Efficient geometric multigrid methods for discontinuous Galerkin discretizations

Robert Saye

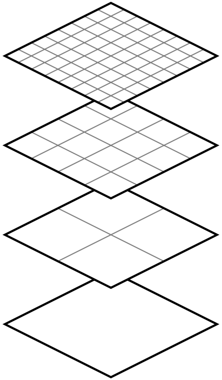
Dan Fortunato
Harvard

ICOSAHOM, July 9th 2018

Chris Rycroft

Introduction

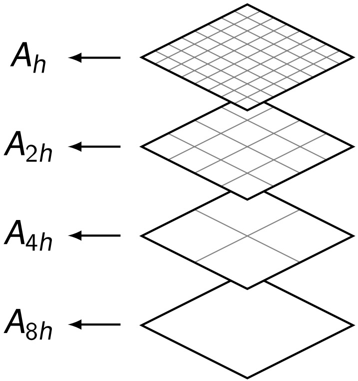
hp–multigrid works well for CG, but can perform poorly for DG... **why?**



Introduction

hp–multigrid works well for CG, but can perform poorly for DG... **why?**

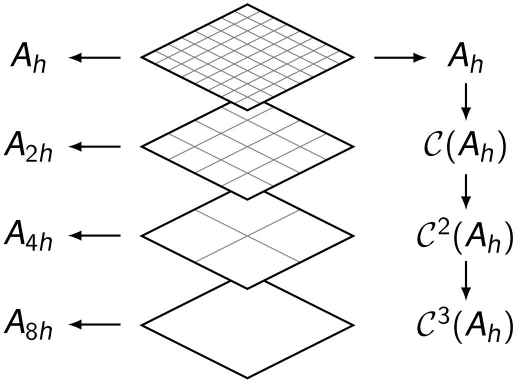
1. Discretize on hierarchy of meshes



Introduction

hp–multigrid works well for CG, but can perform poorly for DG... **why?**

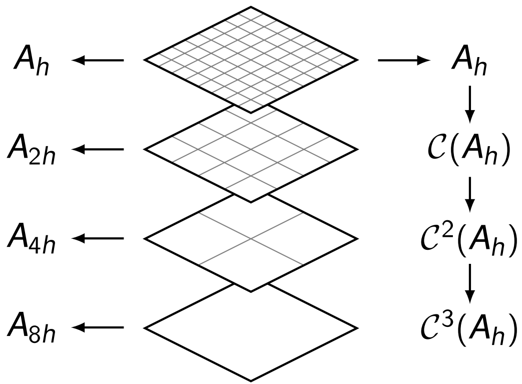
1. Discretize on hierarchy of meshes
2. Automatically build coarse operators: $\mathcal{C}(A_h) \to R_h^{2h} A_h I_{2h}^h$



Introduction

hp–multigrid works well for CG, but can perform poorly for DG... **why?**

1. Discretize on hierarchy of meshes
2. Automatically build coarse operators: $\mathcal{C}(A_h) \to R_h^{2h} A_h I_{2h}^h$

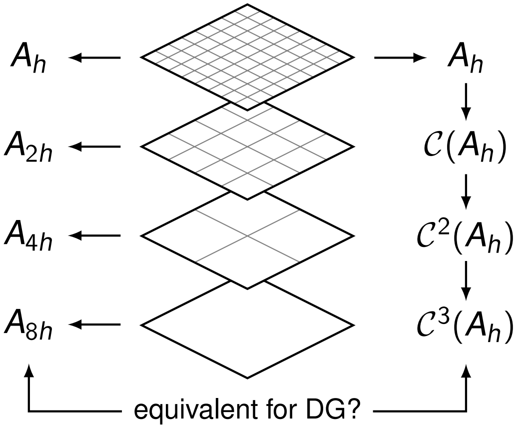


Avoids need to build coarse:
- ✓ quadrature rules
- ✓ lifting operators
- ✓ face-to-element maps
- ✓ mesh-related plumbing

Introduction

hp–multigrid works well for CG, but can perform poorly for DG... **why?**

1. Discretize on hierarchy of meshes
2. Automatically build coarse operators: $\mathcal{C}(A_h) \to R_h^{2h} A_h I_{2h}^h$

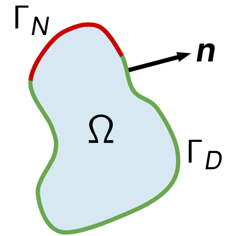


Avoids need to build coarse:
- ✓ quadrature rules
- ✓ lifting operators
- ✓ face-to-element maps
- ✓ mesh-related plumbing

DG for elliptic problems
General approach

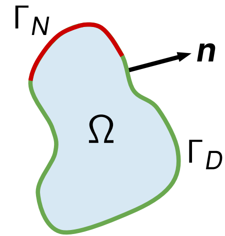
Consider the Poisson problem

$$\begin{aligned} -\nabla^2 u &= f \quad \text{in } \Omega \\ u &= g \quad \text{on } \Gamma_D \\ \nabla u \cdot \boldsymbol{n} &= h \quad \text{on } \Gamma_N \end{aligned}$$



DG for elliptic problems
General approach

Consider the Poisson problem



$$
\begin{aligned}
-\nabla^2 u &= f && \text{in } \Omega \\
u &= g && \text{on } \Gamma_D \\
\nabla u \cdot \boldsymbol{n} &= h && \text{on } \Gamma_N
\end{aligned}
$$

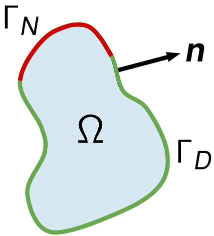
$$\downarrow$$

$$
\begin{aligned}
\boldsymbol{q} &= \nabla u && \text{in } \Omega \\
-\nabla \cdot \boldsymbol{q} &= f && \text{in } \Omega \\
u &= g && \text{on } \Gamma_D \\
\boldsymbol{q} \cdot \boldsymbol{n} &= h && \text{on } \Gamma_N
\end{aligned}
$$

DG for elliptic problems

General approach

Consider the Poisson problem



$$
\begin{aligned}
-\nabla^2 u &= f \quad \text{in } \Omega \\
u &= g \quad \text{on } \Gamma_D \\
\nabla u \cdot \boldsymbol{n} &= h \quad \text{on } \Gamma_N
\end{aligned}
$$

$$\Big\downarrow$$

$$
\begin{aligned}
\boldsymbol{q} &= \nabla u && \text{in } \Omega \\
-\nabla \cdot \boldsymbol{q} &= f && \text{in } \Omega \\
u &= g && \text{on } \Gamma_D \\
\boldsymbol{q} \cdot \boldsymbol{n} &= h && \text{on } \Gamma_N
\end{aligned}
$$

Elements are coupled via **numerical fluxes**.

