Hybridization of convection-diffusion systems in MFEM

MFEM Community Workshop 2024

Jan Nikl
Center for Applied Scientific Computing (CASC),
Lawrence Livermore National Laboratory, CA, USA

October 22, 2024

LLNL-PRES-870699

Mixed systems

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

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

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

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

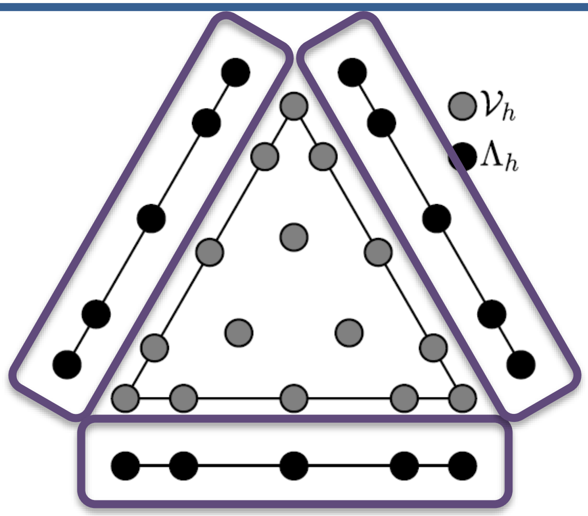
$$\begin{bmatrix} A & -B^T \\ B & D \end{bmatrix}$$

Hybridization

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

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

- *N.C. Nguyen, J. Peraire & B. Cockburn (2009), JCP, 228, 3232–3254.*
- Hybridizable Discontinuous Galerkin (HDG) method
  - Efficiency
  - Convergence rate
  - Preconditioning
  - Direct → iterative
- `darcy-hdg-dev` - [PR #4350]



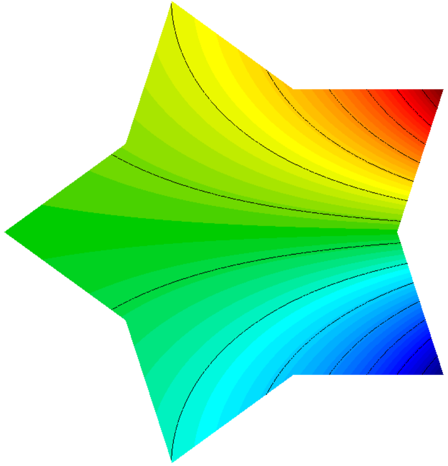
Credit: G. Giorgiani et al. / CPC 254 (2020) 107375

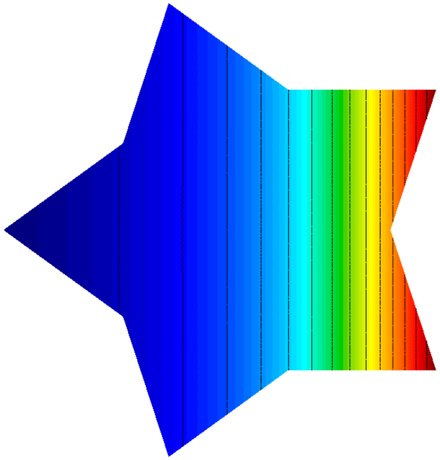
DarcyForm

- Constructor: `DarcyForm(FiniteElementSpace *fes_u, FiniteElementSpace *fes_p, bool symmetrize = true);`
- Constructs B^T operator/matrix
- Constructs `BlockOperator` (`Mult`, `MultTranspose`)
- (Elimination of potential: `void EnablePotentialReduction(const Array<int> &ess_flux_tdof_list)`)
- Elimination of essential BCs/DOFs
- `void FormLinearSystem(const Array<int> &ess_flux_tdof_list, BlockVector &x, BlockVector &b, OperatorHandle &A, Vector &X, Vector &B, int copy_interior = 0);`

