An Implementation of a Coupled Dual-Porosity-Stokes Model with FEniCS

Xiukun Hu[1] and Craig C. Douglas[2]

[1] University of Wyoming Department of Mathematics and Statistics
Laramie, WY 82071-3036, USA
xiukun.hu@outlook.com
[2] University of Wyoming School of Energy Resources and Department of Mathematics and Statistics
Laramie, WY 82071-3036, USA
craig.c.douglas@gmail.com

Abstract. Porous media and conduit coupled systems are heavily used in a variety of areas. A coupled dual-porosity-Stokes model has been proposed to simulate the fluid flow in a dual-porosity media and conduits coupled system. In this paper, we propose an implementation of this multi-physics model. We solve the system with the automated high performance differential equation solving environment FEniCS. Tests of the convergence rate of our implementation in both 2D and 3D are conducted in this paper. We also give tests on performance and scalability of our implementation.

Keywords: Domain decomposition, Finite element method, Multi-Physics, Parallel computing, FEniCS.

1 Introduction

The coupling of porous media flow and free flow is of importance in multiple areas, including groundwater system, petroleum extraction, and biochemical transport [1,2,3]. The Stokes-Darcy equation is widely applied in these areas and has been studied thoroughly over the past decade [4,5,6]. Variants of Stokes-Darcy mode, have also been studied extensively [7,8,9,10].

In a traditional Stokes-Darcy system, Darcy's law is applied to the fluid in porous media. Darcy's law, along with its variants, is great in modeling single porosity model with limited Reynolds number, and is widely used in hydrogeology and reservoir engineering. However, for a porous medium with multiple porosities, for example a naturally fractured reservoir, the accuracy of Darcy's law is limited. In contrast, a dual-porosity model assumes two different systems inside a porous media: the matrix system and the microfracture system. These two systems have significantly different fluid storage and conductivity properties. It gives a better representation of the fractured porous media encountered in hydrology, carbon sequestration, geothermal systems, and petroleum extraction [11,12,13].

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

The dual-porosity model itself fails to consider large conduits inside porous media. Thus, the need of coupling both a dual-porosity model with free flow arises [14].

Our paper expands on a coupled Dual-Porosity-Stokes model [15]. Similar to the Stokes-Darcy model, this coupled model contains two nonoverlapping but contiguous regions: one filled with porous media and the other represents conduits. The dual-porosity model describes the porous media and the Stokes equation governs the free flow in the conduits.

In Section 2, we define the Dual-Porosity-Stokes model. The equations are presented as well as the variational form. In Section 3, we describe the numerical implementation using FEniCS. In Section 4, we analyze the accuracy, speed performance, memory usage, and scalability of our implementation. In Section 5, we draw some conclusions and discuss future work.

2 Dual-Porosity-Stokes Model

The Dual-Porosity-Stokes model was first presented in [15]. This paper demonstrated the well-posedness of the model and derived a numerical solution in 2D using a finite element method. Several numerical experiments are also in the paper. In this section we will demonstrate this model in detail as well as present the variational form.

To better understand the model, let us first take a look at a simple 2D example presented in Fig. 1. The model consists of a dual-porosity subdomain Ω_d and a conduit subdomain Ω_c, with an interface Γ_{cd} in between. Two subdomains are non-overlapping, and only communicate to each other through the interface Γ_{cd}. Γ_d and Γ_c are boundaries of each subsystem.

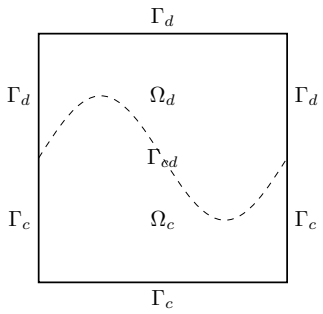

Fig. 1. Coupled Model in 2D

Two fluid pressures are presented in Ω_d, p_m and p_f, for fluids in matrix and fractures respectively. The m subscript stands for *matrix* and f is for *fracture*. We use these two subscripts for other model parameters. The dual-porosity model can be expressed as:

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

$$\phi_m C_{mt} \frac{\partial p_m}{\partial t} - \nabla \cdot \frac{k_m}{\mu} \nabla p_m = -Q \,, \tag{1}$$

$$\phi_f C_{ft} \frac{\partial p_f}{\partial t} - \nabla \cdot \frac{k_f}{\mu} \nabla p_f = Q + q_p \,. \tag{2}$$

The constant σ is a shape factor ranging from 0 to 1. It measures the connectivity between the microfracture and the matrix. μ is the dynamic viscosity. k is the intrinsic permeability. ϕ denotes the porosity. C_{mt} and C_{ft} denote the total compressibility for the two systems respectively. q_p is the sink/source term. Q is the mass exchange between matrix and microfracture systems and can be derived from $Q = \frac{\sigma k_m}{\mu}\left(p_m - p_f\right)$.

