Framework for hybridization of mixed systems in MFEM

MFEM Community Workshop

Sep 10 – 11, 2025

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) Program through the FASTMath Institute, under contract number DE-SC0021285, and RF-SciDAC Center, under contract number DE-SC0024369.

Prepared by LLNL under Contract DE-AC52-07NA27344.

RF-SciDAC

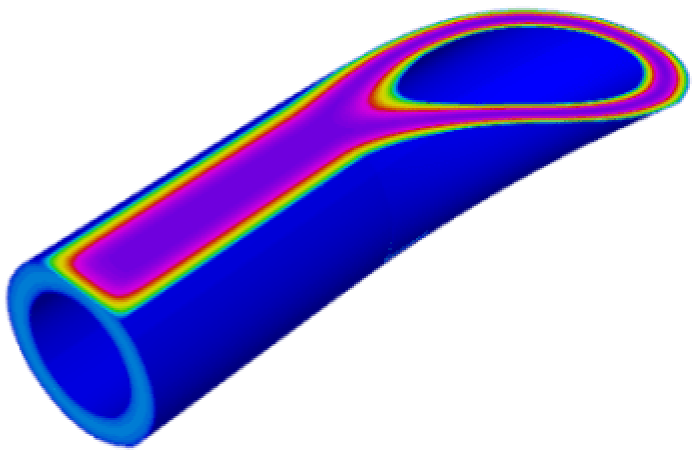
FASTMATH

CASC
Center for Applied Scientific Computing

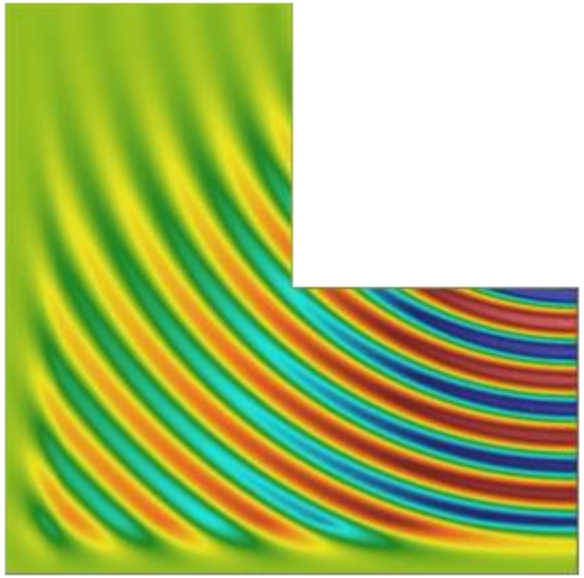
Jan Nikl | NA&S, CASC

Mixed systems

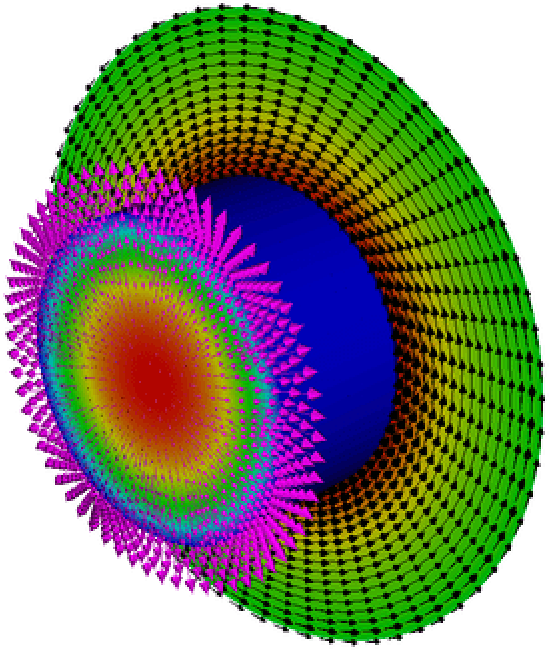
- (In)definite – Darcy, Heat conduction, Maxwell, …



Example 14
(DG diffusion)



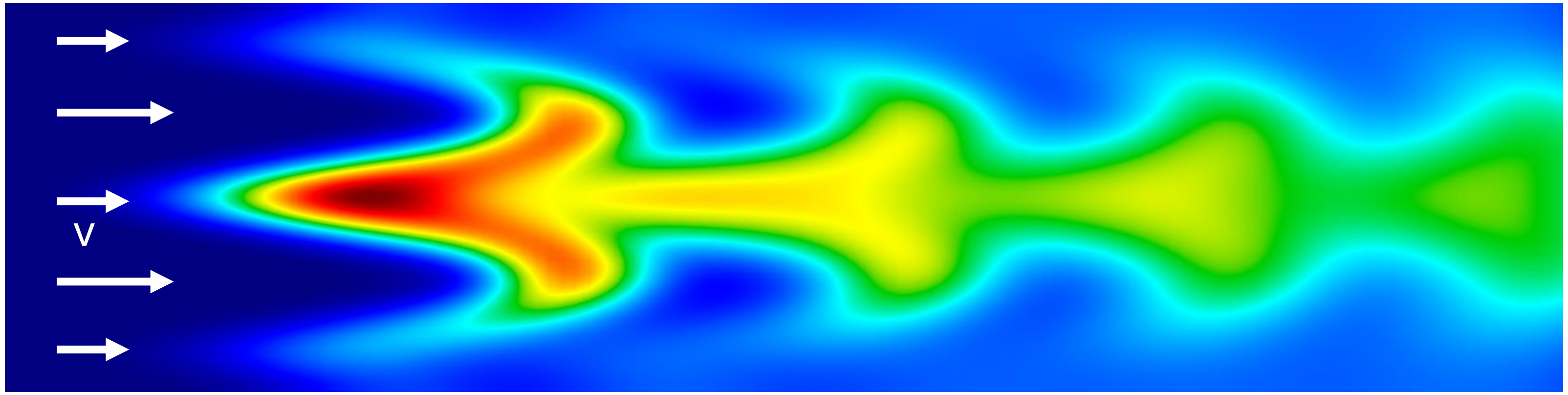
Example 25
(Maxwell problem)



Joule miniapp
(magnetic + thermal diffusion)