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();
```





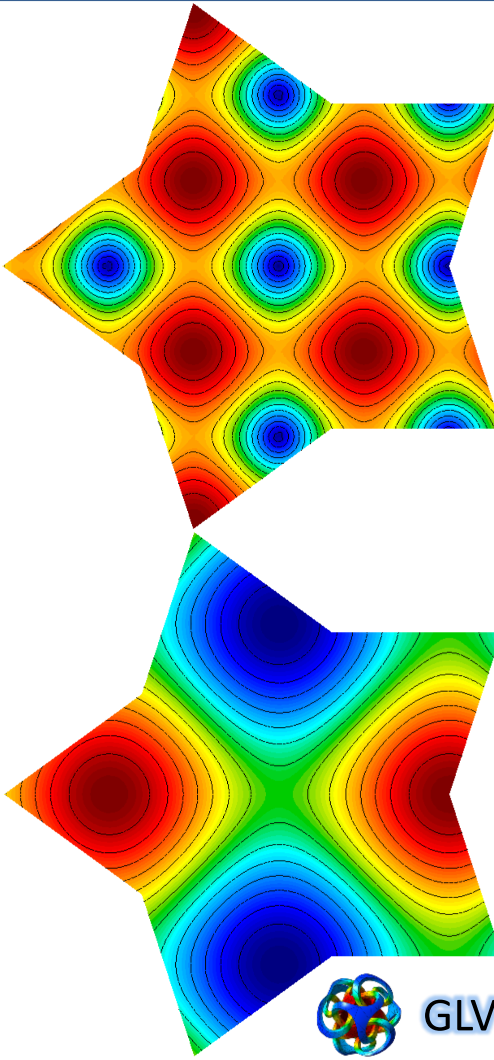
Bonus ★: Example 5 – Maxwell (`ex5-max.cpp`)

- Example 3 – mixed (definite) formulation:

$$\sigma E - \nabla \times B = f$$

$$\nabla \times E + \qquad B = g$$

```
BilinearForm *mEVarf =
        darcy->GetFluxMassForm();
MixedBilinearForm *cVarf =
        darcy->GetFluxDivForm();
BilinearForm *mBVarf =
        darcy->GetPotentialMassForm();

mEVarf->AddDomainIntegrator(
        new VectorFEMassIntegrator(sigma));
cVarf->AddDomainIntegrator(
        new MixedScalarCurlIntegrator());
mBVarf->AddDomainIntegrator(
        new MassIntegrator());
```



GLVis 4.3

Local Discontinuous Galerkin (LDG)

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

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

- Mixed face integration ([#4123](#4123))
- Traces definition → local stabilization

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

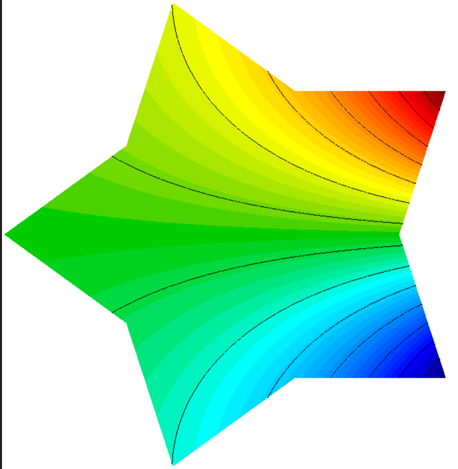
- LDG: C_{22}=0 (flux elimination – `DarcyForm::EnableFluxReduction()`)
- Centered scheme: C_{12}=0, $C_{11}=\kappa h^{-1}/2$

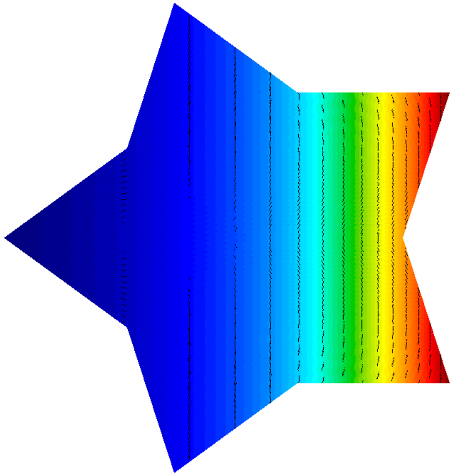
$$(\kappa^{-1} q_h, v) - (u_h, \nabla \cdot v) + \langle \{\{u_h\}\}, [\![v \cdot n]\!] \rangle = 0,$$
$$(\nabla \cdot q_h, w) - \langle [\![q_h \cdot n]\!], \{\{w\}\} \rangle + \langle \frac{\kappa h^{-1}}{2} [\![u_h]\!], [\![v]\!] \rangle = (f, w)$$

≈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();
```