In the conduit subdomain, we use the linear incompressible Stokes equation to describe the free flow:

$$\frac{\partial \boldsymbol{u}_c}{\partial t} - \nabla \cdot \mathbb{T}(\boldsymbol{u}_c, p) = \boldsymbol{f} \,, \tag{3}$$

$$\nabla \cdot \boldsymbol{u}_c = 0 \,. \tag{4}$$

The flow velocity $\boldsymbol{u}_c$ and pressure p together describe the free flow. ν is the kinematic viscosity. $\boldsymbol{f}$ is a general source term. $\mathbb{T}(\boldsymbol{u}_c\,, p) \coloneqq 2\nu\mathbb{D}(\boldsymbol{u}_c) - p\mathbb{I}$ is the stress tensor, where $\mathbb{D}(\boldsymbol{u}_c) \coloneqq \frac{1}{2}\left(\nabla \boldsymbol{u}_c + \nabla^T \boldsymbol{u}_c\right)$ is the deformation tensor, and $\mathbb{I}$ is the $N \times N$ identity matrix.

On the interface Γ_{cd}, a no-exchange condition between the matrix and the conduit is used,

$$-\frac{k_m}{\mu} \nabla p_m \cdot (-\boldsymbol{n}_{cd}) = 0 \,, \tag{5}$$

where $\boldsymbol{n}_{cd}$ is the unit normal vector on the interface pointing toward Ω_d. This equation forces the fluid in the matrix to stay in the porous media. It is based on the fact that the permeability of the matrix system is usually 10^5 to 10^7 times smaller than the microfracture permeability [16,17,18,19].

Three more interface conditions are derived from Stokes-Darcy model:

$$\boldsymbol{u}_c \cdot \boldsymbol{n}_{cd} = -\frac{k_f}{\mu} \nabla p_f \cdot \boldsymbol{n}_{cd} \,, \tag{6}$$

$$-\boldsymbol{n}_{cd}^T \mathbb{T}(\boldsymbol{u}_c, p)\boldsymbol{n}_{cd} = \frac{p_f}{\rho} \,, \tag{7}$$

$$-\mathbb{P}_\tau(\mathbb{T}(\boldsymbol{u}_c, p)\boldsymbol{n}_{cd}) = \frac{\alpha\nu\sqrt{N}}{\sqrt{\mathrm{trace}(\boldsymbol{\Pi})}}\left(\boldsymbol{u}_c + \frac{k_f}{\mu}\nabla p_f\right) . \tag{8}$$

ρ is the density of the fluid. $\mathbb{P}_\tau$ is the projection operator onto the local tangent plane of the interface Γ_{cd}. α is a dimensionless parameter which depends on the properties of the fluid and the permeable material, N is the space dimension, and $\boldsymbol{\Pi} = k_f\mathbb{I}$ is the intrinsic permeability of the fracture media. Condition (6) is the conservation of mass on

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

the interface. Equation (7) represents the balance of forces on the interface [20,21]. Equation (8) is the Beavers-Joseph interface condition [22].

If we introduce a vector valued test function $\vec{\boldsymbol{v}} = \left[\psi_m, \psi_f, \boldsymbol{v}^T, q\right]^T$, the variational form for our model can be written as

$$\int_{\Omega_d} \left(\phi_m C_{mt} \frac{\partial p_m}{\partial t} \psi_m + \frac{k_m}{\mu} \nabla p_m \cdot \nabla \psi_m + \frac{\sigma k_m}{\mu} (p_m - p_f) \psi_m \right) \mathrm{d}\Omega \tag{9a}$$

$$+ \int_{\Omega_d} \left(\phi_f C_{ft} \frac{\partial p_f}{\partial t} \psi_f + \frac{k_f}{\mu} \nabla p_f \cdot \nabla \psi_f + \frac{\sigma k_m}{\mu} (p_f - p_m) \psi_f \right) d\Omega \tag{9b}$$

$$+\eta \int_{\Omega_c} \left(\frac{\partial \boldsymbol{u}_c}{\partial t} \cdot \boldsymbol{v} + 2\nu \mathbb{D}(\boldsymbol{u}_c) : \mathbb{D}(\boldsymbol{v}) - p \nabla \cdot \boldsymbol{v} \right) d\Omega \tag{9c}$$

$$+\eta \int_{\Gamma_{cd}} \left(\frac{1}{\rho} p_f \boldsymbol{v} \cdot \boldsymbol{n}_{cd} + \frac{\alpha \nu \sqrt{N}}{\sqrt{\mathrm{trace}(\boldsymbol{\Pi})}} \mathbb{P}_\tau \left(\boldsymbol{u}_c + \frac{k_f}{\mu} \nabla p_f \right) \cdot \boldsymbol{v} \right) d\Gamma \tag{9d}$$

$$+\eta \int_{\Omega_c} \nabla \cdot \boldsymbol{u}_c q \, d\Omega - \int_{\Gamma_{cd}} \boldsymbol{u}_c \cdot \boldsymbol{n}_{cd} \psi_f \, d\Gamma \tag{9e}$$

$$= \eta \int_{\Omega_c} \boldsymbol{f} \cdot \boldsymbol{v} \, d\Omega + \int_{\Omega_d} q_p \psi_f \, d\Omega \, . \tag{9f}$$

η is a scale factor applied to equations in the conduit subdomain to ensure the whole system is of the same scale.

