity in the numerical pressure produced by our WG solver.

Example 2 (Heterogeneous permeability). The permeability profile is adopted from [9]. We consider a simple Darcy flow problem on the unit square. Dirichlet boundary conditions are posed on the left and right sides: $p = 1$ for $x = 0$; and $p = 0$ for $x = 1$. The other two sides have a homogeneous Neumann (no-flow) boundary condition. The problem was also tested using `Matlab` in [17].

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

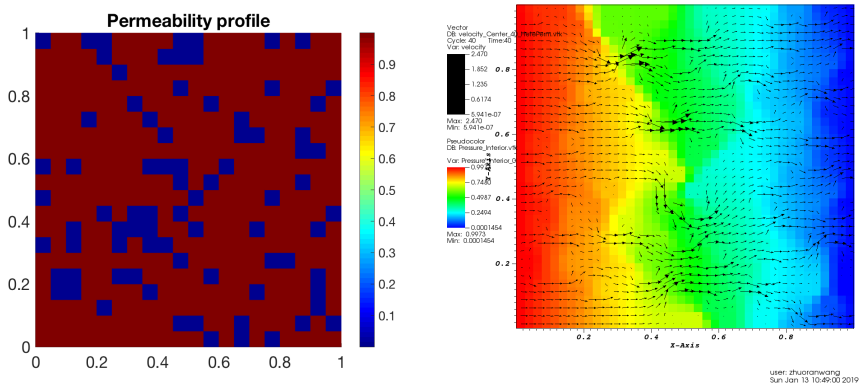


Fig. 2. Example 2 (Heterogeneous permeability): Numerical pressure and velocity

Table 2. Example 2: Comparison between WG and MFEM solvers

tol $= 10^{-9}$	WG			MFEM		
Mesh	$p_{\min}$	$p$max	Runtime	$p_{\min}$	$p$max	Runtime
20×20	1.21321E-4	0.995113	0.857s	1.21320E-4	0.995113	1.410s
40×40	1.45401E-4	0.997289	6.759s	1.45401E-4	0.997289	13.833s
80×80	8.73042E-5	0.998587	59.070s	8.73043E-5	0.998587	103.141s
160×160	4.59350E-5	0.999281	607.556s	4.59345E-5	0.999281	877.648s

Shown in Figure 2 right panel are the numerical pressure and velocity profiles obtained from apply our WG$(Q_0, Q_0; RT_0)$ solver on a uniform 40×40 rectangular mesh. Clearly, the elementwise numerical pressure stays between 0 and 1, the pressure profile demonstrates monotonicity from left to right, and the velocity profile reveals the low-permeability regions and channels for fast flow.

Example 2 was also solved by a mixed finite element solver built in `deal.II` that is based on Schur complement (See `deal.II` tutorial Step-20). We compare the lowest order WG solver ($k = 0$) with the lowest order MFEM solver on a sequence of rectangular meshes on a Toshiba laptop. The tolerance for linear solvers is 10^{-9}. Table 2 shows that the WG solver produces very close results with significantly less runtime.

Example 3 (Permeability profile in SPE10 Model 2). SPE10 was developed as a benchmark for upscaling methods, but the 2nd dataset is becoming a popular testcase for comparing different numerical methods. The dataset is a 3-dim geo-statistical realization from the Jurassic Upper Brent formations [12]. The model has geometric dimensions 1200 (ft) $\times$ 2200 (ft) $\times$ 170 (ft). The dataset is provided on a $60 \times 220 \times 85$ Cartesian grid, in which each block has a size 20(ft) $\times$ 10(ft) $\times$ 2(ft). The top 70 ft (35 layers) are for the shallow-marine Tarbert formation, the bottom 100 ft (50 layers) are for the fluvial Ness formation. The SPE10 model is structurally simple but highly heterogeneous in porosity and permeability. It poses significant challenges to numerical simulators.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

The SPE 10 dataset is publicly available at `http://www.spe.org/web/csp/`. The original data assume the z-axis pointing downwards but use a right-hand coordinate system. A conversion of ordering in blocks is needed for the original data items. We use the code in `Matlab Reservoir Simulation Toolbox` (MRST) [12] to acquire the needed data.