Hybridized Raviart-Thomas (HRT)

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

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

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

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

- Reduction of the system:

$$\begin{bmatrix} A & -B^T & C^T \\ B & 0 & 0 \\ C & 0 & 0 \end{bmatrix} \begin{bmatrix} Q \\ U \\ \Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ F \\ 0 \end{bmatrix}.$$

$$\mathbb{K} = -\begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix}^{-1} \begin{bmatrix} C^T \\ 0 \end{bmatrix},$$

$$\mathbb{F} = -\begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ F \end{bmatrix}.$$

$$\mathbb{K}\,\Lambda = \mathbb{F},$$

- Recovery of the solution:

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

DarcyHybridization

- Integrated with `DarcyForm`:

```
void EnableHybridization(
FiniteElementSpace *constr_space,
BilinearFormIntegrator *constr_flux_integ,
const Array<int> &ess_flux_tdof_list)
```

- Constraint integrator: `NormalTraceJumpIntegrator`

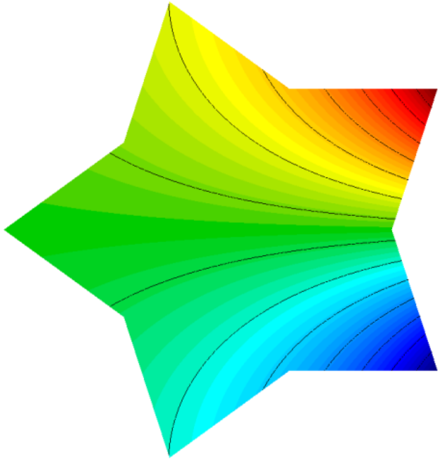
$$\langle [\![\widehat{q}_h \cdot n]\!], \mu \rangle_{\mathcal{E}_h} = 0 \qquad , \quad \forall \mu$$

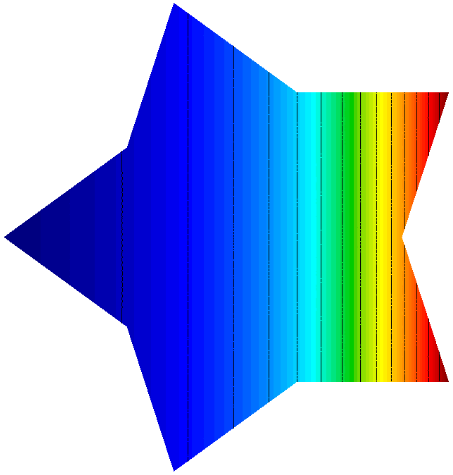
- `FormLinearSystem()` → Hybridized matrix

- `RecoverFEMSolution()` → Recovers q_h, u_h

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

```
...

if (hybridization)
{
   trace_coll = new RT_Trace_FECollection(
                   order, dim, 0);
   trace_space = new FiniteElementSpace(
                   mesh, trace_coll);
   darcy->EnableHybridization(trace_space,
                   new NormalTraceJumpIntegrator(),
                   ess_flux_tdofs_list);
}

darcy->Assemble();
```





Hybridizable Discontinuous Galerkin (HDG)

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

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

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

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

- *N.C. Nguyen, J. Peraire & B. Cockburn (2009), JCP, 228, 3232–3254.*

- Reduction of the system:

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

HDG – stabilization

- Stabilization parameter τ (double-valued!)