3 Implementation

Hou et al [15] numerically solved such a system in 2D using a finite element method with Taylor-Hood elements for the conduit domain and demonstrated the stability and convergence rate of their method.

In this section we describe an implementation of a finite element solver based on FEniCS [23,24], which allows us to run our model in both 2D and 3D, in parallel, and can be easily modified and extended. FEniCS is a popular open source computing platform for solving partial differential equations (PDEs). The automatic code generation of FEniCS enables people to implement a FEM code using the Unified Form Language (UFL) [25], which is close to a mathematical description of the variational form.

Many multi-physics models have been implemented using FEniCS, e.g., the adaptive continuum mechanics solver Unicorn (Unified Continuum modeling) [26,27]. It can solve continuum mechanics with moving meshes adaptively. As the coupled systems Unicorn solves always consist of moving meshes, subsystems are solved independently and iteratively until a satisfied error is reached. In our case, we prefer to solve the coupled system as a whole as they together form a linear system and can be solved directly.

A solution to the coupled Navier-Stokes-Darcy model has been implemented with FEniCS by Ida Norderhaug Drøsdal [28]. In the coupled Navier-Stokes-Darcy model, the conduit subdomain and the porosity subdomain contain the same two variables: fluid velocity and pressure. The solver regards the two subsystems as a whole and the

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

variables exist on both subdomains. The interface conditions then are implemented by interior facet integration. However, in our coupled Stokes-Dual-Porosity model, we have two scalar variables p_f and p_m in the dual-porosity domain, but one vector variable $\boldsymbol{u}_c$ and one scalar variable p in the conduit domain.

The disagreement of the variable dimensions on the two subdomains differentiates our model from Navier-Stokes-Darcy model. Here we expand every variable to the whole system and force them to be zero in the opposite subdomain.

3.1 Implementation with FEniCS

Our implementation in Python is described in this section. Since our model involves interior interface integration, adjacent cells need to share information from the common facet. In order for our implementation to run in parallel, the following parameter in FEniCS needs to be set correctly.

```
from fenics import *
parameters['ghost_mode'] = 'shared_facet'
```

For any given mesh object mesh, with any geometric dimension, our function space can be created as:

```
# Given mesh and degree
velem = VectorElement('CG', mesh.ufl_cell(), degree)
selem = FiniteElement('CG', mesh.ufl_cell(), degree)
pelem = FiniteElement('CG', mesh.ufl_cell(), degree-1)

V = FunctionSpace(mesh, MixedElement(selem,selem,
                                     velem,pelem))
```

The four ordered elements `selem`, `selem`, `velem`, `pelem` in the last statement are for p_f, p_m, $\boldsymbol{u}_c$ and p respectively. Note that since the Taylor-Hood method is applied, the degree of p should be less than that of $\boldsymbol{u}_c$.

Initiate constants `phim` $= \varphi_m$, `phif` $= \varphi_f$, `km` $= k_m$, `kf` $= k_f$, `mu` $= \mu$, `nu` $= \nu$, `rho` $= \rho$, `sigma` $= \sigma$, `Cmt` $= C_{mt}$, `Cft` $= C_{ft}$, `alpha` $= \alpha$, and `eta` $= \eta$, and function expressions `qp` $= q_p$ and `f` $= \boldsymbol{f}$. Also define initial conditions for all variables, interpolated into our function space `V`, and stored in the variable `x0`. The variational form can be defined in UFL as below. Note the one-to-one correspondence between the variational form below and the one presented in (9a)–(9f).

```
n = FacetNormal(V.mesh()) # n_cd
proj = lambda u: u-dot(u,n('+'))*n('+') # P_τ
pm0, pf0, u0, p0 = x0.split()
avN = alpha*nu/math.sqrt(kf)  # αν√N)/√(trace(Π)
pm, pf, u, p = TrialFunctions(V)
psim, psif, v, q = TestFunctions(V)
F = ((phim*Cmt*(pm-pm0)/dt*psim
```

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

```
     + km/mu*dot(grad(pm), grad(psim))
     + sigma*km/mu*(pm-pf)*psim        )*dD # (9a)
  + (phif*Cft*(pf-pf0)/dt*psif
     + kf/mu*dot(grad(pf), grad(psif))
     + sigma*km/mu*(pf-pm)*psif        )*dD # (9b)
  + eta*
    (dot((u-u0)/dt,v)
     + 2*nu*inner(epsilon(u),epsilon(v))
    - p*div(v)                         )*dC # (9c)
  + eta*
    (1/rho*pf('-')*dot(v('+'),n('+'))
     + avN*dot(proj(u('+')+kf/mu*grad(pf('-'))),
               v('+'))                 )*dI # (9d)
  + (eta*(div(u)*q)*dC
     - dot(u('+'),n('+'))*psif('-')*dI)     # (9e)
  - eta*dot(f, v)*dC - qp*psif*dD    )      # (9f)
a, L = lhs(F), rhs(F)
```

Note that the backward Euler scheme can be easily extended to θ method. `dC`, `dD` and `dI` are predefined `Measure` objects in UFL, and represents integrations on Ω_c, Ω_d and Γ_{cd} respectively. Note that the sign in `n('+')` needs to be adjusted for different domain structures. The signs of other variables for interface integration terms are not affecting the result of the model in any of our test cases.