Mixed systems

- (In)definite – Darcy, Heat conduction, Maxwell, ...
- Convection-diffusion



Kovasznay flow

Mixed systems

- (In)definite – Darcy, Heat conduction, Maxwell, ...
- Convection-diffusion

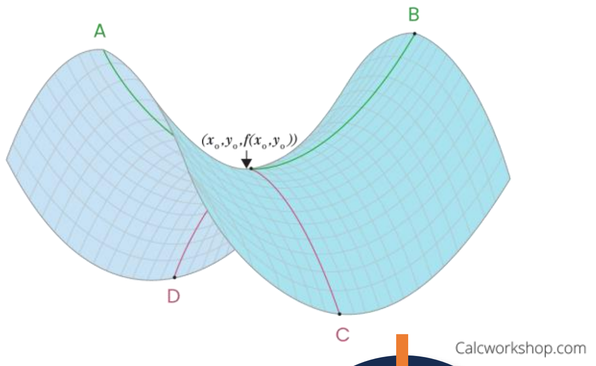
$$\underline{\boldsymbol{q}} + \kappa \nabla \underline{u} = 0, \quad \text{in } \Omega,$$
$$\nabla \cdot (\boldsymbol{c}\underline{u} + \underline{\boldsymbol{q}}) = f, \quad \text{in } \Omega,$$

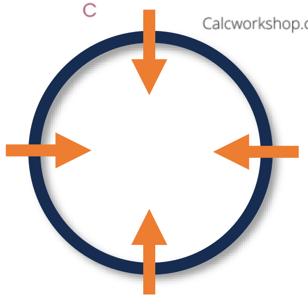
- Flux – continuous (RT, ND,...) / discontinuous
- Potential – discontinuous
- Block-(anti)symmetric weak form (not symmetric!)

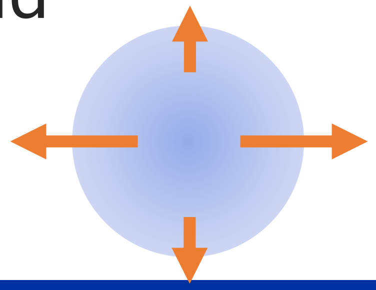
$$\begin{bmatrix} A & -B^T \\ B & D \end{bmatrix}$$

Challenges

- **Definiteness** – saddle-point → *primary*?
- **Divergence-free** – potential eq. → *mixed*?
- **Conservation** – local cons. of DG potential
- **Preconditioning** – tight coupling → *primary*?
- **Memory consumption / data motion** – memory bound
- **Anisotropy** – ringing of *CG*, preconditioning → *DG*?





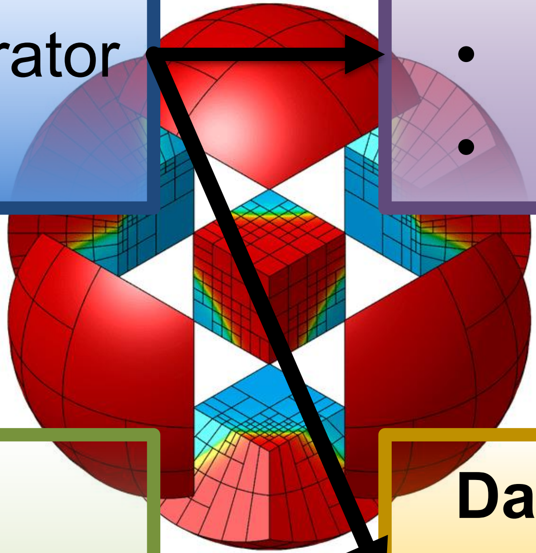


Best of *primary* and *mixed* formulation? *DG*s? And beyond?

Framework for mixed systems

(Par)DarcyForm
- Constructs the block operator
- Elimination of ess. BCs

DarcyHybridization
- Hybridization of total flux
- Static condensation to traces



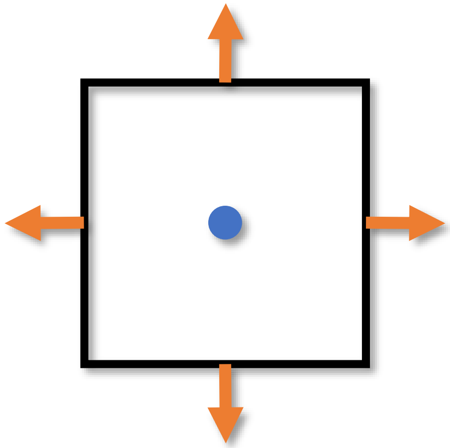
(DarcyOperator)
- Schur complement precond.
- Newton/GMRES/AMG setup

DarcyReduction
- Elimination of flux/potential
- Elimination of ess. BCs

Mixed formulation – RTDG/LDG

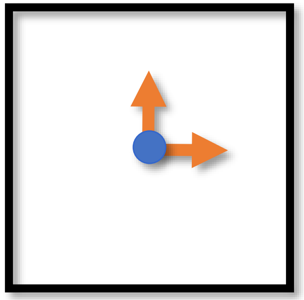
- Finite element method – discrete fxs u_h,q_h test fxs v, w
- (Bi)linear forms – (•,•) volume, ⟨•, •⟩ face
- *Raviart-Thomas + Discontinuous Galerkin (RTDG)*

$$
\begin{aligned}
(\kappa^{-1}\mathrm{q}_h, \mathrm{v})_K - (u_h, \nabla\cdot\mathrm{v})_K &= 0, \\
(\nabla\cdot\mathrm{q}_h, w)_K - (\mathrm{c}u_h, \nabla w)_K + \langle \widehat{\mathrm{c}u_h}\cdot\mathrm{n}, w\rangle_{\partial K} &= (f, w)_K,
\end{aligned}
$$



- *(Local) Discontinuous Galerkin (DG)*

$$
\begin{aligned}
(\kappa^{-1}\mathrm{q}_h, \mathrm{v})_K - (u_h, \nabla\cdot\mathrm{v})_K + \langle \hat{u}_h, \mathrm{v}\cdot\mathrm{n}\rangle_{\partial K} &= 0, \\
(\nabla\cdot\mathrm{q}_h, w)_K - (\mathrm{c}u_h, \nabla w)_K + \langle(\widehat{\mathrm{c}u_h} - \widehat{\mathrm{q}}_h)\cdot\mathrm{n}, w\rangle_{\partial K} &= (f, w)_K,
\end{aligned}
$$



RTDG solution

- `(Par)DarcyForm`
  - Constructs B^T operator/matrix
  - Constructs `BlockOperator` (`Mult`, `MultTranspose`)
  - Elimination of essential BCs/DOFs

- Schur complement preconditioner (`DarcyOperator`)

$$\begin{bmatrix} A_d^{-1} & \\ & (D + BA_d^{-1}B^T)_{GS}^{-1} \end{bmatrix} \begin{bmatrix} A & -B^T \\ B & D \end{bmatrix}$$