The local discontinuous Galerkin method
A popular choice

- Numerical fluxes "upwind" $\boldsymbol{q}$ and u in opposite directions

The local discontinuous Galerkin method

A popular choice

- Numerical fluxes "upwind" $\boldsymbol{q}$ and u in opposite directions
- Leads to a system of the form

$$\text{“}\boldsymbol{q} = \nabla u\text{”} \qquad \int_\Omega \boldsymbol{q}_h \cdot \boldsymbol{w}_h - \int_\Omega G u_h \cdot \boldsymbol{w}_h = \int_{\Gamma_D} g\, \boldsymbol{w}_h^- \cdot \boldsymbol{n}$$

$$\text{“}-\nabla \cdot \boldsymbol{q} = f\text{”} \qquad \int_\Omega \boldsymbol{q}_h \cdot G v_h + \text{penalty} = \int_\Omega f\, v_h + \int_{\Gamma_N} h\, v_h^-$$

G is the **discrete gradient operator**:

$$G = \text{broken gradient} + \text{lifting operator}$$

where the lifting operator accounts for jumps between elements.

The local discontinuous Galerkin method

A popular choice

- Numerical fluxes "upwind" $\boldsymbol{q}$ and u in opposite directions
- Leads to a system of the form

$$
\begin{aligned}
m(\boldsymbol{q}_h, \boldsymbol{w}_h) - \mathrm{grad}(u_h, \boldsymbol{w}_h) &= j(\boldsymbol{w}_h) \\
-\mathrm{div}(\boldsymbol{q}_h, v_h) + \tau(u_h, v_h) &= k(v_h)
\end{aligned}
$$

The local discontinuous Galerkin method
A popular choice

- Numerical fluxes "upwind" $\boldsymbol{q}$ and u in opposite directions
- Leads to a system of the form

mass

$$\begin{aligned} m(\boldsymbol{q}_h, \boldsymbol{w}_h) - \operatorname{grad}(u_h, \boldsymbol{w}_h) &= j(\boldsymbol{w}_h) \\ -\operatorname{div}(\boldsymbol{q}_h, v_h) + \tau(u_h, v_h) &= k(v_h) \end{aligned}$$

The local discontinuous Galerkin method
A popular choice

- Numerical fluxes "upwind" $\boldsymbol{q}$ and u in opposite directions
- Leads to a system of the form

$$\underbrace{m(\boldsymbol{q}_h, \boldsymbol{w}_h)}_{\text{mass}} - \underbrace{\mathrm{grad}(u_h, \boldsymbol{w}_h)}_{\text{gradient}} = j(\boldsymbol{w}_h)$$

$$-\,\mathrm{div}(\boldsymbol{q}_h, v_h) + \tau(u_h, v_h) = k(v_h)$$

The local discontinuous Galerkin method

A popular choice

- Numerical fluxes “upwind” $\boldsymbol{q}$ and u in opposite directions
- Leads to a system of the form

$$
\begin{aligned}
\overset{\text{mass}}{m(\boldsymbol{q}_h, \boldsymbol{w}_h)} - \overset{\text{gradient}}{\operatorname{grad}(u_h, \boldsymbol{w}_h)} &= j(\boldsymbol{w}_h) \\
-\underbrace{\operatorname{div}(\boldsymbol{q}_h, v_h)}_{\text{divergence}} + \tau(u_h, v_h) &= k(v_h)
\end{aligned}
$$

The local discontinuous Galerkin method
A popular choice

- Numerical fluxes "upwind" $\boldsymbol{q}$ and u in opposite directions
- Leads to a system of the form

$$
\begin{aligned}
\overset{\text{mass}}{m(\boldsymbol{q}_h, \boldsymbol{w}_h)} - \overset{\text{gradient}}{\operatorname{grad}(u_h, \boldsymbol{w}_h)} &= j(\boldsymbol{w}_h) \\
-\underset{\text{divergence}}{\operatorname{div}(\boldsymbol{q}_h, v_h)} + \underset{\text{penalty}}{\tau(u_h, v_h)} &= k(v_h)
\end{aligned}
$$

The local discontinuous Galerkin method

A popular choice

- Numerical fluxes "upwind" $\boldsymbol{q}$ and u in opposite directions
- Leads to a system of the form

mass gradient

$$
\begin{aligned}
m(\boldsymbol{q}_h, \boldsymbol{w}_h) - \mathrm{grad}(u_h, \boldsymbol{w}_h) &= j(\boldsymbol{w}_h) \\
-\,\mathrm{div}(\boldsymbol{q}_h, v_h) + \tau(u_h, v_h) &= k(v_h)
\end{aligned}
$$

RHS + BCs

divergence penalty

Primal formulation

“Eliminate-then-discretize”

Eliminate $\boldsymbol{q}_h$ to get bilinear form of Laplacian: $a(u_h, v_h) = \ell(v_h)$

$$
\begin{aligned}
a(u_h, v_h) &= \int_\Omega G u_h \cdot G v_h + \text{penalty} \\
\ell(v_h) &= \int_\Omega f\, v_h - \int_{\Gamma_D} g\, G v_h^- \cdot \boldsymbol{n} + \int_{\Gamma_N} h\, v_h^-
\end{aligned}
$$

Primal formulation

"Eliminate-then-discretize"

Eliminate $\boldsymbol{q}_h$ to get bilinear form of Laplacian: $a(u_h, v_h) = \ell(v_h)$

$$
\begin{aligned}
a(u_h, v_h) &= \int_\Omega G u_h \cdot G v_h + \text{penalty} \\
\ell(v_h) &= \int_\Omega f\, v_h - \int_{\Gamma_D} g\, G v_h^- \cdot \boldsymbol{n} + \int_{\Gamma_N} h\, v_h^-
\end{aligned}
$$

Discretization leads to symmetric positive (semi)definite system

$$A_h u_h = \ell_h.$$

$A_h = -M_h \Delta_h$ and Δ_h is **discrete Laplacian operator**,

$$\Delta_h = D_h G_h + T_h.$$

$$
\begin{pmatrix}
M_h = \text{mass matrix} & G_h = \text{gradient matrix} \\
T_h = \text{penalty matrix} & D_h = \text{divergence matrix} = M_h^{-1} G_h^\top M_h
\end{pmatrix}
$$

Flux formulation

"Discretize-then-eliminate"

$$
\begin{aligned}
m(\boldsymbol{q}_h, \boldsymbol{w}_h) - \operatorname{grad}(u_h, \boldsymbol{w}_h) &= j(\boldsymbol{w}_h) \\
-\operatorname{div}(\boldsymbol{q}_h, v_h) + \tau(u_h, v_h) &= k(v_h)
\end{aligned}
$$