$$\widehat{\mathbf{c}u_h} + \widehat{\mathbf{q}}_h = \mathbf{c}\widehat{u}_h + \mathbf{q}_h + \tau(u_h - \widehat{u}_h)\mathbf{n},$$

$$a(\mathbf{q},\mathbf{v}) = (\kappa^{-1}\mathbf{q},\mathbf{v})_{\mathcal{T}_h},$$
$$b(u,\mathbf{v}) = (u, \nabla\cdot\mathbf{v})_{\mathcal{T}_h},$$
$$c(\lambda,\mathbf{v}) = \langle \lambda, \mathbf{v}\cdot\mathbf{n}\rangle_{\partial\mathcal{T}_h},$$
$$d(u,w) = -(\mathbf{c}u, \nabla w)_{\mathcal{T}_h} + \langle w, \tau u\rangle_{\partial\mathcal{T}_h},$$
$$e(\lambda,w) = \langle w, (\mathbf{c}\cdot\mathbf{n} - \tau)\lambda\rangle_{\partial\mathcal{T}_h},$$
$$g(\mu,u) = \langle \mu, \tau u\rangle_{\partial\mathcal{T}_h},$$
$$h(\mu,\lambda) = \langle \mu, (\mathbf{c}\cdot\mathbf{n} - \tau)\lambda\rangle_{\partial\mathcal{T}_h},$$
$$f(w) = (f, w)_{\mathcal{T}_h},$$
$$r(\mathbf{v}) = 0$$
$$\ell(\mu) = 0$$

- Centered scheme: $\tau_c^+ = \tau_c^- = |\mathbf{c}\cdot\mathbf{n}|, \quad \tau_d^+ = \tau_d^- = \dfrac{\kappa}{\ell},$
- Upwinded scheme: $(\tau_c^\pm, \tau_d^\pm) = \left(|\mathbf{c}\cdot\mathbf{n}|, \dfrac{\kappa}{\ell}\right) \dfrac{|\mathbf{c}\cdot\mathbf{n}^+| \pm \mathbf{c}\cdot\mathbf{n}^+}{2|\mathbf{c}\cdot\mathbf{n}^+|},$

DarcyHybridization

- Face integrator:

```
HDGDiffusionIntegrator(Coefficient &q, const
real_t a = 0.5)
```

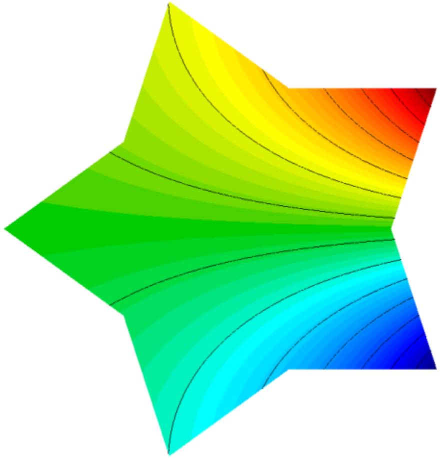
- Potential constraint: *face + constraint + flux + trace face matrix = „HDG face matrix“*

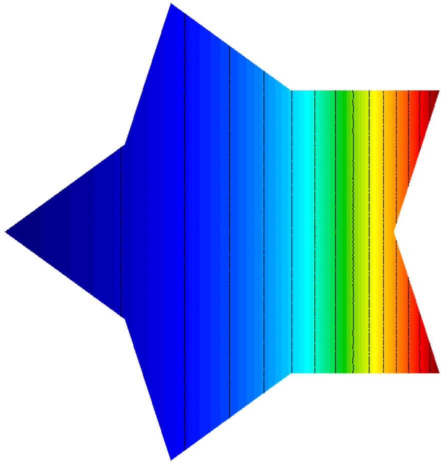
```
void AssembleHDGFaceMatrix(const FiniteElement &trace_el,
                           const FiniteElement &el1,
                           const FiniteElement &el2,
                           FaceElementTransformations &Trans,
                           DenseMatrix &elmat);
```

$$\begin{bmatrix} D_1 & & E_1 \\ & D_2 & E_2 \\ G_1 & G_2 & H_{12} \end{bmatrix}$$

Example 5 – HDG (ex5-hdg.cpp)