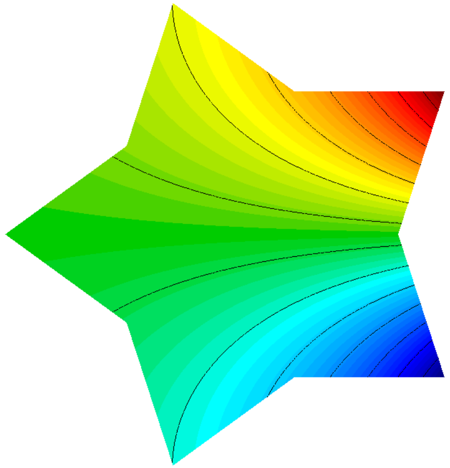
- No convection, steady-state → saddle-point problem (*D* = *0*)
- Anisotropy → direct solver (*SuiteSparse* – UMFPACK)
- (Elim. of potential → `DarcyReduction` (~~convection~~, ~~steady~~):

```
void EnablePotentialReduction(const Array<int> &ess_flux_tdof_list)
```
)

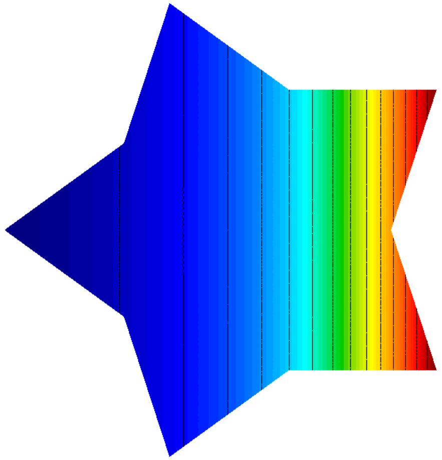
$$(A + B^T D^{-1} B) q_h = B^T f$$

Example 5 - RTDG (ex5.cpp)

```
DarcyForm *darcy = new DarcyForm(R_space, W_space,
                                 false);

BilinearForm *mVarf = darcy->GetFluxMassForm();
MixedBilinearForm *bVarf = darcy->GetFluxDivForm();

mVarf->AddDomainIntegrator(
       new VectorFEMassIntegrator(kcoeff));


ConstantCoefficient cdiv(-1.);
bVarf->AddDomainIntegrator(
       new VectorFEDivergenceIntegrator(cdiv));

if (pa) { darcy->SetAssemblyLevel(
                     AssemblyLevel::PARTIAL); }


darcy->Assemble();
```



Potential



Flux

Local Discontinuous Galerkin (LDG)

$$(\kappa^{-1}\mathrm{q}_h, \mathrm{v})_K - (u_h, \nabla\cdot\mathrm{v})_K + \langle \hat{u}_h, \mathrm{v}\cdot\mathrm{n}\rangle_{\partial K} = 0, \quad \forall \mathrm{v} \in (\mathcal{P}^p(K))^d,$$

$$-(\mathrm{c}u_h + \mathrm{q}_h, \nabla w)_K + \langle(\widehat{\mathrm{c}u_h} + \widehat{\mathrm{q}}_h)\cdot\mathrm{n}, w\rangle_{\partial K} = (f, w)_K, \quad \forall w \in \mathcal{P}^p(K).$$

- Traces definition → local stabilization

$$\widehat{\mathrm{q}}_h = \{\{\mathrm{q}_h\}\} + C_{11}[\![u_h\mathrm{n}]\!] + C_{12}[\![\mathrm{q}_h\cdot\mathrm{n}]\!],$$
$$\lambda_h = \hat{u}_h = \{\{u_h\}\} - C_{12}\cdot[\![u_h\mathrm{n}]\!] + C_{22}[\![\mathrm{q}_h\cdot\mathrm{n}]\!],$$

- LDG: C_{22}=0 (flux elimination → **DarcyReduction** $(D + BA^{-1}B^T)u_h = f$)
- Centered scheme: C_{12}=0, $C_{11}=\kappa h^{-1}/2$

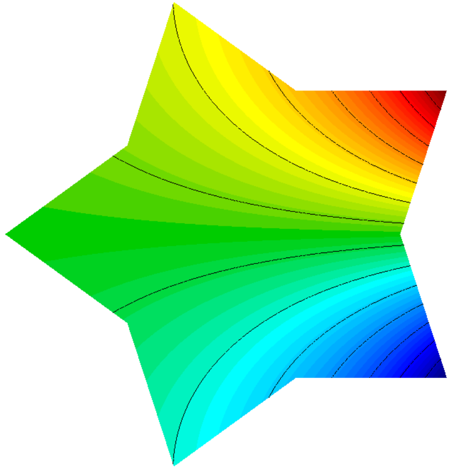
$$(\kappa^{-1}q_h, v) - (u_h, \nabla\cdot v) + \langle\{\{u_h\}\}, [\![v\cdot n]\!]\rangle = 0,$$

$$(\nabla\cdot q_h, w) - \langle[\![q_h\cdot n]\!], \{\{w\}\}\rangle + \langle\frac{\kappa h^{-1}}{2}[\![u_h]\!], [\![v]\!]\rangle = (f, w)$$

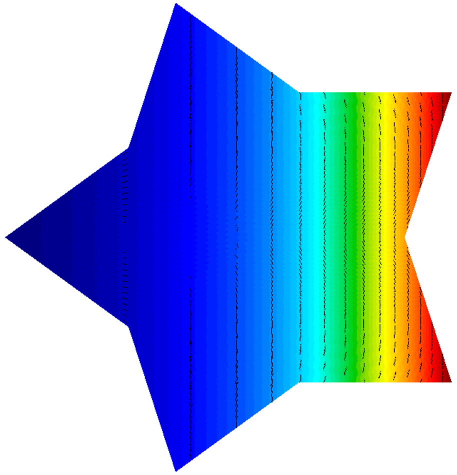
≈DG diffusion

Example 5 – LDG (ex5-hdg.cpp)

```
DarcyForm *darcy = new DarcyForm(R_space, W_space);

BilinearForm *mVarf = darcy->GetFluxMassForm();
MixedBilinearForm *bVarf = darcy->GetFluxDivForm();
BilinearForm *mtVarf = GetPotentialMassForm();

mVarf->AddDomainIntegrator(new
              VectorMassIntegrator(kcoeff));
bVarf->AddDomainIntegrator(new
              VectorDivergenceIntegrator());
bVarf->AddInteriorFaceIntegrator(new
              TransposeIntegrator(new
                DGNormalTraceIntegrator(-1.)));
mtVarf->AddInteriorFaceIntegrator(new
              HDGDiffusionIntegrator(ikcoeff));

darcy->Assemble();
```