Now we constrain $p_m, p_f, \boldsymbol{u}_c, p$ on opposite subdomains by defining the following Dirichlet boundary conditions.

```
fix_pm = DirichletBC(V.sub(0), Constant(0),
                     on_conduit_but_not_interface,
                     method='pointwise')
fix_pf = DirichletBC(V.sub(1), Constant(0),
                     on_conduit_but_not_interface,
                     method='pointwise')
fix_u = DirichletBC(V.sub(2), Constant([0]*N),
                    on_dual_but_not_interface,
                    method='pointwise')
fix_p = DirichletBC(V.sub(3), Constant(0),
                    on_dual_but_not_interface,
                    method='pointwise')
fix_bcs = [fix_pm, fix_pf, fix_u, fix_p]
```

The boundary markers `on_conduit/dual_but_not_interface`, as their names might suggest, should not include the interface Γ_{cd}. Note that we need to use the method "pointwise" for these special "boundary" conditions.

After creating other Dirichlet boundary conditions `bcs`, we can solve the PDE as follows.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

```
A, b = assemble(a), assemble(L)
for bc in fix_bcs + bcs:
    bc.apply(A, b)
solver = KrylovSolver(A, method='bicgstab',
                      preconditioner='hypre_euclid')
now = 0
while now <= T:  # T is endtime
    now += dt    # dt is length of timestep
    for expr in [pm, pf, u, p, qp, f]:
        expr.t = now
    b = assemble(L)
    for bc in bcs:
        bc.apply(b)
    solver.solve(x.vector(), b)
    x0.assign(x)
```

Due to the interface conditions our matrix A is nonsymmetric. Hence, methods like conjugate gradients and Cholesky decomposition might not work as expected.

4 Result

The implementation works in 2D and 3D with the same code. Here we test our implementation on a unit cubic mesh defined by $\Omega = [0,1] \times [0,1] \times [0,1]$. Let $\Omega_d = \{(x,y,z) \in \Omega \mid x \leq y\}, \Omega_c = \{(x,y,z) \in \Omega \mid x \geq y\}$. We simulate our model on the time interval [0,1].

For the constants, we let $k_m = 0.1$ and all the rest be 1. We set up our coefficients and essential boundary conditions so that the following is our solution:

$$p_m = \frac{1}{5}\sin(x+y+z)\cos(\pi t)$$

$$p_f = -2\pi \sin(\pi t)\sin(x+y+z) + \frac{4}{5}\sin(x+y+z)\cos(\pi t)$$

$$\boldsymbol{u_c} = \begin{bmatrix} 2\pi \sin(\pi t)\cos(3x-y+z) - \frac{4}{5}\cos(\pi t)\cos(3y-x+z) \\ 2\pi \sin(\pi t)\cos(3y-x+z) - \frac{4}{5}\cos(\pi t)\cos(3x-y+z) \\ 2\pi \sin(\pi t)\cos(x+y+z) - \frac{4}{5}\cos(\pi t)\cos(x+y+z) \end{bmatrix}$$

$$p = -8\left(\pi \sin(\pi t) + \frac{2}{5}\cos(\pi t)\right)(\sin(3y-x+z) + \sin(3x-y+z)) + 2\left(\frac{2}{5}\cos(\pi t) - \pi \sin(\pi t)\right)\sin(x+y+z)$$

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

It is not hard to verify that the above solution satisfies equation (1) and (5)-(8). Source terms q_p and $\boldsymbol{f}$ can be calculated from (2) and (3) respectively. However, the divergence of the free flow velocity $\boldsymbol{u}_c$ is not zero, so we need to modify (4) to a more general case $\nabla \cdot \boldsymbol{u}_c = g$, and calculate g from it.

For the boundary conditions, we apply corresponding essential boundaries for all variables except for p.

We tested our implementation on the University of Wyoming's Teton HPC cluster [29].

4.1 Convergence

We examine the convergence rate of our implementation for different timestep length Δt's. To make the result reproducible, all of the experiments are run in a single processor with the direct linear solver MUMPS (Multifrontal Massively Parallel sparse direct Solver) [30,31]. Piecewise quadratic functions are used for p_m, p_f, and $\boldsymbol{u}_c$. Degree 1 Lagrange elements are used for p.