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

if (hybridization)
{
   trace_coll = new RT_Trace_FECollection(
                    order, dim, 0);
   trace_space = new FiniteElementSpace(
                    mesh, trace_coll);
   darcy->EnableHybridization(trace_space,
                    new NormalTraceJumpIntegrator(),
                    ess_flux_tdofs_list);
}

darcy->Assemble();
```





L/HDG – upwinding

- Upwinded diffusion:
  - Flux constraint: `DGNormalTraceIntegrator(VectorCoefficient &u_, real_t a)`
  - HDG face integrator (no LDG stabilization!): `HDGDiffusionIntegrator(VectorCoefficient &u_, Coefficient &q, const real_t a = 0.5)`
- Upwinded convection:

```
class HDGConvectionUpwindedIntegrator
: public DGTraceIntegrator
```

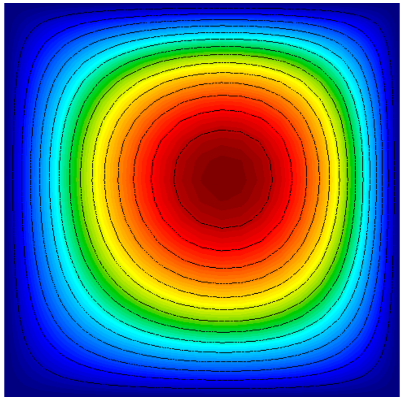
- (Centered convection:

```
class HDGConvectionCenteredIntegrator
: public DGTraceIntegrator
```
)

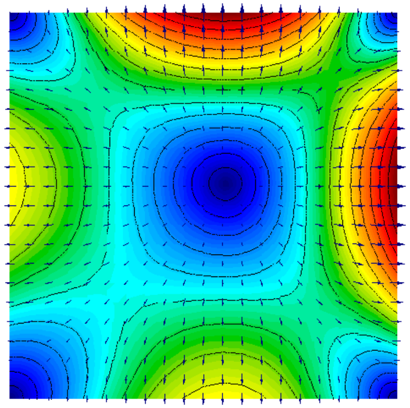
Example 5 – convection (ex5-nguyen.cpp)

- 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));
   }
}
```