Potential

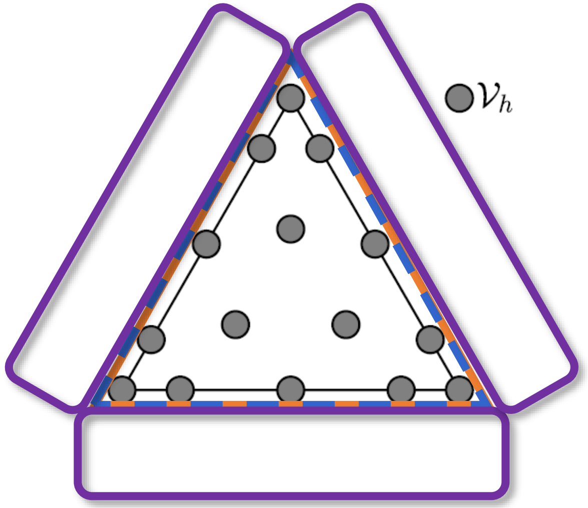


Flux

Hybridization

- Lagrange multipliers $\lambda_h \approx \hat{u}_h$
- Weak continuity of total flux

$$\langle [\![(\widehat{c u}_h + \widehat{\mathrm{q}}_h) \cdot \mathrm{n}]\!], \mu \rangle_{\mathcal{E}_h} = 0 \qquad , \quad \forall \mu$$



Credit: G. Giorgiani et al. / CPC 254 (2020) 107375

- *N.C. Nguyen, J. Peraire & B. Cockburn (2009), JCP, 228, 3232–3254.*
- Reduction to trace DOFs of λ_h (**`DarcyHybridization`**)
- `darcy-hdg-dev` - [PR #4350]

[WIP] Hybridization of mixed systems (HRT, HDG) [darcy-hdg-dev] #4350

Open najlkin wants to merge 497 commits into `master` from `darcy-hdg-dev`

Hybridized Raviart-Thomas (HRT)

- Lagrange mulitpler $\lambda_h \approx \hat{u}_h$ (`EnableHybridization()`)

$$(\kappa^{-1}\mathrm{q}_h, \mathrm{v})_{\mathcal{T}_h} - (u_h, \nabla\cdot\mathrm{v})_{\mathcal{T}_h} + \langle \lambda_h, \mathrm{v}\cdot\mathrm{n}\rangle_{\partial\mathcal{T}_h} = 0 \quad , \quad \forall \mathrm{v} \in V_h^p,$$

$$(\nabla\cdot\mathrm{q}_h, w)_{\mathcal{T}_h} = (f, w)_{\mathcal{T}_h} \quad , \quad \forall w \in W_h^p,$$

$$\langle [\![\widehat{\mathrm{q}}_h \cdot \mathrm{n}]\!], \mu\rangle_{\mathcal{E}_h} = 0 \quad , \quad \forall \mu \in M_h^p(0).$$

- Reduction of the system: (`FormLinearSystem()`)

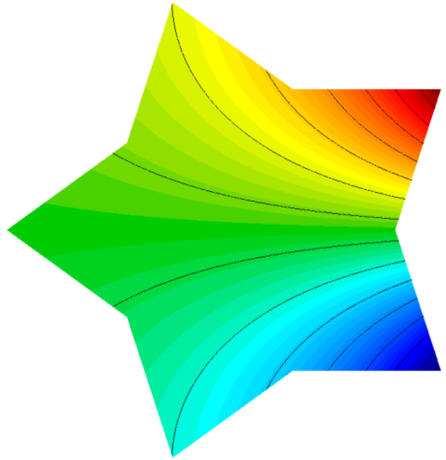
$$\begin{bmatrix} A & -B^T & C^T \\ B & 0 & 0 \\ C & 0 & 0 \end{bmatrix} \begin{bmatrix} Q \\ U \\ \Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ F \\ 0 \end{bmatrix}. \Rightarrow \mathbb{K} = -[C \ \ 0] \begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix}^{-1} \begin{bmatrix} C^T \\ 0 \end{bmatrix}, \quad \mathbb{F} = -[C \ \ 0] \begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ F \end{bmatrix}. \Rightarrow \boxed{\mathbb{K}\Lambda = \mathbb{F},}$$

- Recovery of the solution: (`RecoverFEMSolution()`)

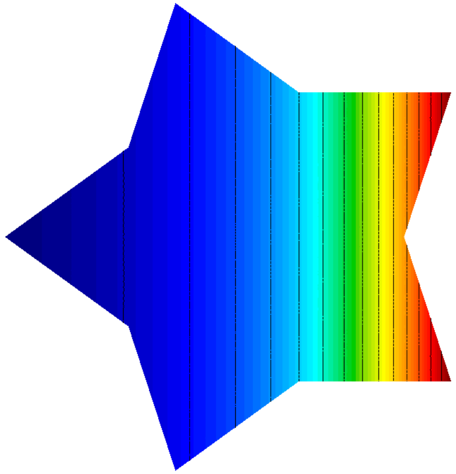
$$\begin{bmatrix} Q \\ U \end{bmatrix} = \begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix}^{-1} \left(\begin{bmatrix} 0 \\ F \end{bmatrix} - \begin{bmatrix} C^T \\ 0 \end{bmatrix} \Lambda \right),$$

Example 5 – HRT (ex5.cpp / ex5-hdg.cpp)

```
…

if (hybridization)
{
   trace_coll = new DG_Interface_FECollection(
                        order, dim);
   trace_space = new FiniteElementSpace(
                        mesh, trace_coll);
   darcy->EnableHybridization(trace_space,
                        new NormalTraceJumpIntegrator(),
                        ess_flux_tdofs_list);
}
darcy->Assemble();
```



Potential



Flux

Hybridizable Discontinuous Galerkin (HDG)

- Lagrange mulitplier $\lambda_h \approx \hat{u}_h$

$$(\kappa^{-1}\mathrm{q}_h, \mathrm{v})_{\mathcal{T}_h} - (u_h, \nabla\cdot\mathrm{v})_{\mathcal{T}_h} + \langle \lambda_h, \mathrm{v}\cdot\mathrm{n}\rangle_{\partial\mathcal{T}_h} = 0 \quad , \quad \forall \mathrm{v} \in \mathrm{V}_h^p,$$

$$-(\mathrm{c}u_h + \mathrm{q}_h, \nabla w)_{\mathcal{T}_h} + \langle (\widehat{\mathrm{c}u}_h + \widehat{\mathrm{q}}_h)\cdot\mathrm{n}, w\rangle_{\partial\mathcal{T}_h} = (f, w)_{\mathcal{T}_h} \quad , \quad \forall w \in W_h^p,$$

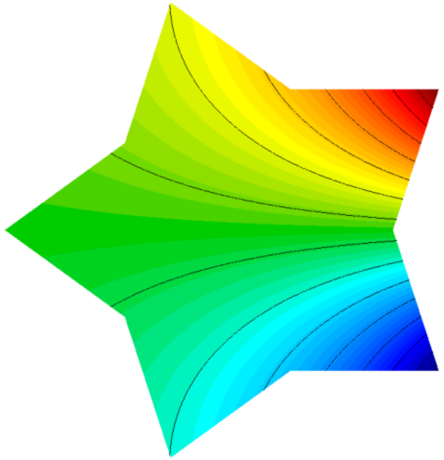
$$\langle [\![(\widehat{\mathrm{c}u}_h + \widehat{\mathrm{q}}_h)\cdot\mathrm{n}]\!], \mu\rangle_{\mathcal{E}_h} = 0 \quad , \quad \forall \mu \in M_h^p(0).$$

- *N.C. Nguyen, J. Peraire & B. Cockburn (2009), JCP, 228, 3232–3254.*
- Stabilization: $\widehat{\mathrm{c}u}_h + \widehat{\mathrm{q}}_h = \mathrm{c}\hat{u}_h + \mathrm{q}_h + \tau(u_h - \hat{u}_h)\mathrm{n},$ → Centered/upwinded
- Reduction of the system:

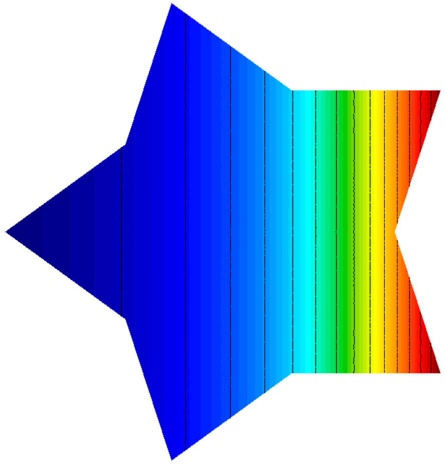
$$\begin{bmatrix} A & -B^T & C^T \\ B & D & E \\ C & G & H \end{bmatrix} \begin{bmatrix} \mathrm{Q} \\ U \\ \Lambda \end{bmatrix} = \begin{bmatrix} R \\ F \\ L \end{bmatrix}. \Rightarrow \begin{array}{l} \mathbb{K} = -[C \quad G] \begin{bmatrix} A & -B^T \\ B & D \end{bmatrix}^{-1} \begin{bmatrix} C^T \\ E \end{bmatrix} + H, \\ \mathbb{F} = L - [C \quad G] \begin{bmatrix} A & -B^T \\ B & D \end{bmatrix}^{-1} \begin{bmatrix} R \\ F \end{bmatrix}. \end{array} \Rightarrow \boxed{\mathbb{K}\,\Lambda = \mathbb{F},}$$

Example 5 – HDG (ex5-hdg.cpp)

```
…
BilinearForm *mtVarf = GetPotentialMassForm();
…
mtVarf->AddInteriorFaceIntegrator(new
              HDGDiffusionIntegrator(ikcoeff));

if (hybridization)
{
   trace_coll = new DG_Interface_FECollection(
                    order, dim);
   trace_space = new FiniteElementSpace(
                    mesh, trace_coll);
   darcy->EnableHybridization(trace_space,
                    new NormalTraceJumpIntegrator(),
                    ess_flux_tdofs_list);
}

darcy->Assemble();
```



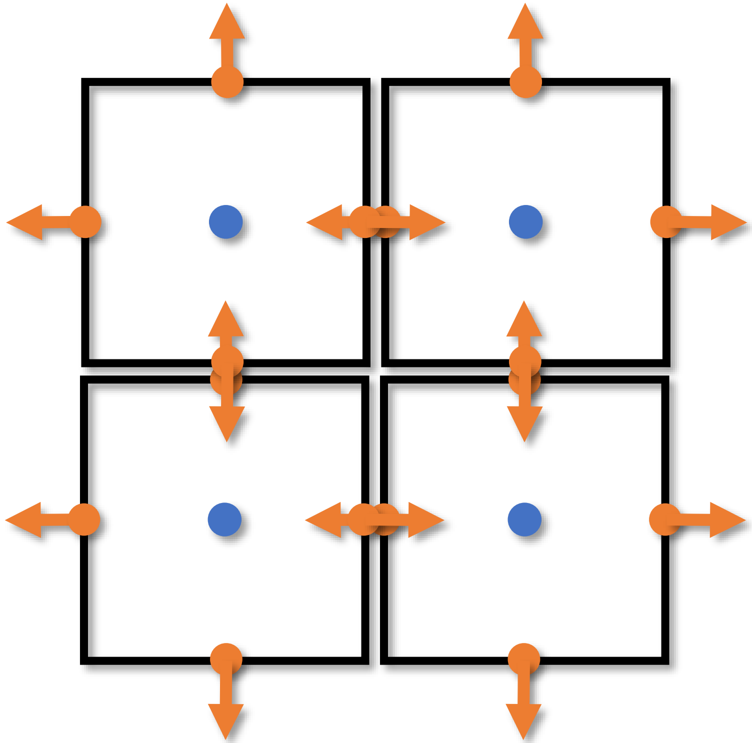
Potential



Flux

H/LBRT – broken RT

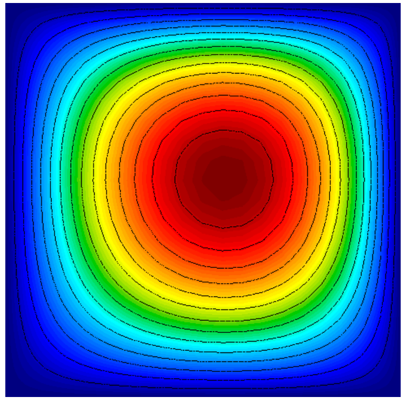
- **Broken Raviart-Thomas** (`BrokenRT_FECollection`)
- *H. Egger, J. Schöberl (2010), IMA J. Numer. Anal., 30, 1206–1234.*
- Local compatibility → *No stabilization*
- Convection-diffusion
- LDG-style system (LBRT)



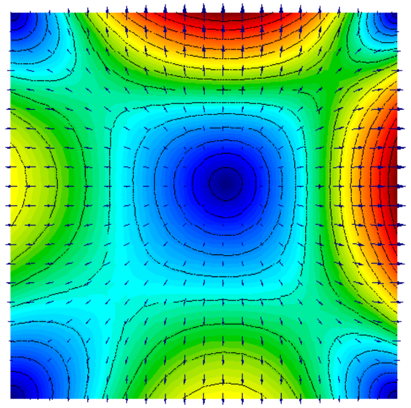
Convection-diffusion (ex5-nguyen.cpp)

- Problem 2 (-p 2) – steady advection-diffusion

```
…
BilinearForm *Mt = GetPotentialMassForm();
…

Mt->AddDomainIntegrator(
     new ConservativeConvectionIntegrator(ccoeff));

if (upwinded) {
    Mt->AddInteriorFaceIntegrator(
      new HDGConvectionUpwindedIntegrator(ccoeff));
} else {
    Mt->AddInteriorFaceIntegrator(
      new HDGConvectionCenteredIntegrator(ccoeff));
   }
}
```

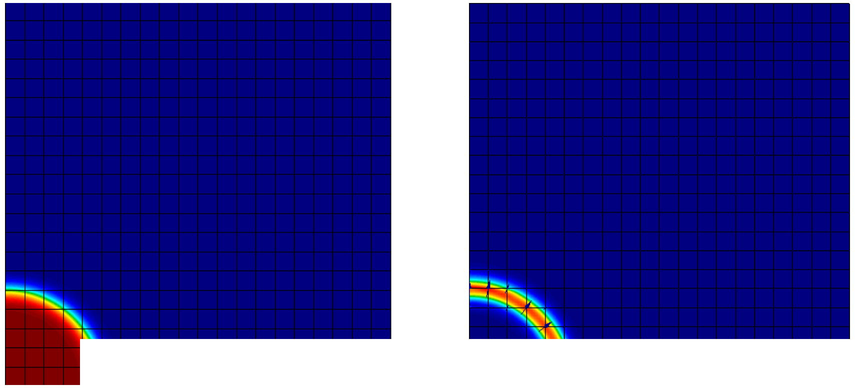


Potential



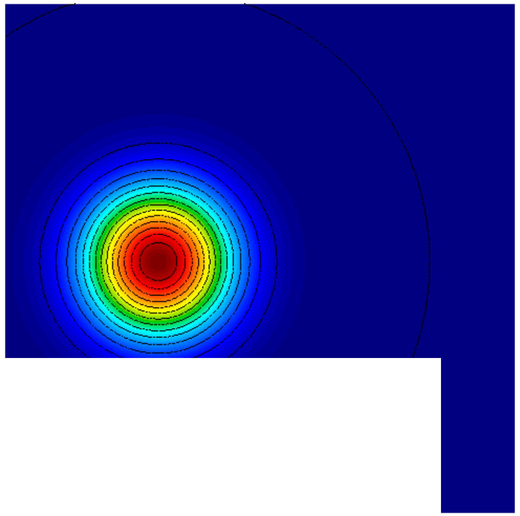
Flux

Example 5 – Nguyen (ex5-nguyen.cpp)



Potential *Flux*

Problem 3 – steady advection

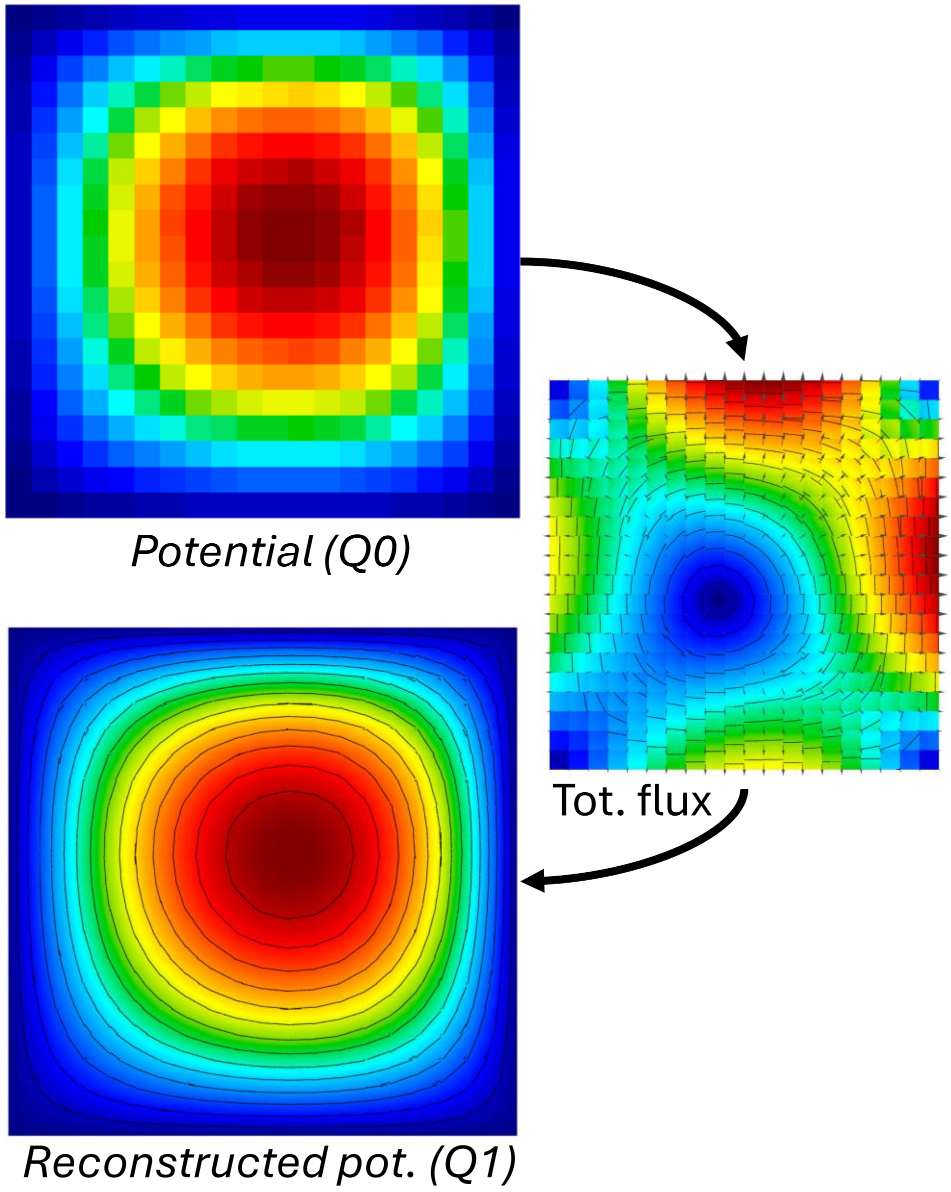


Problem 4 – non-steady advection(-diffusion)

Problem 5 – Kovasznay flow

Superconvergent reconstruction

- DG diff. flux (+ DG conv. flux) + HDG trace
  → **RT total flux**
- `DarcyForm::ReconstructTotalFlux()`
  - Auto FE space construction
  - Auto velocity recognition from conv. integs.
- RT total flux (+ DG potential) →
  superconvergent diff. flux + potential
- `DarcyForm::ReconstructFluxAndPot()`
  - Auto FE space construction
  - Auto (non-)steady case treatment

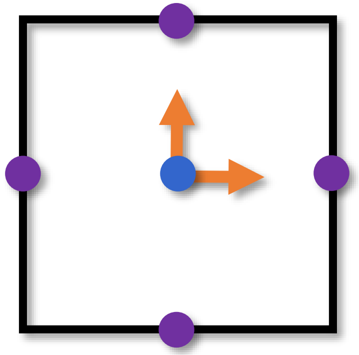


Non-linear convection

- Non-linear flux ***F**(u)*

$$\begin{aligned} \boldsymbol{q} + \kappa \nabla u &= 0, && \text{in} \quad \Omega, \\ \nabla \cdot (\boldsymbol{q} + \boldsymbol{F}(u)) &= f, && \text{in} \quad \Omega, \end{aligned}$$

- `HyperbolicFormIntegrator` + `NumericalFlux`
  - `RusanovFlux`
  - `ComponentwiseUpwindFlux`
  - (`HDGFlux` – HDG-I / HDG-II schemes)
- *N.C. Nguyen, J. Peraire & B. Cockburn (2009), JCP, 228, 8841–8855.*



- `DarcyHybridization` → `Operator`

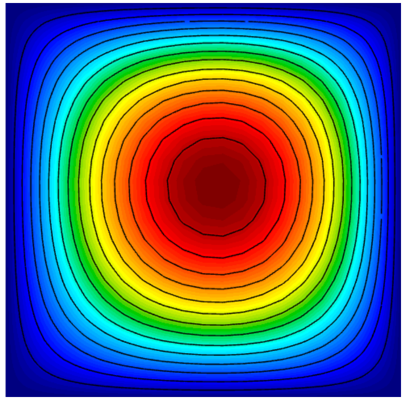
$$-[C \quad G]\begin{bmatrix} A & -B^T \\ B & D \end{bmatrix}^{-1} \begin{bmatrix} C^T \\ E \end{bmatrix} \Lambda \; + H\Lambda$$

- Global+Local solver (LBFGS/LBB/Newton)

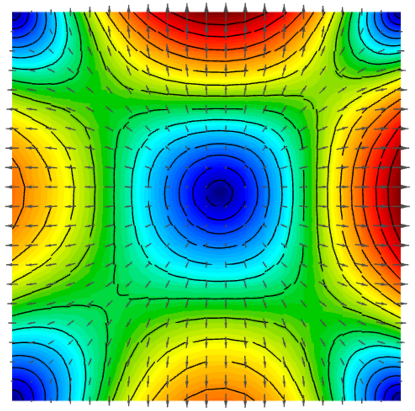
Burgers + diffusion (ex5-nguyen.cpp)

- Problem 6 (-p 6) – steady Burgers-diffusion

```
NonlinearForm *Mtnl = darcy->GetPotentialMassNonlinearForm();
…
FluxFun = new BurgersFlux(ccoef.GetVDim());

switch (hdg_scheme)
{
case 1: FluxSolver = new HDGFlux(*FluxFun,
           HDGFlux::HDGScheme::HDG_1); break;
case 2: FluxSolver = new HDGFlux(*FluxFun,
           HDGFlux::HDGScheme::HDG_2); break;
case 3: FluxSolver = new RusanovFlux(*FluxFun); break;
case 4: FluxSolver = new ComponentwiseUpwindFlux(*FluxFun);
break;
}

Mtnl->AddDomainIntegrator(
           new HyperbolicFormIntegrator(*FluxSolver, 0, -1.));

Mtnl->AddInteriorFaceIntegrator(
           new HyperbolicFormIntegrator(*FluxSolver, 0, -1.));
```



Potential



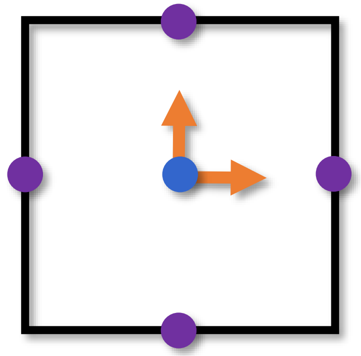
Flux

Non-linear diffusion

- Non-linear conductivity *κ(u)*

$$\begin{aligned} \mathbf{q} + \kappa(u)\nabla u &= 0, \\ \nabla \cdot \mathbf{q} &= 0 \end{aligned}$$

- `MixedConductionNLFIntegrator` + `MixedFluxFunction`
  - `LinearDiffusionFlux`
  - `FunctionDiffusionFlux`

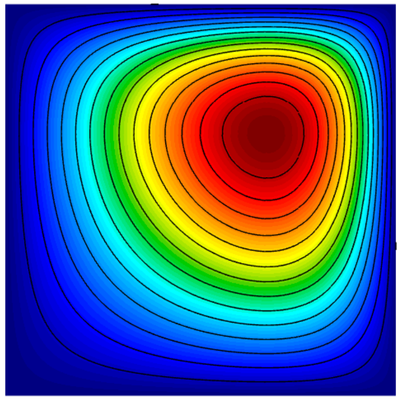


- `BlockNonlinearForm + BlockOperator`
- Global+Local solver (LBFGS/LBB/Newton)

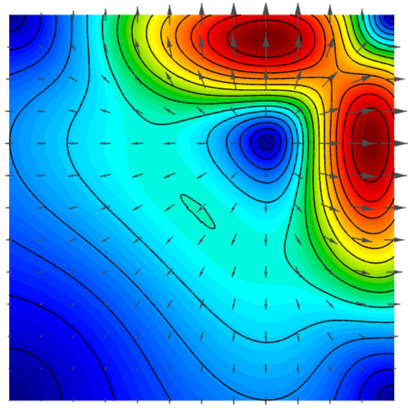
Non-linear diffusion (ex5-nguyen.cpp)

- Problem 8 (-p 8) – lin. conductivity (*κ(u) = k+u*)

```
BlockNonlinearForm *Mnl = darcy->GetBlockNonlinearForm();
…
auto ikappa = [=](const Vector &x, real_t u)
                  { return 1./(k+u); };
auto dikappa = [=](const Vector &x, real_t u)
                  { return -1./((k+u)*(k+u)); };

 HeatFluxFun = new FunctionDiffusionFlux(dim, ikappa, dikappa);

 Mnl->AddDomainIntegrator(
         new MixedConductionNLFIntegrator(*HeatFluxFun));

 if (upwinded) {
     Mnl->AddInteriorFaceIntegrator(
         new MixedConductionNLFIntegrator(*HeatFluxFun, ccoef));
 } else {
     Mnl->AddInteriorFaceIntegrator(
         new MixedConductionNLFIntegrator(*HeatFluxFun));
 }
```



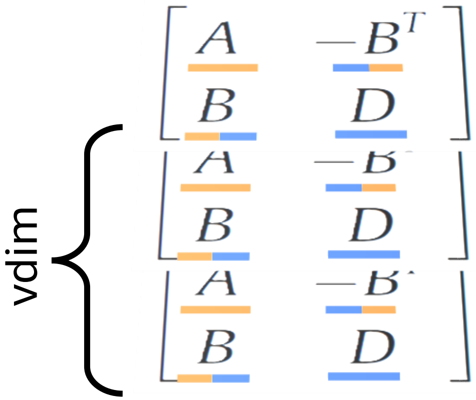
Potential

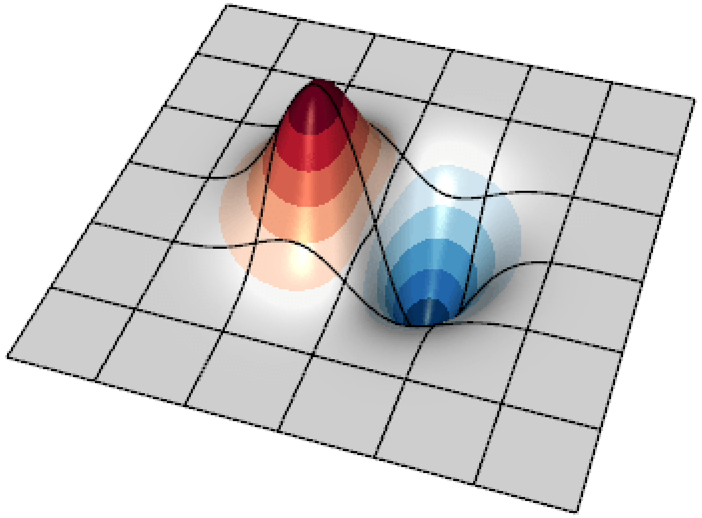


Flux

Systems of equations [WIP]

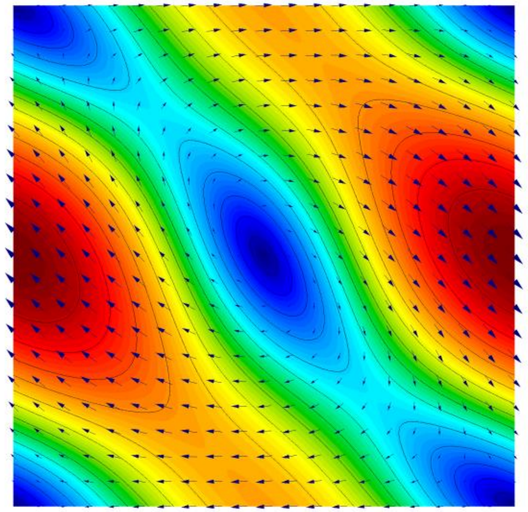
- Vector dimension
- `VectorBlockDiagonalIntegrator`
- *Implicit* Euler equations with diffusion (`ex18-hdg.cpp`) – `HyperbolicFormIntegrator` + `EulerFlux` + `DarcyOperator`
- Status:
  - Mixed ✓
  - Reduced (linear) ✓
  - Hybridized [WIP]



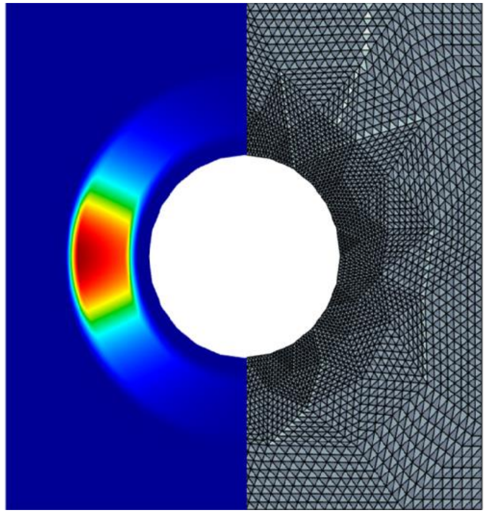


Anisotropic diffusion (`ex5-aniso.cpp`)

- Problems:
  - Stationary/asymptotic diffusion
  - MFEM text random conv-diffusion
  - Diffusion ring arc/Gauss/sine
  - Boundary layer
  - Steady peak/varying angle
  - Sovinec
  - Umansky



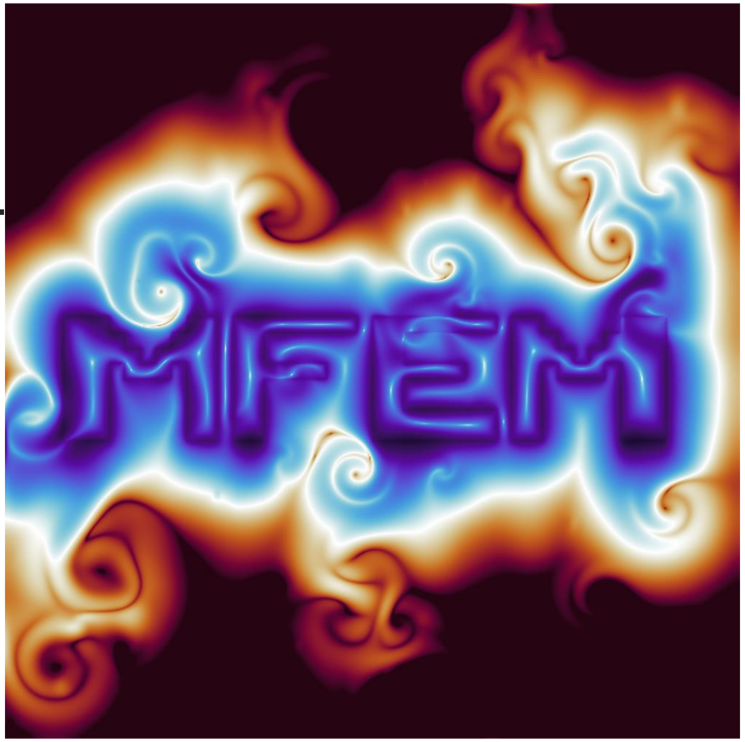
Steady diffusion (`-p 1`)



Diffusion ring (`-p 3`)

Conclusions

- Framework for mixed systems – (Par)DarcyForm
- Potential/flux reduction (DarcyReduction)
- Total flux hybridization (DarcyHybridization) – reduced system, preconditioning, convergence, stabilization, …
- Superconv. reconstruction – H(div) total flux
- Non-linear convection / diffusion
- Systems of equations [WIP]
- TODOs – miniapps, GPUs, Maxwell, …



Single-step anisotropic diffusion-convection simulation

Thank you for your attention

CASC
Center for Applied
Scientific Computing