We examine the convergence rate for both $\Delta t = h$ and $\Delta t = h^2$, where h is the cell size. Recall that our domain is a unit cube. A model with cell size h means our domain is partitioned into $h \times h \times h$ small cubes. Each cube contains 6 tetrahedral cells. The L^2 norm of each variable's error at time $T = 1$ is calculated. The convergence rate is calculated as $\ln(e_i/e_{i-1})/\ln(h_i/h_{i-1})$. The result is shown in Table 1 and Table 2.

Table 1. The L^2 error at $T = 1$ with $\Delta t = h$.

h	p_m		p_f		$\boldsymbol{u}_c$		p	
	error	rate	Error	rate	error	rate	error	Rate
1/2	3.56e-2	0.60	1.46e-1	1.12	3.93e-2	3.78	1.50	56.40
1/4	2.01e-2	0.82	5.64e-2	1.37	1.35e-2	1.54	3.48e-1	2.10
1/8	1.07e-2	0.91	2.17e-2	1.38	4.80e-3	1.49	8.87e-2	1.97
1/16	5.54e-3	0.95	9.04e-3	1.27	1.89e-3	1.34	3.51e-2	1.34

Table 2. The L^2 error at $T = 1$ with $\Delta t = h^2$.

h	p_m		p_f		$\boldsymbol{u}_c$		p	
	error	rate	error	rate	error	rate	error	rate
1/2	2.03e-2	1.41	5.59e-2	2.51	1.68e-2	5.01	9.10e-1	57.12
1/4	5.69e-3	1.83	9.38e-3	2.57	2.26e-3	2.90	1.72e-1	2.41
1/8	1.44e-3	1.97	1.95e-3	2.26	4.08e-4	2.47	3.90e-2	2.14
1/16	3.62e-4	2.00	4.58e-4	2.08	9.98e-5	2.16	9.35e-3	2.06

4.2 Performance

The solver consists of two major parts: linear system assembling and linear system solving. Below we investigate the performance of our implementation in these two parts separately.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

Assembling. Despite that the linear form L is assembled at each time step, the assembling of the bilinear form a is usually more time consuming. Fig. 2 and Fig. 3 show the time spent for assembling the bilinear form under different conditions.

Fig. 2 shows the assembling time when using a single CPU versus degrees of freedom (DoF) of our system. We can see that the assembling time is linear to total DoF.

Fig. 3 presents the performance of assembling along different number of processors. Each line presents the scalability with fixed problem size. We can see that assembling scales well when the problem size is large enough. However, too many processors may lead to a performance drop.

Solving. For our time-dependent model, the same linear system is solved at each timestep with varying right hand side. In this case, direct linear solvers can benefit from reuse of decompositions, as we will see in Fig. 4 and Fig. 5.

A collection of high performance solvers and preconditioners are available (callable) from FEniCS, assuming it was built with corresponding packages. To reduce complexity, we choose the ILU preconditioner `hypre_euclid` from Livermore's HYPRE package [32] for all iterative solvers we use.

For a single solve, direct solvers like MUMPS and `superlu_dist` (Supernodal LU [33,34]) is slower than iterative solvers, as shown in Fig. 4. But if we simulate for 100 steps, the direct solver MUMPS can overpass iterative solvers in not too large systems.

However, we can see in both figures, `superlu_dist` suffers from scalability. For large systems, a direct solver can still be slower and is much more memory consuming.

Memory Usage. Fig. 6 and Fig. 7 present the memory usages of our model under different situations. All memory usages are measured as the "Resident Set Size" of running processes. The memory usage is measured by the Resident Set Size used when running a simulation for $\Delta t = 0.01, t \in [0,1]$, with specific solver. Note that the memory usage for iterative solvers are very similar: all their lines are overlapped with each other and some becomes invisible.

For large systems, the memory usages of iterative solvers are about linear with respect to DoF and are worse than linear for direct solvers. In the case of $h = 1/32$, the total memory usage of a system with `superlu_dist` is about 7 times as large as that of a system with an iterative solver, as shown in Fig. 6.

Fig. 7 shows how memory usage scales if we add more processors. The memory usage is the memory used by a single processor.

5 Conclusions and Future Work

We have presented an implementation of a coupled dual-porosity-Stokes model using the automated FEM solver FEniCS. We proposed a solution to modeling the coupled interface by using FEniCS' built-in interface integration and expanding variables to the whole domain. This approach enables us to simulate both 2D and 3D models in parallel

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

with minimum coding. Future work will include adding data assimilation from active sensors and experimenting with different interface conditions to see better solutions can be computed. Another approach is to implement one of the non-iterative domain decomposition methods that have been developed for Stokes-Darcy systems [35,36], which can decompose our asymmetric matrix into two small symmetric matrices and reduce communications between two subsystems.

Acknowledgment: This research was supported in part by NSF grant 1722692.

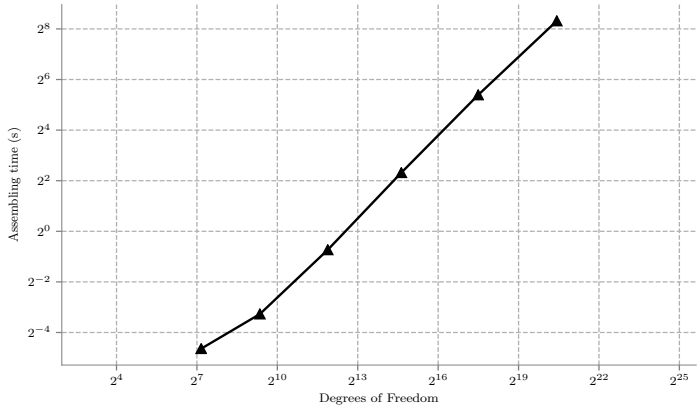

Fig. 2. Assembling time is linear to DoF.

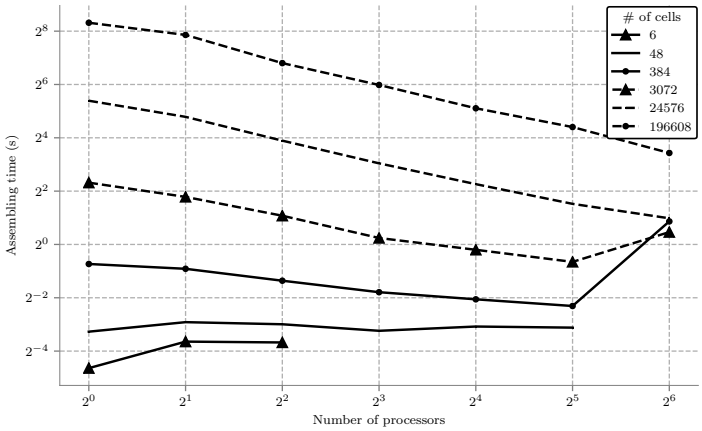

Fig. 3. Assembling scales well in large systems.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

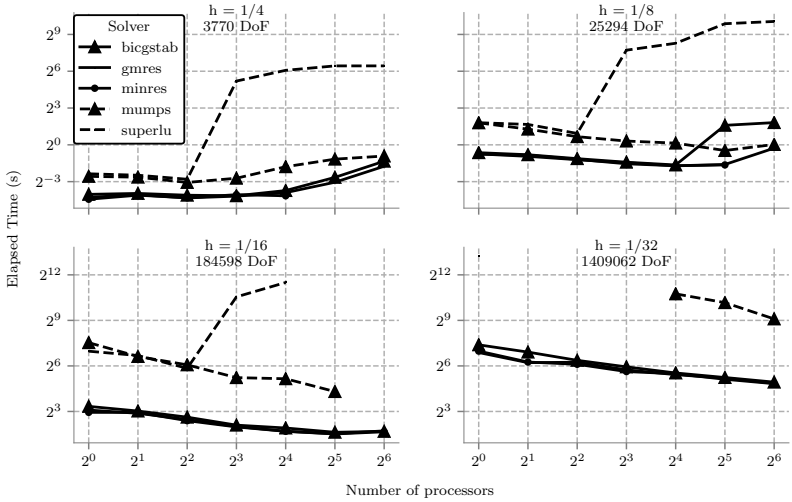

Fig. 4. Time for solving a single step.

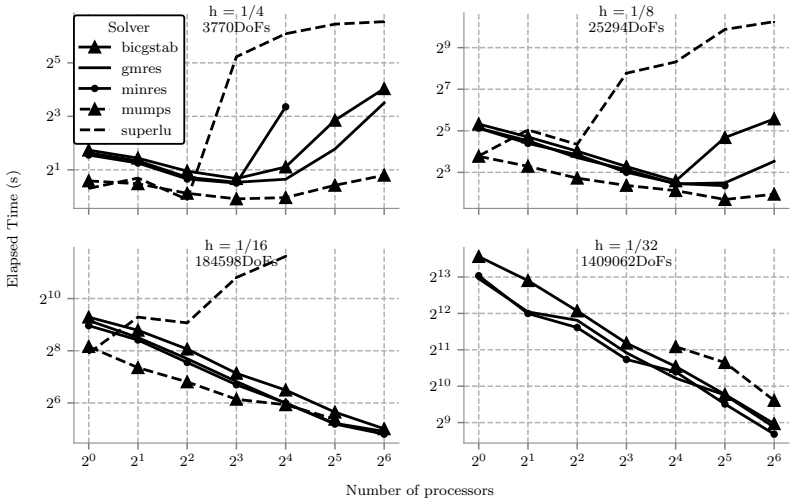

Fig. 5. Time for solving 100 steps.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