Flux formulation
"Discretize-then-eliminate"

$$\begin{bmatrix} M_h & -M_h G_h \\ -M_h D_h & M_h T_h \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_h \\ u_h \end{bmatrix} = \begin{bmatrix} j_h \\ k_h \end{bmatrix}$$

$$\begin{aligned} M_h &= \text{mass matrix} \\ G_h &= \text{discrete gradient matrix} \\ D_h &= \text{discrete divergence matrix} = -M_h^{-1} G_h^\top M_h \\ T_h &= \text{discrete penalty matrix} \end{aligned}$$

Flux formulation

"Discretize-then-eliminate"

$$\begin{bmatrix} M_h & -M_h G_h \\ -M_h D_h & M_h T_h \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_h \\ u_h \end{bmatrix} = \begin{bmatrix} j_h \\ k_h \end{bmatrix}$$

$$\begin{aligned} M_h &= \text{mass matrix} \\ G_h &= \text{discrete gradient matrix} \\ D_h &= \text{discrete divergence matrix} = -M_h^{-1} G_h^\top M_h \\ T_h &= \text{discrete penalty matrix} \end{aligned}$$

M_h is block diagonal, so can easily take Schur complement:

$$A_h u_h = \ell_h$$

where $A_h = -M_h \Delta_h$ and $\Delta_h = D_h G_h + T_h$.

Multigrid
Coarse operators

Primal formulation
"Eliminate-then-discretize"

$$-D_h G_h + T_h$$

$$\downarrow$$

$$\mathcal{C}(-D_h G_h + T_h)$$

Flux formulation
"Discretize-then-eliminate"

$$\begin{bmatrix} I & -G_h \\ -D_h & T_h \end{bmatrix}$$

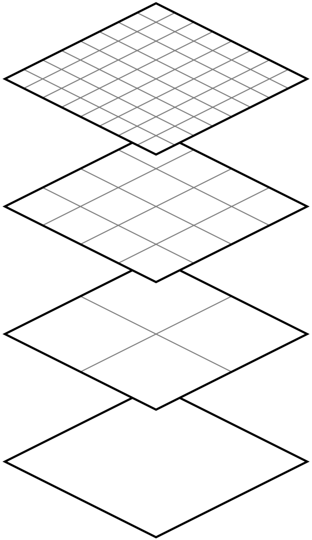
$$\downarrow$$

$$\begin{bmatrix} I & -\mathcal{C}(G_h) \\ -\mathcal{C}(D_h) & \mathcal{C}(T_h) \end{bmatrix}$$

$$\downarrow$$

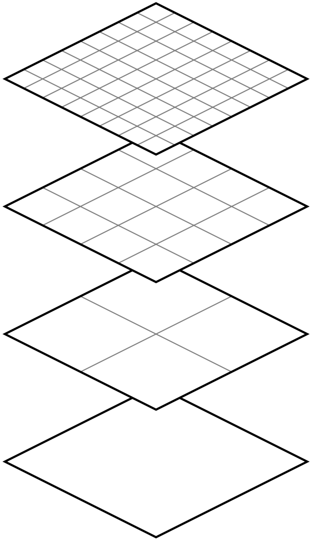
$$-\mathcal{C}(D_h)\mathcal{C}(G_h) + \mathcal{C}(T_h)$$

Multigrid
Coarse operators



Multigrid
Coarse operators

“geometric”



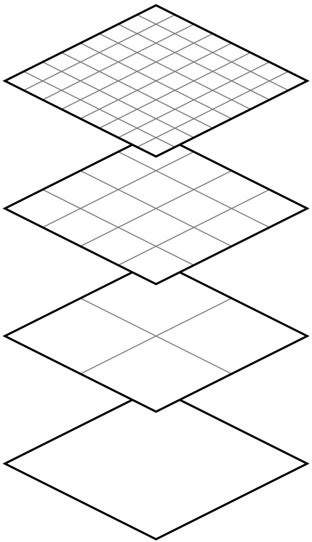
$\longrightarrow$ A_h

$\longrightarrow$ A_{2h}

$\longrightarrow$ A_{4h}

$\longrightarrow$ A_{8h}

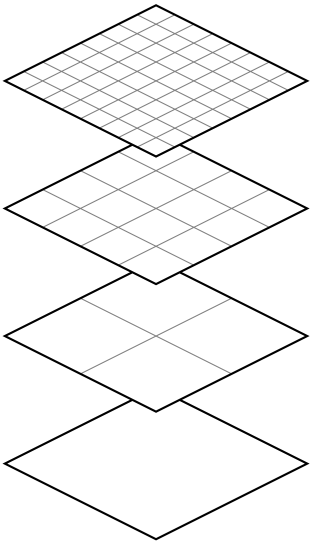
Multigrid
Coarse operators



"geometric" "primal"

$$
\begin{tikzcd}
A_h \arrow[r] & A_h \arrow[d] \\
A_{2h} & \mathcal{C}(A_h) \arrow[d] \\
A_{4h} & \mathcal{C}^2(A_h) \arrow[d] \\
A_{8h} & \mathcal{C}^3(A_h)
\end{tikzcd}
$$

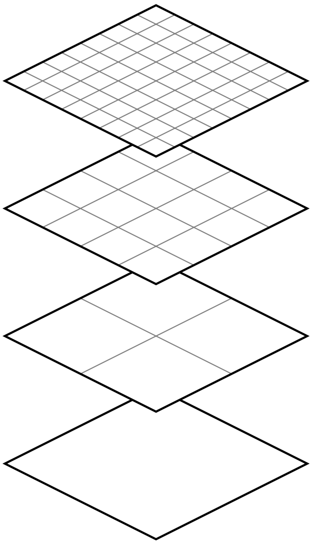
Multigrid
Coarse operators



“geometric” “primal” “flux”