In this paper, we focus on the Darcy flow part. We use the original permeability data and consider a flow problem. Dirichlet boundary conditions are posed on two boundary faces: $p = 1$ for $y = 0$; and $p = 0$ for $y = 1200$. All other four boundary faces have a homogeneous Neumann (no-flow) boundary condition.

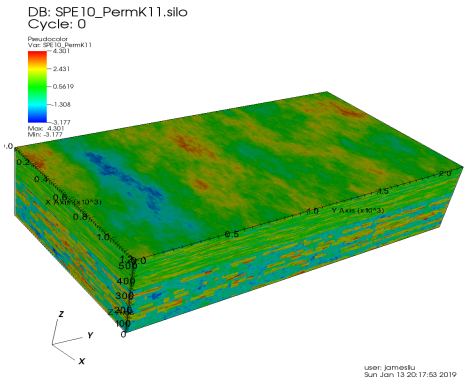
We test the WG solver on three meshes (coarse, medium, fine). For better visualization, we tripled the z dimension.

(i) A **coarse mesh** with $12 \times 44 \times 17$ partitions. For the WG$(Q_0, Q_0; RT_{[0]})$ solver, there are $8,976$ pressure degrees of freedom (DOFs) for element interiors; $28,408$ pressure DOFs for all faces, and $37,384$ total DOFs. The local $RT_{[0]}$ spaces are used to compute the discrete weak gradients of the pressure basis functions, but they do not constitute any DOFs.

(ii) A **medium mesh** with $30 \times 110 \times 85$ partitions. We use WG$(Q_0, Q_0; RT_{[0]})$ again. There are $280,500$ DOFs for the pressure in element interiors, $856,700$ DOFs for all faces, and totally $1,137,200$ (about 1M) DOFs.

(iii) A **fine mesh** with $60 \times 220 \times 85$ partitions, which is the same as the original gridblock. Again WG$(Q_0, Q_0; RT_{[0]})$ is used. There are $1,122,000$ interior DOFs; $3,403,000$ face DOFs; and a total $4,525,000$ (about 4M) DOFs.

As shown in Figure 4, the coarse mesh is too coarse to reveal the reservoir geological features. The medium-mesh result is good enough to reflect the channel features of the fluvial Ness formation. The fine-mesh result is smoother and exposes further details about the heterogeneity, but requires expensive computation. Tables 3 and 4 (results on a server with 40 Intel CPUs) together demonstrate that our new WG solver is more efficient than the classical MFEM.