Temperature

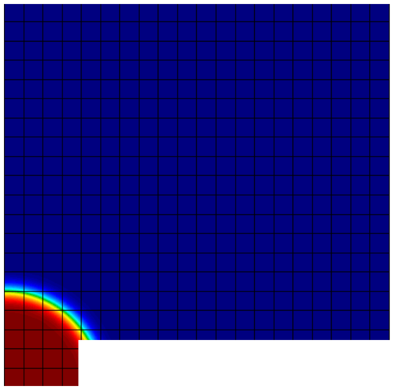


Heat flux

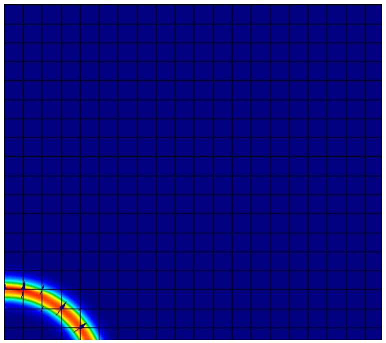
Boundary conditions

- Natural potential BCs – *RTDG, HRT, L/HDG*
  - Flux eq. – `Vector(FE)BoundaryFluxLFIntegrator` (#4082)
  - Pot. eq. – `HDGConvectionUpwinded/CenteredIntegrator` + `BoundaryFlowIntegrator`
  -
    ```
    void DarcyHybridization::AddBdrPotConstraintIntegrator(
    BilinearFormIntegrator *c_integ, Array<int> &bdr_marker)
    ```
  - (Centered HRT/HDG – diverging system! → full rhs flux, undefined λ_h)
- Essential flux BCs – *RTDG, HRT*
- Natural flux BCs – *L/HDG*
  - LDG – pot. eq. lhs + total flux rhs + flux eq. `darcy->GetFluxDivForm()->AddBdrFaceIntegrator(…)`
  - HDG – constraint $\langle [\![(\widehat{c u_h} + \widehat{q}_h) \cdot \mathrm{n}]\!], \mu \rangle_{\mathcal{E}_h} = \langle g_N, \mu \rangle_{\Gamma_N}$ (#4082)
    ```
    void Hybridization::AddBdrConstraintIntegrator(
    BilinearFormIntegrator *c_integ, Array<int> &bdr_marker)
    ```

Example 5 – Nguyen (ex5-nguyen.cpp)

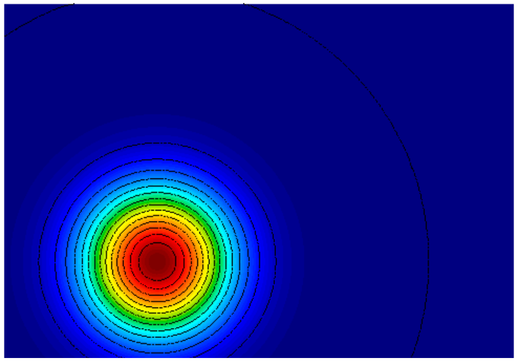


Temperature



Heat flux

Problem 3 – steady advection



Problem 4 – non-steady advection(-diffusion)

Problem 5 – Kovasznay flow

Non-linear convection

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

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

- (L)DG formulation

$$(\kappa^{-1} \boldsymbol{q}_h, \boldsymbol{v})_{\mathcal{T}_h} - (u_h, \nabla \cdot \boldsymbol{v})_{\mathcal{T}_h} + \langle \hat{u}_h, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial \mathcal{T}_h} = 0,$$

$$-(\boldsymbol{q}_h + \boldsymbol{F}(u_h), \nabla w)_{\mathcal{T}_h} + \left\langle \left(\hat{\boldsymbol{q}}_h + \widehat{\boldsymbol{F}}_h \right) \cdot \boldsymbol{n}, w \right\rangle_{\partial \mathcal{T}_h} = (f, w)_{\mathcal{T}_h},$$