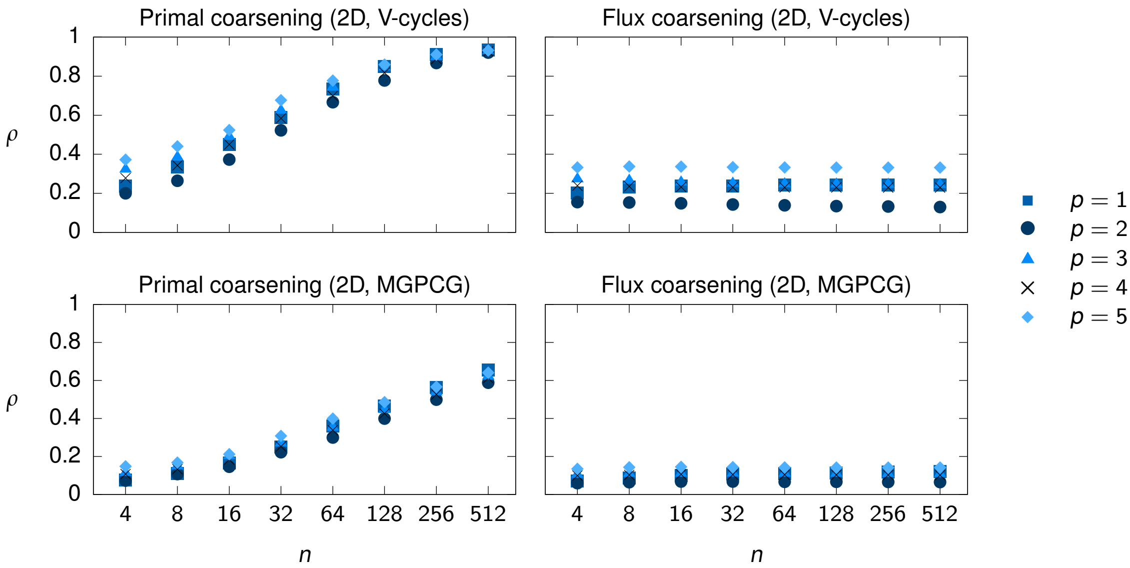
$$\begin{tikzcd}
A_h \arrow[r] & A_h \arrow[r] \arrow[d] & -D_h G_h + T_h \arrow[d] \\
A_{2h} & \mathcal{C}(A_h) \arrow[d] & -\mathcal{C}(D_h)\mathcal{C}(G_h) + \mathcal{C}(T_h) \arrow[d] \\
A_{4h} & \mathcal{C}^2(A_h) \arrow[d] & -\mathcal{C}^2(D_h)\mathcal{C}^2(G_h) + \mathcal{C}^2(T_h) \arrow[d] \\
A_{8h} & \mathcal{C}^3(A_h) & -\mathcal{C}^3(D_h)\mathcal{C}^3(G_h) + \mathcal{C}^3(T_h)
\end{tikzcd}$$

Multigrid
Coarse operators



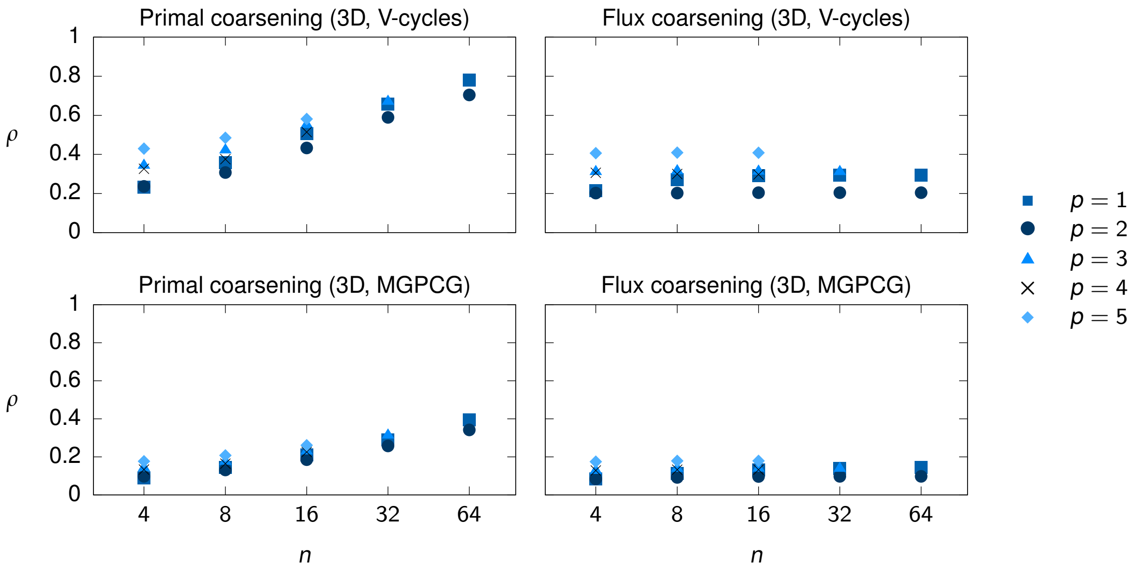
$$
\begin{tikzcd}
\text{“geometric”} & \text{“primal”} & \text{“flux”} \\
A_h \arrow[r] & A_h \arrow[r] \arrow[d] & -D_h G_h + T_h \arrow[d] \\
A_{2h} & \mathcal{C}(A_h) \arrow[d] & -\mathcal{C}(D_h)\mathcal{C}(G_h) + \mathcal{C}(T_h) \arrow[d] \\
A_{4h} & \mathcal{C}^2(A_h) \arrow[d] & -\mathcal{C}^2(D_h)\mathcal{C}^2(G_h) + \mathcal{C}^2(T_h) \arrow[d] \\
A_{8h} & \mathcal{C}^3(A_h) & -\mathcal{C}^3(D_h)\mathcal{C}^3(G_h) + \mathcal{C}^3(T_h) \\
 & \text{equivalent!} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

Example
Uniform Cartesian grids, 2D



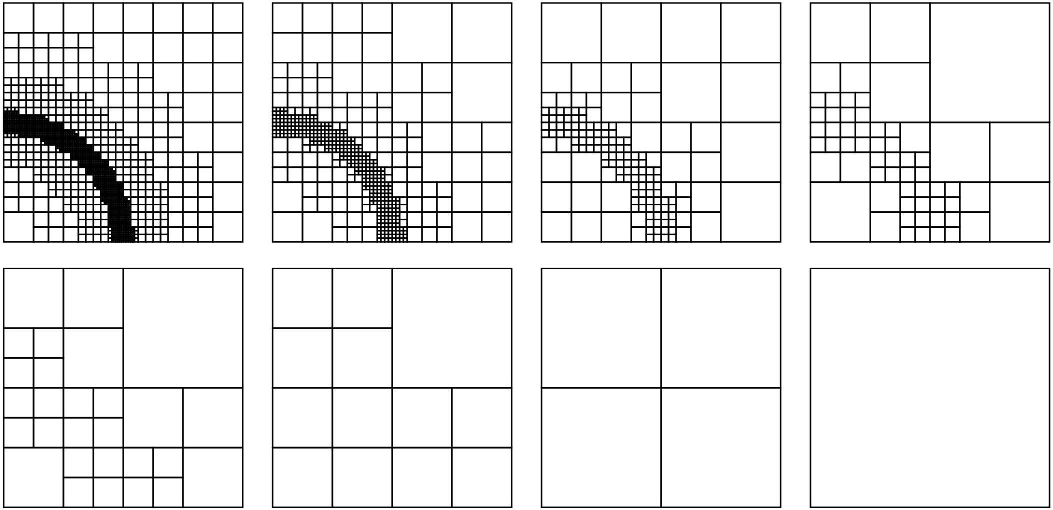
Example

Uniform Cartesian grids, 3D

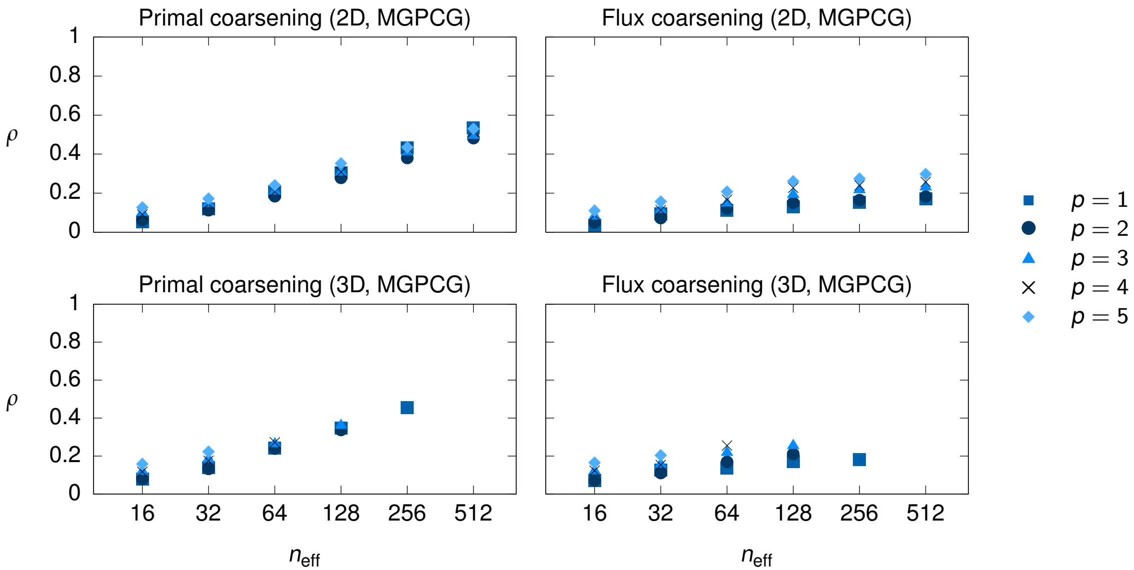


Example

Adaptive grids

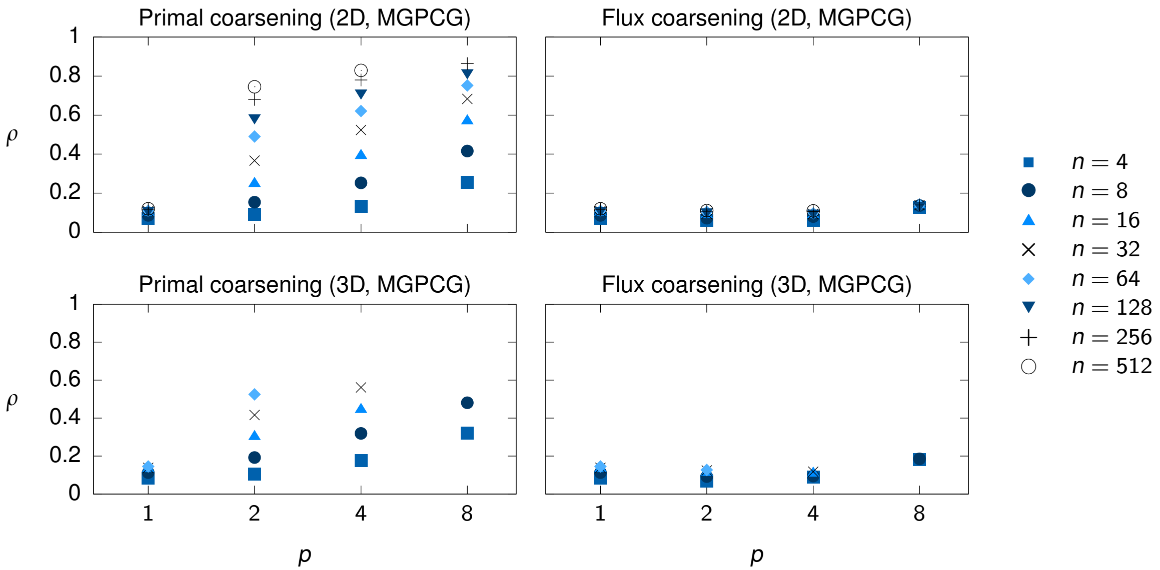


Example
Adaptive grids

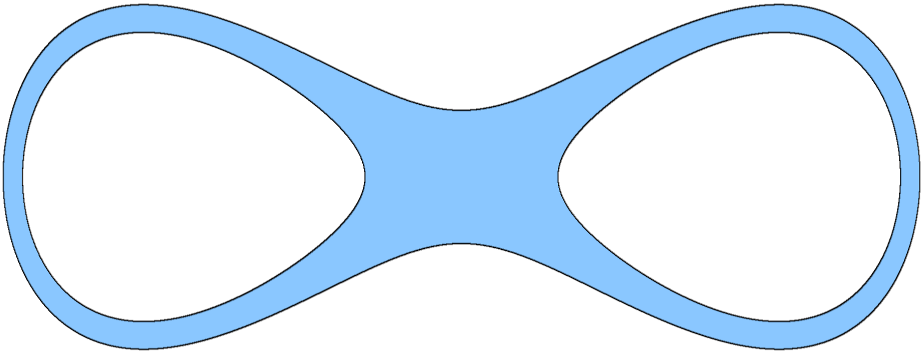


Example

p-multigrid: $p \to p/2 \to \cdots \to 1$

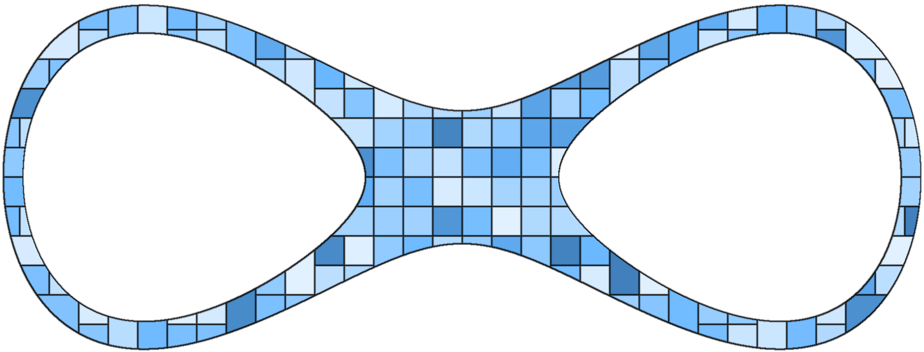


$$-\nabla^2 u = 0, \quad u|_\Gamma = 0$$



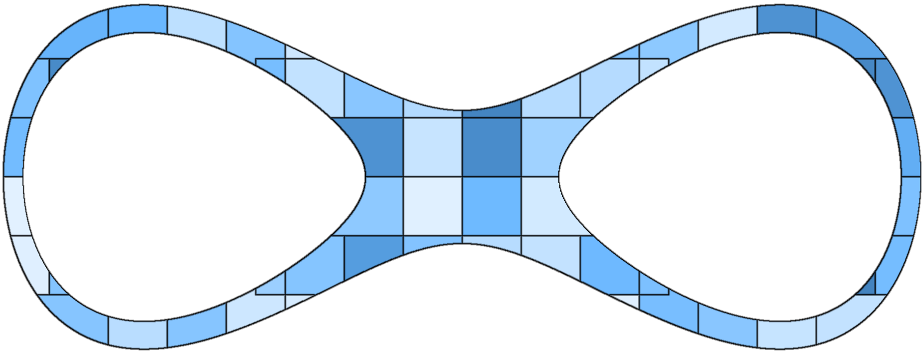
Example

Implicitly defined mesh — single phase



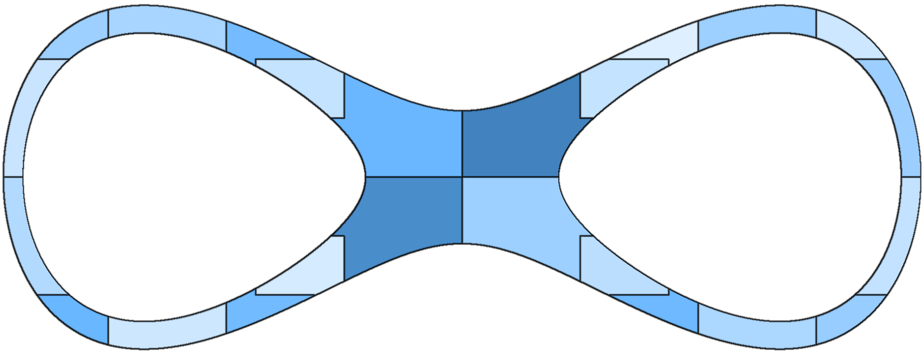
Example

Implicitly defined mesh — single phase



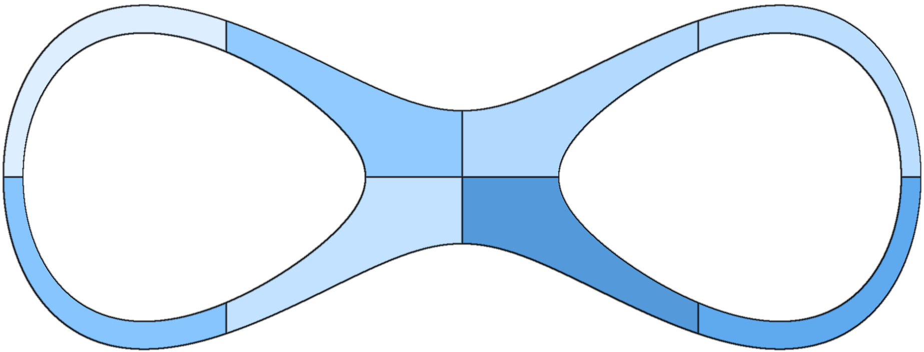
Example

Implicitly defined mesh — single phase



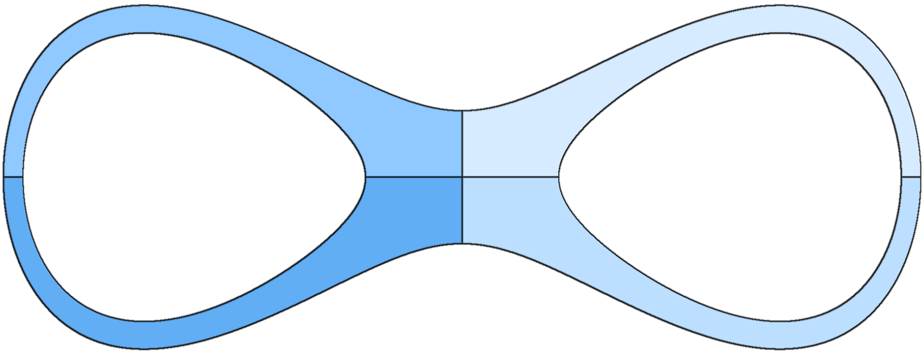
Example

Implicitly defined mesh — single phase



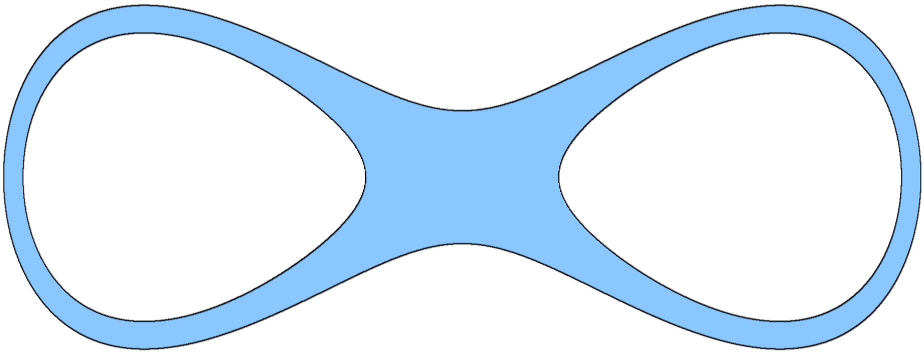
Example

Implicitly defined mesh — single phase



Example

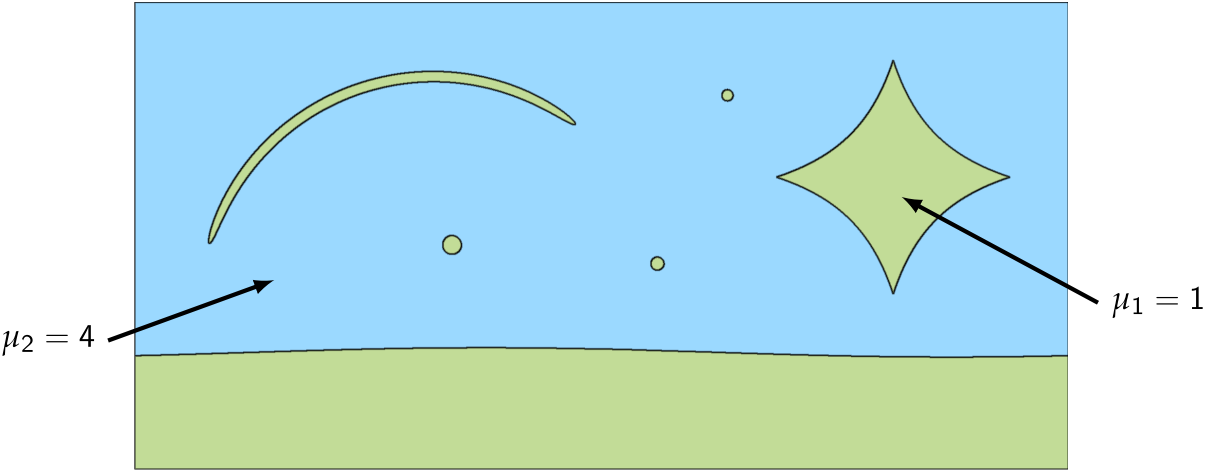
Implicitly defined mesh — single phase



Example

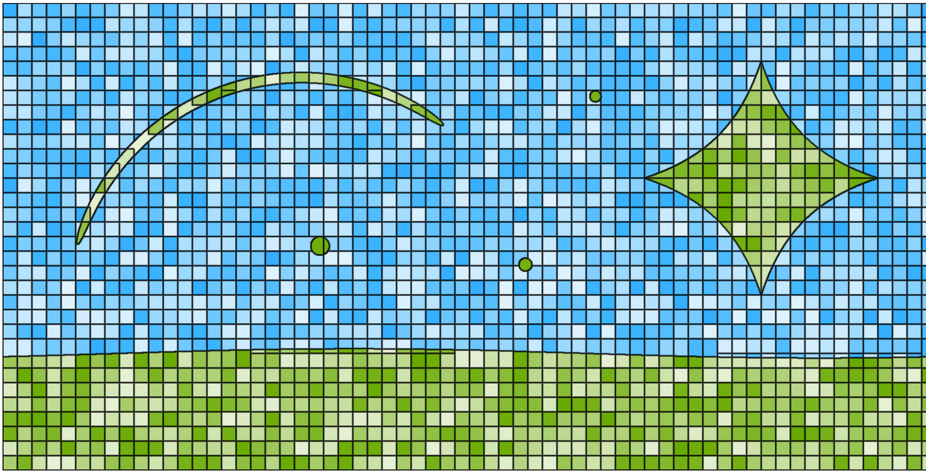
Implicitly defined mesh — multiphase

$$-\nabla \cdot (\mu_i \nabla u) = 0, \quad [u] = 0, \quad u|_\Gamma = 0$$



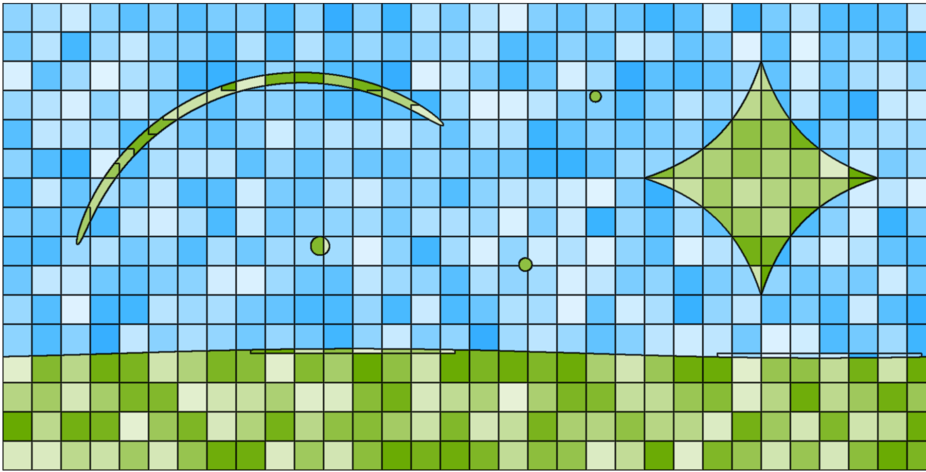
Example

Implicitly defined mesh — multiphase



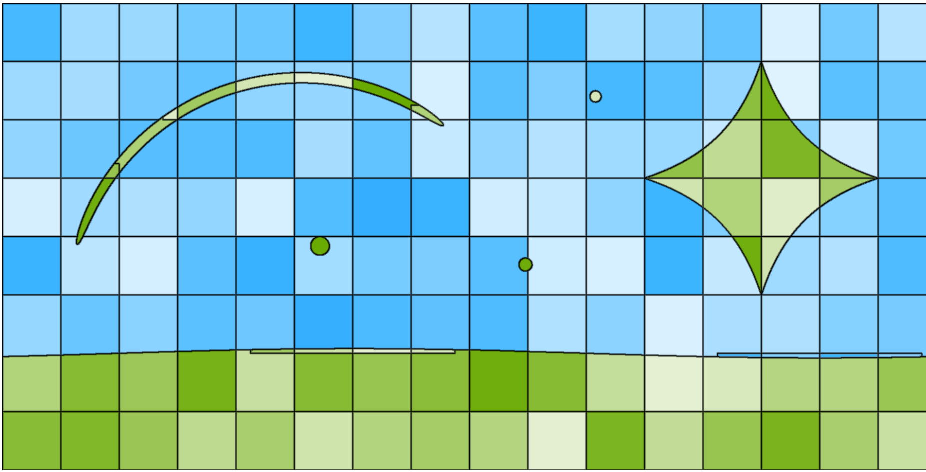
Example

Implicitly defined mesh — multiphase



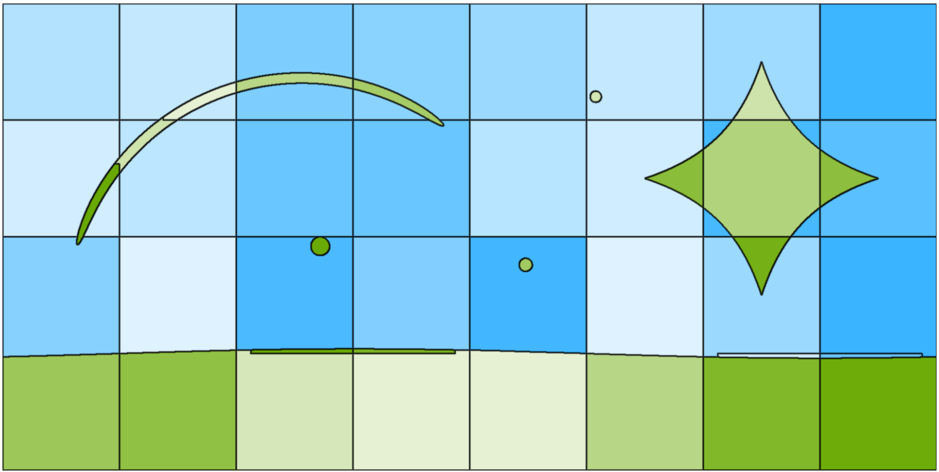
Example

Implicitly defined mesh — multiphase



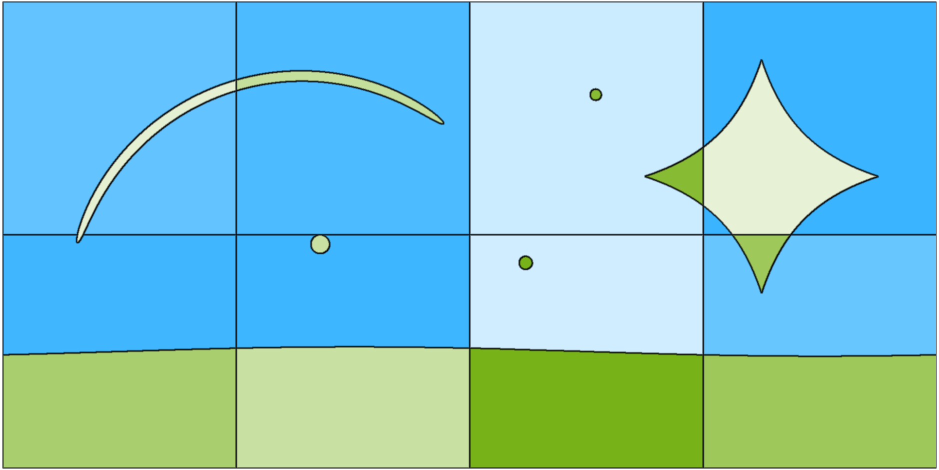
Example

Implicitly defined mesh — multiphase



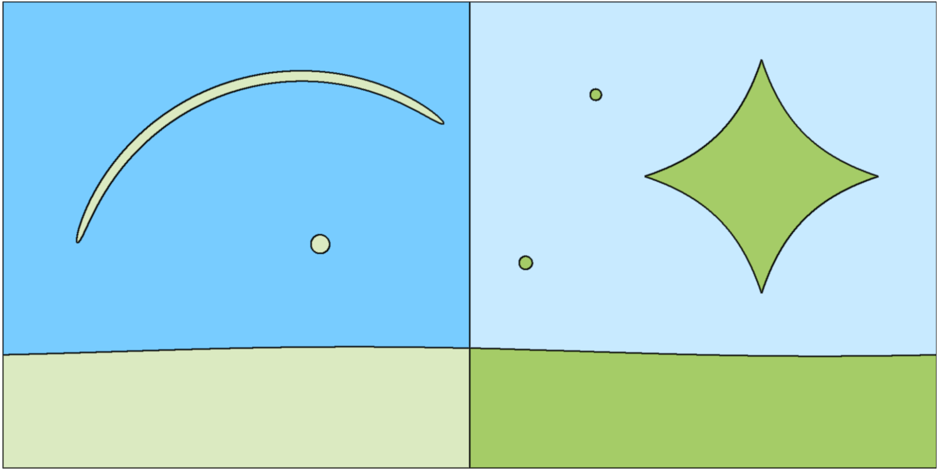
Example

Implicitly defined mesh — multiphase



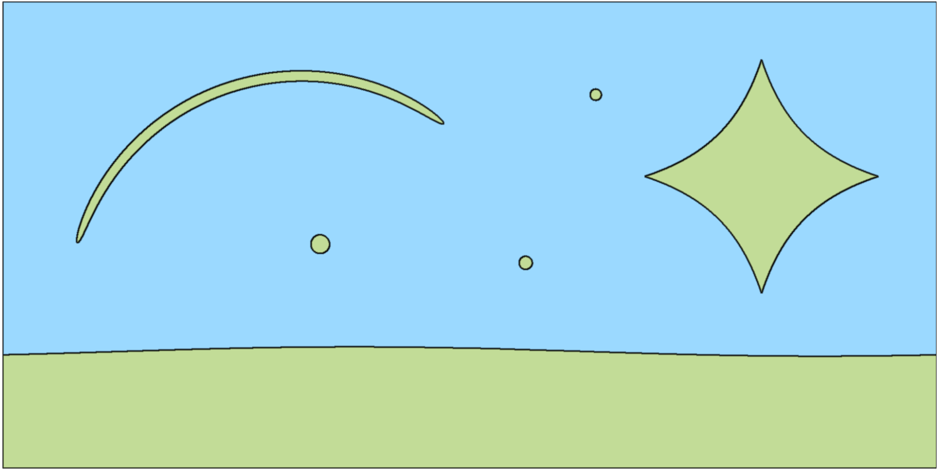
Example

Implicitly defined mesh — multiphase

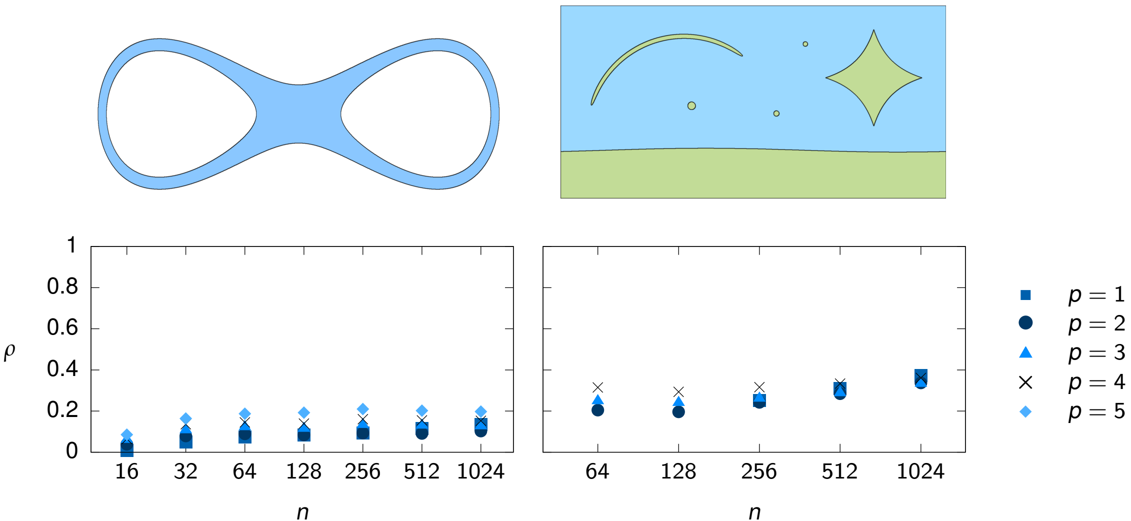


Example

Implicitly defined mesh — multiphase



Example
Implicitly defined mesh — multiphase



Thanks for listening!

Preprint to be posted: `danfortunato.com/multigrid-ldg`