(a) Permeability $K_{11} = K_{22}$ (a) Permeability K_{33}

Fig. 3. Example 3 (SPE10 Model 2): Permeability profiles on log_{10} scales

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

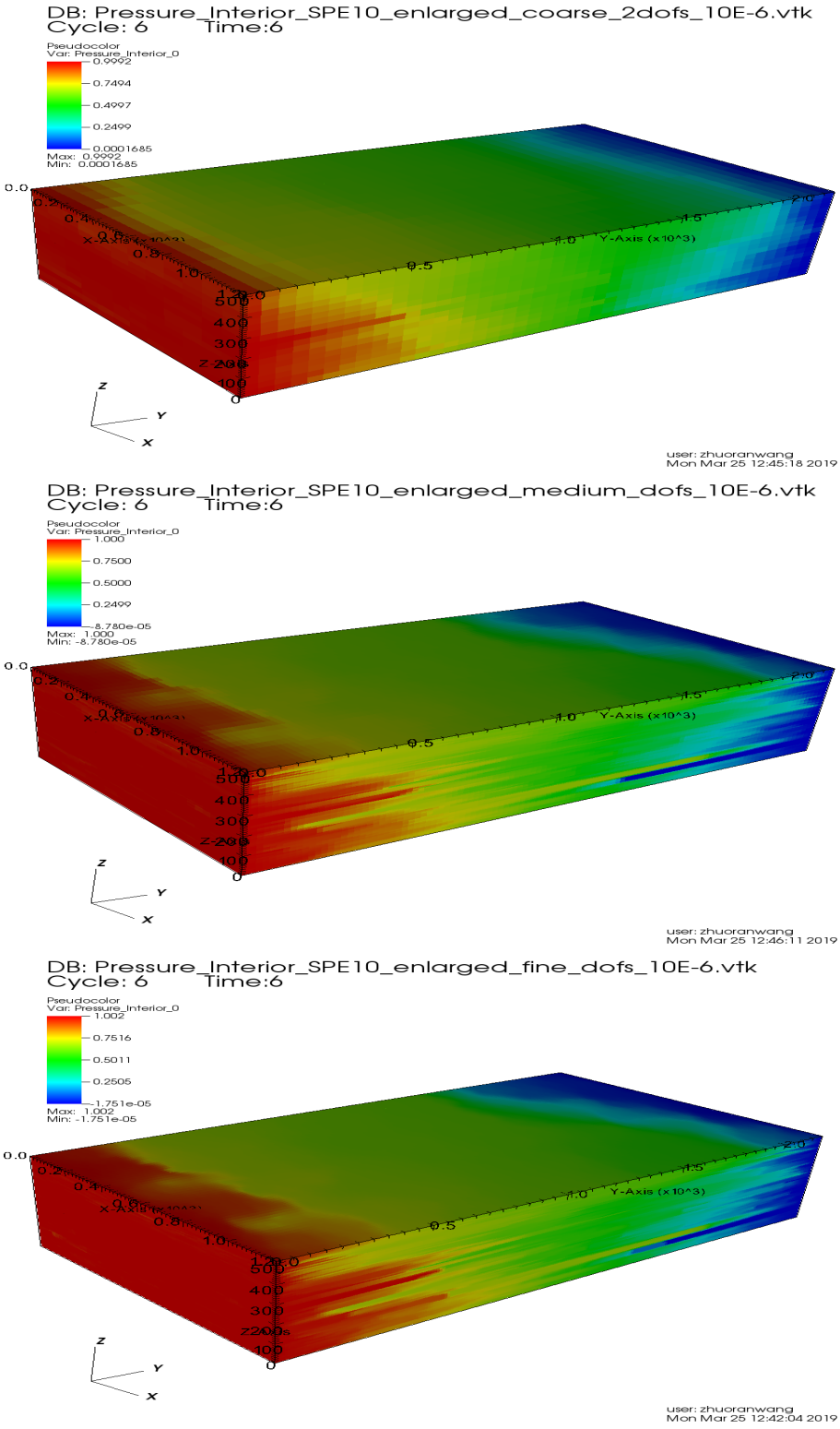


Fig. 4. Example 3 (SPE10): Numerical pressure for coarse, medium, and fine meshes

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

Table 3. Example 3: SPE10_Darcy by WG($Q_0, Q_0; RT_{[0]}$) on 3 meshes

tol=10^{-6}	MaxItrs	#Itrs	$p_{\min}$	$p$max	Runtime
coarse	2*DOFs	40,383	1.684E-4	0.999,151	2m45s
medium	DOFs	207,704	-8.779E-5	1.000,003	1h34m
fine	DOFs	241,492	-1.750E-5	1.002,136	6h42m
tol=10^{-9}	MaxItrs	#Itrs	$p_{\min}$	$p$max	Runtime
coarse	6*DOFs	213,316	1.682E-4	0.999,151	5m24s
medium	DOFs	901,327	-8.782E-5	1.000,002	3h20m
fine	DOFs	1,371,887	-1.750E-5	1.000,083	17h38m

Table 4. Example 3: SPE10_Darcy by MFEM($RT_{[0]}, Q_0$) on 3 meshes

tol	$p_{\min}$	$p$max	Runtime
coarse mesh			
10^{-3}	1.693E-4	0.998,574	8m39s
10^{-6}	1.682E-4	0.999,150	34m02s
10^{-9}	1.682E-4	0.999,151	1h16m
medium mesh			
10^{-3}	-8.712E-5	1.001,850	6h04m
10^{-6}	-8.782E-5	1.000,002	31h25m
10^{-9}	-8.782E-5	1.000,002	82h58m
fine mesh			
10^{-3}	-1.745E-5	1.028,902	71h16m
10^{-6}	DidNotTry		
10^{-9}	DidNotTry		

6 Concluding Remarks

The novel weak Galerkin finite element methods represent a different type of methodology for solving many real-world problems modeled by partial differential equations. There have been efforts on implementing WG FEMs in `Matlab` and `C++`. But the work reported in this paper represents the first ever attempt for implementing WG FEMs in a popular finite element package like `deal.II`. This shall provide open access to the scientific community for examining usefulness of the WG methodology for large-scale scientific computing tasks.

Listed below are some projects for further research.

(i) Preconditioning and parallelization of the WG solver for Darcy flow;
(ii) `deal.II` implementation for coupled WG Darcy solvers and transport solvers for the full problem of SPE10 and alike;
(iii) `deal.II` implementation for both 2-dim and 3-dim for the 2-field poroelasticity solver developed in [10];
(iv) Implementation of WGFEMs for triangular/tetrahedral meshes on `FEniCS`;
(v) Comparison with the hybridizable discontinuous Galerkin (HDG) methods.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37

References

1. https://github.com/dealii/dealii/tree/master/examples/step-61
2. Arbogast, T., Correa, M.: Two families of mixed finite elements on quadrilaterals of minimal dimension. SIAM J. Numer. Anal. **54**, 3332–3356 (2016)
3. Bangerth, W., Hartmann, R., Kanschat, G.: deal.II – A general purpose object oriented finite element library. ACM Trans. Math. Softw. **33**, 24.1–24.27 (2007)
4. Bastian, P., Riviere, B.: Superconvergence and $H(div)$ projection for discontinuous Galerkin methods. Int. J. Numer. Meth. Fluids **42**, 1043–1057 (2003)
5. Brenner, S., Scott, L.: The mathematical theory of finite element methods, Texts in Applied Mathematics, vol. 15. Springer-Verlag, New York, 3rd edn. (2008)
6. Brezzi, F., Fortin, M.: Mixed and hybrid finite element methods. Springer (1991)
7. Bush, L., Ginting, V.: On the application of the continuous Galerkin finite element method for conservation problems. SIAM J. Sci. Comput. **35**, A2953–A2975 (2013)
8. Cockburn, B., Gopalakrishnan, J., Wang, H.: Locally conservative fluxes for the continuous Galerkin method. SIAM J. Numer. Anal. **45**, 1742–1770 (2007)
9. Durlofsky, L.: Accuracy of mixed and control volume finite element approximations to Darcy velocity and related quantities. Water Resour. Res. **30**, 965–973 (1994)
10. Harper, G., Liu, J., Tavener, S., Wang, Z.: A two-field finite element solver for poroelasticity on quadrilateral meshes. Lec. Notes Comput. Sci. **10862**, 76–88 (2018)
11. Harper, G., Liu, J., Zheng, B.: The THex algorithm and a simple Darcy solver on hexahedral meshes. Proc. Comput. Sci. **108C**, 1903–1912 (2017)
12. Lie, K.A.: An introduction to reservoir simulation using MATLAB/GNU Octave:. No. ISBN 9781108492430, Cambridge University Press (2019)
13. Lin, G., Liu, J., Mu, L., Ye, X.: Weak Galerkin finite element methdos for Darcy flow: Anistropy and heterogeneity. J. Comput. Phys. **276**, 422–437 (2014)
14. Lin, G., Liu, J., Sadre-Marandi, F.: A comparative study on the weak Galerkin, discontinuous Galerkin, and mixed finite element methods. J. Comput. Appl. Math. **273**, 346–362 (2015)
15. Liu, J., Sadre-Marandi, F., Wang, Z.: Darcylite: A Matlab toolbox for Darcy flow computation. Proc. Comput. Sci. **80**, 1301–1312 (2016)
16. Liu, J., Tavener, S., Wang, Z.: Lowest-order weak Galerkin finite element method for Darcy flow on convex polygonal meshes. SIAM J. Sci. Comput. **40**, B1229–B1252 (2018)
17. Liu, J., Tavener, S., Wang, Z.: The lowest-order weak Galerkin finite element method for the Darcy equation on quadrilateral and hybrid meshes. J. Comput. Phys. **359**, 312–330 (2018)
18. Mu, L., Wang, J., Ye, X.: A weak Galerkin finite element method with polynomial reduction. J. Comput. Appl. Math. **285**, 45–58 (2015)
19. Sun, S., Liu, J.: A locally conservative finite element method based on piecewise constant enrichment of the continuous Galerkin method. SIAM J. Sci. Comput. **31**, 2528–2548 (2009)
20. Wang, J., Ye, X.: A weak Galerkin finite element method for second order elliptic problems. J. Comput. Appl. Math. **241**, 103–115 (2013)

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_37