- `HyperbolicFormIntegrator` + `RiemannSolver` 4.7
  - `RusanovFlux` – $\widehat{\boldsymbol{F} \cdot \boldsymbol{n}}^{LF}(a,b) = \frac{1}{2}(\boldsymbol{F}(a) + \boldsymbol{F}(b)) \cdot \boldsymbol{n} - \frac{C}{2}(b-a),$
  - `GodunovFlux` (#4513) – $\widehat{\boldsymbol{F} \cdot \boldsymbol{n}}^{G}(a,b) = \begin{cases} \min_{s \in [a,b]} \boldsymbol{F}(s) \cdot \boldsymbol{n}, & \text{if} \quad a \leqslant b, \\ \max_{s \in [b,a]} \boldsymbol{F}(s) \cdot \boldsymbol{n}, & \text{if} \quad a > b, \end{cases}$

Non-linear HDG

- HDG formulation

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

$$-(\boldsymbol{q}_h + \boldsymbol{F}(u_h), \nabla w)_{\mathcal{T}_h} + \left\langle \left(\hat{\boldsymbol{q}}_h + \widehat{\boldsymbol{F}}_h\right)\cdot\boldsymbol{n}, w\right\rangle_{\partial\mathcal{T}_h} = (f, w)_{\mathcal{T}_h},$$

$$\left\langle \left(\hat{\boldsymbol{q}}_h + \widehat{\boldsymbol{F}}_h\right)\cdot\boldsymbol{n}, \mu\right\rangle_{\partial\mathcal{T}_h} = 0,$$

- *N.C. Nguyen, J. Peraire & B. Cockburn (2009), JCP, 228, 8841–8855.*

- `RiemannSolver` (`Average()` – #4513)
  - HDG-I $\widehat{\boldsymbol{F}}_h = \boldsymbol{F}(\hat{u}_h) + C_\tau(u_h - \hat{u}_h)\boldsymbol{n},$
  - HDG-II $\widehat{\boldsymbol{F}}_h = \boldsymbol{F}(u_h) + C_\tau(u_h - \hat{u}_h)\boldsymbol{n},$
  - Rusanov, Godunov $\widehat{\boldsymbol{F}}_h\cdot\boldsymbol{n} = \frac{1}{u_h - \hat{u}_h}\int_{\hat{u}_h}^{u_h} \widehat{\boldsymbol{F}}\cdot\boldsymbol{n}(s, \hat{u}_h)ds,$

- `DarcyHybridization` → `Operator`

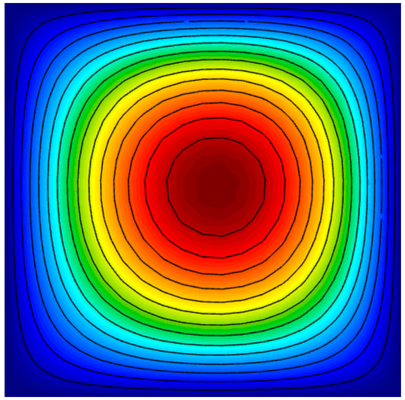