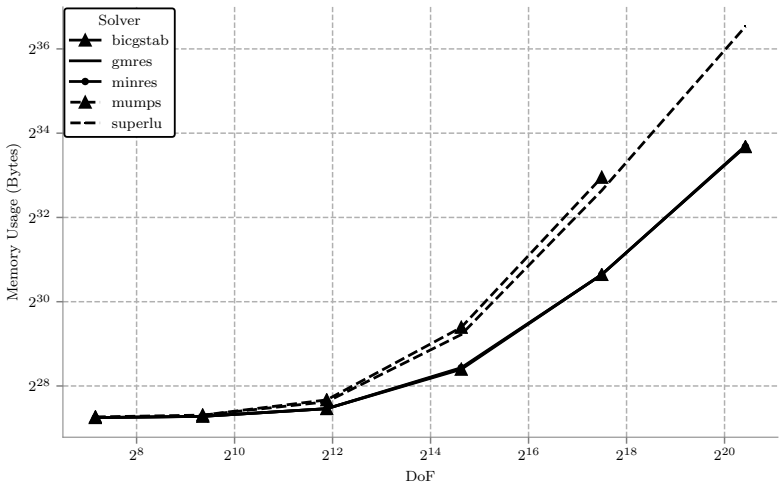

Fig. 6. Memory usage versus degrees of freedom.

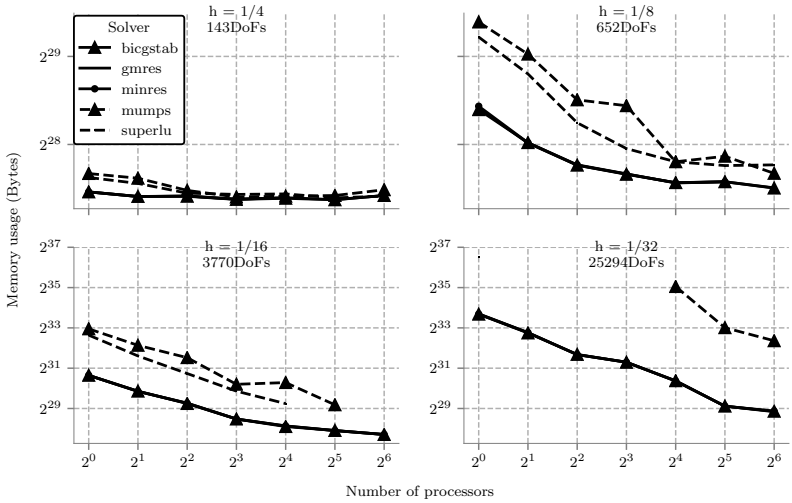

Fig. 7. Memory usage versus number of processors.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

References

1. Çeşmelioğlu, A., Rivière, B.: Primal discontinuous Galerkin methods for time-dependent coupled surface and subsurface flow. Journal of Scientific Computing 40(1–3), 115–140 (2009)
2. Arbogast, T., Brunson, D.: A computational method for approximating a Darcy–Stokes system governing a vuggy porous medium. Computational geosciences 11(3), 207–218 (2007)
3. Cao, S., Pollastrini, J., Jiang, Y.: Separation and characterization of protein aggregates and particles by field flow fractionation. Current pharmaceutical biotechnology 10(4), 382–390 (2009)
4. Babuška, I., Gatica, G.: A residual-based a posteriori error estimator for the Stokes–Darcy coupled problem. SIAM Journal on Numerical Analysis 48(2), 498–523 (2010)
5. Badea, L., Discacciati, M., Quarteroni, A.: Numerical analysis of the Navier–Stokes/Darcy coupling. Numerische Mathematik 115(2), 195–227 (2010)
6. Boubendir, Y., Tlupova, S.: Domain decomposition methods for solving Stokes--Darcy problems with boundary integrals. SIAM Journal on Scientific Computing 35(1), B82–B106 (2013)
7. Badia, S., Codina, R.: Unified stabilized finite element formulations for the Stokes and the Darcy problems. SIAM journal on Numerical Analysis 47(3), 1971–2000 (2009)
8. Bernardi, C., Hecht, F., Pironneau., O.: Coupling Darcy and Stokes equations for porous media with cracks. ESAIM: Mathematical Modelling and Numerical Analysis 39(1), 7–35 (2005)
9. Amara, M., Capatina, D., Lizaik, L.: Coupling of Darcy–Forchheimer and compressible Navier–Stokes equations with heat transfer. SIAM Journal on Scientific Computing 31(2), 1470–1499 (2009)
10. Dawson, C.: Analysis of discontinuous finite element methods for ground water/surface water coupling. SIAM Journal on Numerical Analysis 44(4), 1375–1404 (2006)
11. Arbogast, T., Douglas, J., Hornung, U.: Derivation of the double porosity model of single phase flow via homogenization theory. SIAM Journal on Mathematical Analysis 21(4), 823–836 (1990)
12. Carneiro, J.: Numerical simulations on the influence of matrix diffusion to carbon sequestration in double porosity fissured aquifers. International Journal of Greenhouse Gas Control 3(4), 431–443 (2009)
13. Gerke, H., Genuchten., M.: Evaluation of a first-order water transfer term for variably saturated dual-porosity flow models. Water Resources Research 29(4), 1225–1238 (1993)
14. Douglas, C., Hu, X., Bai, B., He, X., Wei, M., Hou, J.: A Data Assimilation Enabled Model for Coupling Dual Porosity Flow with Free Flow. In: 2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Wuxi, pp.304–307 (2018)
15. Hou, J., Qiu, M., He, X., Guo, C., Wei, M., Bai, B.: A Dual-Porosity-Stokes Model and Finite Element Method for Coupling Dual-Porosity Flow and Free Flow., B710–B739 (2016)
16. Bello, R., Wattenbarger, R.: Multi-stage hydraulically fractured horizontal shale gas well rate transient analysis. In: North Africa technical conference and exhibition (2010)
17. Brohi, I., Pooladi-Darvish, M., Aguilera, R.: Modeling fractured horizontal wells as dual porosity composite reservoirs-application to tight gas, shale gas and tight oil cases. In: SPE Western North American Region Meeting (2011)

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5

18. Carlson, E., Mercer, J.: Devonian shale gas production: mechanisms and simple models. Journal of Petroleum technology 43(04), 476–482 (1991)
19. Guo, C., Wei, M., Chen, H., Xiaoming, H., Bai, B.: Improved numerical simulation for shale gas reservoirs. In: Offshore Technology Conference-Asia (2014)
20. Çeşmelioğlu, A., Rivière, B.: Analysis of time-dependent Navier-Stokes flow coupled with Darcy flow. Journal of Numerical Mathematics 16(4), 249–280 (2008)
21. Chidyagwai, P., Rivière, B.: On the solution of the coupled Navier--Stokes and Darcy equations. Computer Methods in Applied Mechanics and Engineering 198(47–48), 3806–3820
22. Beavers, G., Joseph, D.: Boundary conditions at a naturally permeable wall. Journal of fluid mechanics 30(1), 197–207 (1967)
23. Alnæs, M., Jan, B., Hake, J., August, J., Kehlet, B., Logg, A., Ring, J., Rognes, M., Wells, G.: The FEniCS Project Version 1.5. Archive of Numerical Software 3(100) (2015)
24. Logg, A., Mardal, K.-A., Wells, G.: Automated Solution of Differential Equations by the Finite Element Method. Springer (2012)
25. Logg, A., Mardal, K.-A., Wells, G.: UFL: a finite element form language. In: Automated Solution of Differential Equations by the Finite Element Method, Volume 84 of Lecture Notes in Computational Science and Engineering 17. Springer (2012)
26. Hoffman, J., Jansson, J., Degirmenci, C., Jansson, N., Nazarov, M.: Unicorn: A unified continuum mechanics solver. In: Automated Solution of Differential Equations by the Finite Element Method, Volume 84 of Lecture Notes in Computational Science and Engineering 18. Springer (2012)
27. Hoffman, J., Jansson, J., Jansson, N.: FEniCS-HPC: Automated predictive high-performance finite element computing with applications in aerodynamics. In: International Conference on Parallel Processing and Applied Mathematics, Cham, pp.356–365 (2015)
28. Drøsdal, I.: Porous and viscous modeling of cerebrospinal fluid flow in the spinal canal associated with syringomyelia. Master's Thesis (2011)
29. Advanced Research Computing Center: Teton Computing Environment, Intel x86_64 cluster. University of Wyoming https://doi.org/10.15786/M2FY47, Laramie (2018)
30. Amestoy, P., Duff, I., Koster, J., L'Excellent, J.-Y.: A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM Journal on Matrix Analysis and Applications 23(1), 15–41 (2001)
31. Amestoy, P., Cuermouche, A., L'Excellent, J.-Y., Pralet, S.: Hybrid scheduling for the parallel solution of linear systems. Parallel Computing 32(2), 136–156 (2006)
32. Falgout, R., Yang, U.: hypre: A Library of High Performance Preconditioners. In Sloot, P., Hoekstra, A., Tan, C., Dongarra, J., eds.: Computational Science — ICCS 2002, Berlin, pp.631–641 (2002)
33. Li, X., Demmel, J., Gilbert, J., Grigori, i., Yamazaki, M.: SuperLU Users' Guide. Lawrence Berkeley National Laboratory LBNL-44289 (2005) http://crd.lbl.gov/~xiaoye/SuperLU/.
34. Li, X., Demmel, J.: SuperLU_DIST: A Scalable Distributed-Memory Sparse Direct Solver for Unsymmetric Linera Systems. ACM Trans. Mathematical Software 29(2), 110–140 (June 2003)
35. Cao, Y., Gunzburger, M., He, X., Wang, X.: Paralle, non-iterative, multi-pysics domain decomposition methods for time-dependent Stokes-Darcy systems., 1617–1644 (2014)
36. Feng, W., He, X., Wang, Z., Zhang, X.: Non-iterate domain decomposition methods for a non-stationary Stokes-Darcy model with Beavers-Joseph interface condition., 453–463 (2012)

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22747-0_5