$$-\begin{bmatrix} C & G \end{bmatrix} \begin{bmatrix} A & -B^T \\ B & D \end{bmatrix}^{-1} \begin{bmatrix} C^T \\ E \end{bmatrix} \Lambda + H\Lambda$$

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

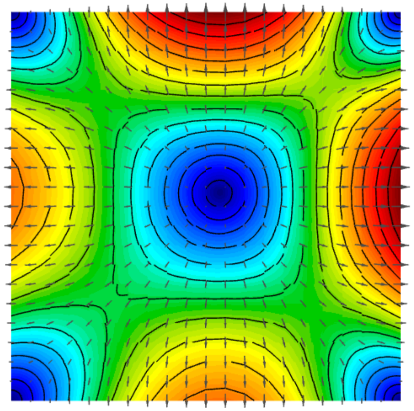
Example 5 – Burgers (ex5-nguyen.cpp)

- 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 GodunovFlux(*FluxFun); break;
}

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

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



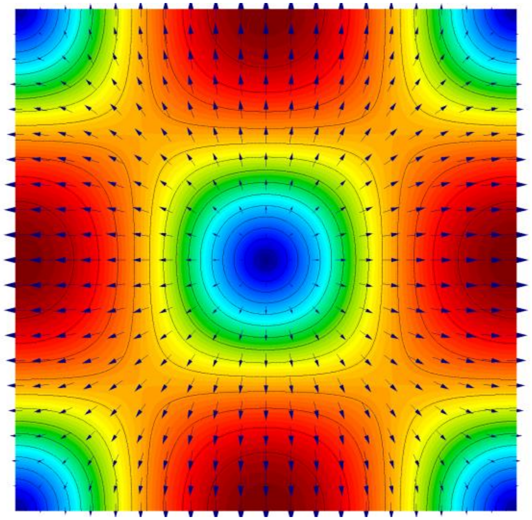
Temperature

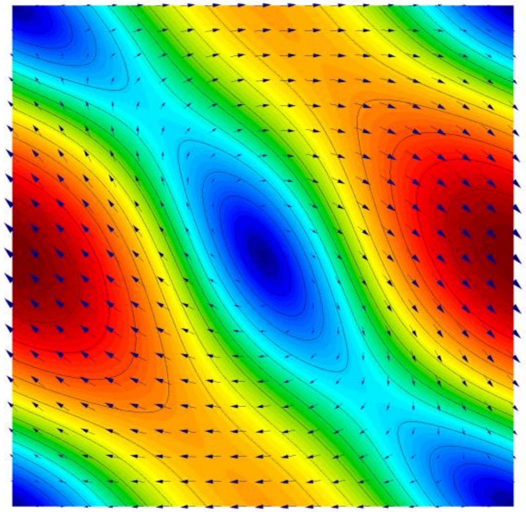


Heat flux

Example 5 – anisotropy (`ex5-heat.cpp / ex5-aniso.cpp`)

- Stationary/asymptotic heat conduction (problem `-p 1`)
- *T=sin(x)*sin(y)*
- Tensor heat conductivity
  – *sym.* + *antisym.* anisotropy
- 20x20 Q2 RT + L2 elements
- *Isotropic*: 47 HRT **x** 251 RTDG iters
- *Anisotropic* (10x sym. & antisym.):
  119 HRT iters **x** No convergence of RTDG

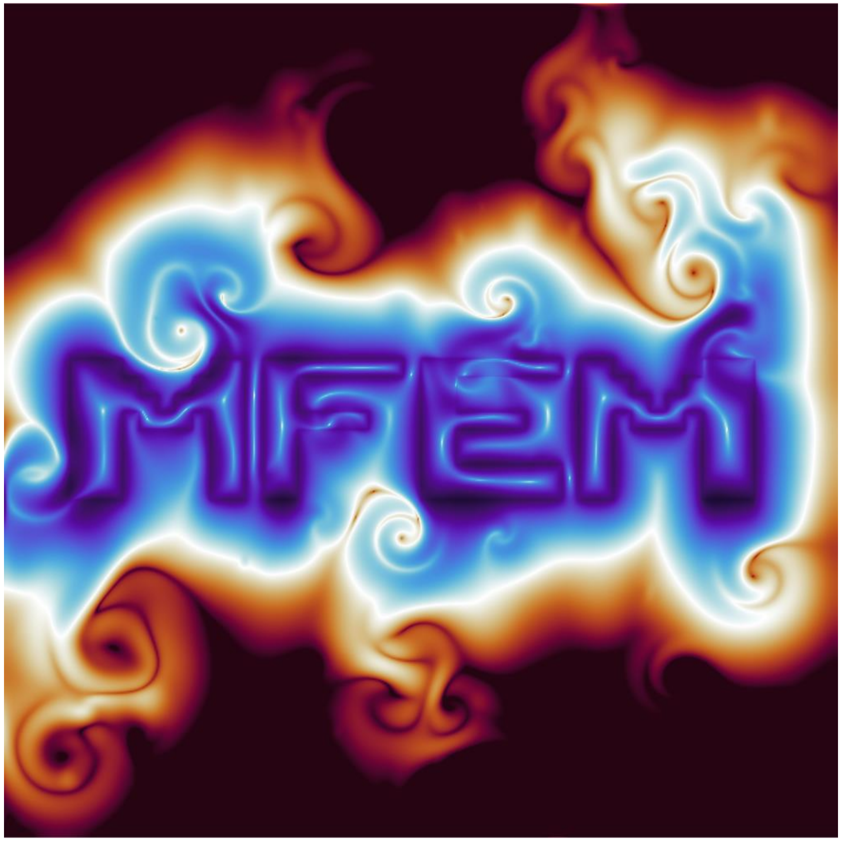




Example 5 – anisotropy (`ex5-aniso.cpp`)

- Problem 2 (`-p 2`) – MFEM logo *single-step* advection-diffusion
- 200 x 200 Q3 HDG ≈ *15 s* !

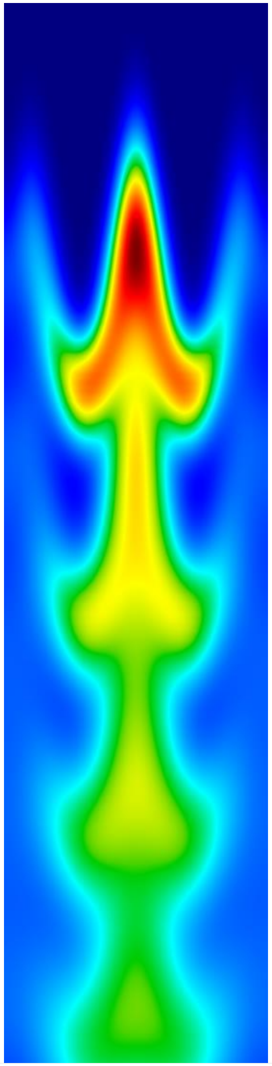
Temperature



Heat flux

Conclusions

- Framework for mixed systems – `DarcyForm`
- Total flux hybridization (ala *Nguyen & Cockburn*) – `DarcyHybridization`
- One-line hybridization – `DarcyForm + DarcyHybridization`
- HRT/HDG – Reduced system, preconditioning, convergence, stabilization, definitness, …
- Non-linear convection – Riemann solvers
- TODOs – non-linear diffusion, parallelization, systems of equations, reconstruction, …



Thank you for your attention

Thanks to C. Migliore *for regression testing and documentation*

Lawrence Livermore National Laboratory

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC