Fast and accurate element methods

Dan Fortunato

Harvard University

Qualifying Exam

May 22, 2017

Advisors: Chris Rycroft, Alex Townsend

Motivation

The numerical solution of PDEs arises everywhere in simulation, but...

- solving PDEs **accurately** can be **slow**.
- solving PDEs **fast** can be **inaccurate**.

Methods that are both **fast** and **accurate** are essential for large-scale simulation.

Fast → optimal complexity

- Time to solution is proportional to number of unknowns (up to log factors).

Accurate → spectrally accurate

- Discretization error is limited only by smoothness of input and output functions.

Overview

- An optimal complexity spectral element method
  - Poisson on a rectangle
  - Poisson on a quadrilateral
  - Towards a spectral element method
- Discontinuous Galerkin methods
  - Eulerian fluid-structure interaction
  - Elliptic problems

An optimal complexity spectral element method
Motivation

hp-adaptive spectral element methods exist in theory but not in practice.

If we have an optimal complexity spectral element method, then *h* and *p* can be chosen **based on physical considerations only**.

h-refinement is good for:

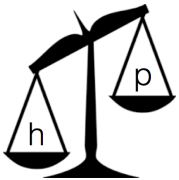
- corner singularities
- discontinuities/shocks

p-refinement is good for:

- smooth solutions
- advection-dominated fluid flow
- high-frequency acoustic scattering

An optimal complexity spectral element method
Motivation

hp-adaptive spectral element methods exist in theory but not in practice.



If we have an optimal complexity spectral element method, then *h* and *p* can be chosen **based on physical considerations only**.

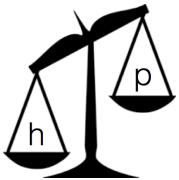
h-refinement is good for:

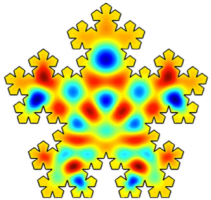
- corner singularities
- discontinuities/shocks

p-refinement is good for:

- smooth solutions
- advection-dominated fluid flow
- high-frequency acoustic scattering

hp-adaptive spectral element methods exist in theory but not in practice.





If we have an optimal complexity spectral element method, then *h* and *p* can be chosen **based on physical considerations only**.

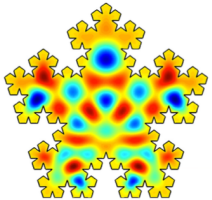
h-refinement is good for:

- corner singularities
- discontinuities/shocks

p-refinement is good for:

- smooth solutions
- advection-dominated fluid flow
- high-frequency acoustic scattering

hp-adaptive spectral element methods exist in theory but not in practice.



If we have an optimal complexity spectral element method, then *h* and *p* can be chosen **based on physical considerations only**.

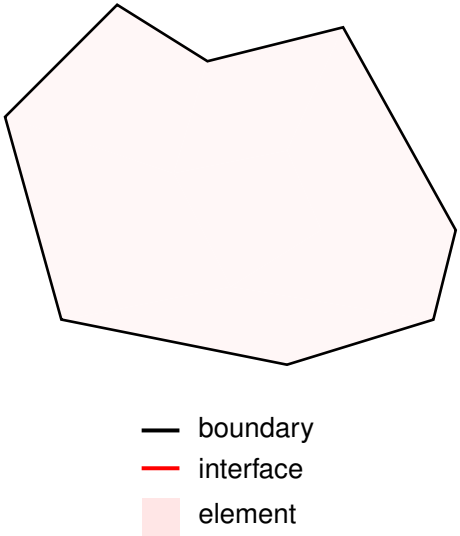
h-refinement is good for:

- corner singularities
- discontinuities/shocks

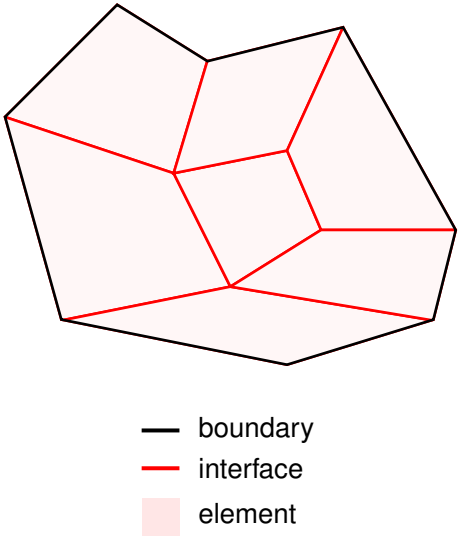
p-refinement is good for:

- smooth solutions
- advection-dominated fluid flow
- high-frequency acoustic scattering

- An element solver (local)
- An interface solver (global)

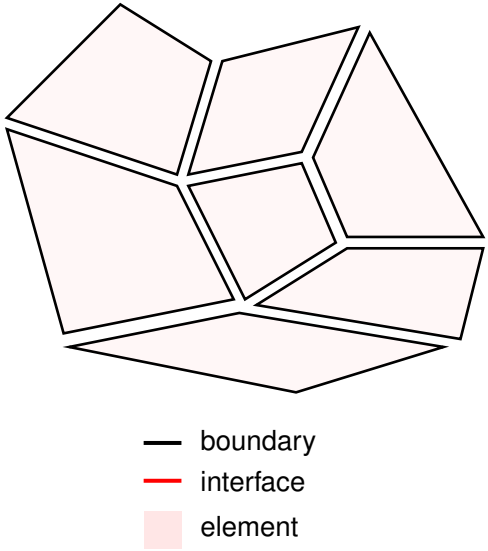


- An element solver (local)
- An interface solver (global)



- An element solver (local)
- An interface solver (global)

If we can solve for the interfaces, the elements **decouple**!



Poisson on a rectangle

Consider Poisson's equation on $[-1,1]^2$ with homogeneous Dirichlet conditions,

$$u_{xx} + u_{yy} = f, \quad (x,y) \in [-1,1]^2, \qquad u(\pm 1, \cdot) = u(\cdot, \pm 1) = 0.$$

The classic fast Poisson solver using finite differences:

$$\underbrace{KX + XK = F}_{\text{solve with FFT, } \mathcal{O}(p^2 \log p)}, \qquad K = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

[Buzbee et al, 1970]

- Based on **structured eigenvectors**
- Complexity increases with order of accuracy

Can we make a spectrally-accurate
Poisson solver with $\mathcal{O}(p^2 \log p)$ complexity?

Consider Poisson's equation on $[-1,1]^2$ with homogeneous Dirichlet conditions,

$$u_{xx} + u_{yy} = f, \quad (x,y) \in [-1,1]^2, \qquad u(\pm 1, \cdot) = u(\cdot, \pm 1) = 0.$$

The classic fast Poisson solver using finite differences:

$$\underbrace{KX + XK = F}_{\text{solve with FFT, } \mathcal{O}(p^2 \log p)}, \qquad K = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

[Buzbee et al, 1970]

- Based on **structured eigenvectors**
- Complexity increases with order of accuracy

Can we make a spectrally-accurate Poisson solver with $\mathcal{O}(p^2 \log p)$ complexity?

Consider Poisson's equation on $[-1,1]^2$ with homogeneous Dirichlet conditions,

$$u_{xx} + u_{yy} = f, \quad (x,y) \in [-1,1]^2, \qquad u(\pm 1, \cdot) = u(\cdot, \pm 1) = 0.$$

The classic fast Poisson solver using finite differences:

$$\underbrace{KX + XK = F}_{\text{solve with FFT, } \mathcal{O}(p^2 \log p)}, \qquad K = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

[Buzbee et al, 1970]

- Based on **structured eigenvectors**
- Complexity increases with order of accuracy

Can we make a spectrally-accurate
Poisson solver with $\mathcal{O}(p^2 \log p)$ complexity?

Consider Poisson's equation on $[-1,1]^2$ with homogeneous Dirichlet conditions,

$$u_{xx} + u_{yy} = f, \quad (x,y) \in [-1,1]^2, \qquad u(\pm 1, \cdot) = u(\cdot, \pm 1) = 0.$$

The classic fast Poisson solver using finite differences:

$$\underbrace{KX + XK = F}_{\text{solve with FFT, } \mathcal{O}(p^2 \log p)}, \qquad K = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

[Buzbee et al, 1970]

breaks down for spectral

- Based on **structured eigenvectors**
- Complexity increases with order of accuracy

Can we make a spectrally-accurate
Poisson solver with $\mathcal{O}(p^2 \log p)$ complexity?

Consider Poisson's equation on $[-1,1]^2$ with homogeneous Dirichlet conditions,

$$u_{xx} + u_{yy} = f, \quad (x,y) \in [-1,1]^2, \qquad u(\pm 1, \cdot) = u(\cdot, \pm 1) = 0.$$

The classic fast Poisson solver using finite differences:

$$\underbrace{KX + XK = F}_{\text{solve with FFT, } \mathcal{O}(p^2 \log p)}, \qquad K = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

[Buzbee et al, 1970]

breaks down for spectral

- Based on **structured eigenvectors**
- Complexity increases with order of accuracy

Can we make a spectrally-accurate Poisson solver with $\mathcal{O}(p^2 \log p)$ complexity?

A clever choice of basis
The ultraspherical polynomials

Dirichlet on $[-1, 1]$ $\longleftrightarrow$ Pick a basis that vanishes at ± 1

The classical orthogonal polynomials, f_k, satisfy

$$A(x)f_k''(x) + B(x)f_k'(x) = q_k f_k(x), \qquad x \in [-1, 1].$$

The second derivative of $(1 - x^2)C_k^{(\lambda)}(x)$ is given by

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(\lambda)}(x)\right] = (1-x^2)C_k^{(\lambda)''}(x) - 4xC_k^{(\lambda)'}(x) - 2C_k^{(\lambda)}(x).$$

Idea: Choose $\lambda = \frac{3}{2}$

A clever choice of basis
The ultraspherical polynomials

Dirichlet on $[-1, 1]$ $\longleftrightarrow$ Pick a basis that vanishes at ± 1

The classical orthogonal polynomials, f_k, satisfy

$$A(x)f_k''(x) + B(x)f_k'(x) = q_k f_k(x), \qquad x \in [-1, 1].$$

The second derivative of $(1 - x^2)C_k^{(\lambda)}(x)$ is given by

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(\lambda)}(x)\right] = (1-x^2)C_k^{(\lambda)''}(x) - 4xC_k^{(\lambda)'}(x) - 2C_k^{(\lambda)}(x).$$

Idea: Choose $\lambda = \frac{3}{2}$

A clever choice of basis
The ultraspherical polynomials

Dirichlet on $[-1,1]$ $\longleftrightarrow$ Pick a basis that vanishes at ± 1

The ultraspherical polynomials of parameter λ, $C_k^{(\lambda)}$, satisfy [NIST DLMF, 18.8.1]

$$(1-x^2)C_k^{(\lambda)''}(x) - (2\lambda+1)xC_k^{(\lambda)'}(x) = -k(k+2\lambda)C_k^{(\lambda)}(x), \qquad x \in [-1,1].$$

The second derivative of $(1-x^2)C_k^{(\lambda)}(x)$ is given by

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(\lambda)}(x)\right] = (1-x^2)C_k^{(\lambda)''}(x) - 4xC_k^{(\lambda)'}(x) - 2C_k^{(\lambda)}(x).$$

Idea: Choose $\lambda = \frac{3}{2}$

A clever choice of basis
The ultraspherical polynomials

Dirichlet on $[-1,1]$ $\longleftrightarrow$ Pick a basis that vanishes at ± 1

The ultraspherical polynomials of parameter λ, $C_k^{(\lambda)}$, satisfy [NIST DLMF, 18.8.1]

$$(1-x^2)C_k^{(\lambda)\prime\prime}(x) - (2\lambda+1)xC_k^{(\lambda)\prime}(x) = -k(k+2\lambda)C_k^{(\lambda)}(x), \qquad x \in [-1,1].$$

The second derivative of $(1-x^2)C_k^{(\lambda)}(x)$ is given by

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(\lambda)}(x)\right] = (1-x^2)C_k^{(\lambda)\prime\prime}(x) - 4xC_k^{(\lambda)\prime}(x) - 2C_k^{(\lambda)}(x).$$

Idea: Choose $\lambda = \frac{3}{2}$

A clever choice of basis
The ultraspherical polynomials

Dirichlet on $[-1,1]$ $\longleftrightarrow$ Pick a basis that vanishes at ± 1

The ultraspherical polynomials of parameter λ, $C_k^{(\lambda)}$, satisfy [NIST DLMF, 18.8.1]

$$(1-x^2)C_k^{(\lambda)''}(x) - (2\lambda+1)xC_k^{(\lambda)'}(x) = -k(k+2\lambda)C_k^{(\lambda)}(x), \qquad x \in [-1,1].$$

The second derivative of $(1-x^2)C_k^{(\lambda)}(x)$ is given by

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(\lambda)}(x)\right] = (1-x^2)C_k^{(\lambda)''}(x) - 4xC_k^{(\lambda)'}(x) - 2C_k^{(\lambda)}(x).$$

Idea: Choose $\lambda = \frac{3}{2}$

A clever choice of basis
The ultraspherical polynomials

Dirichlet on $[-1,1]$ $\longleftrightarrow$ Pick a basis that vanishes at ± 1

The ultraspherical polynomials of parameter λ, $C_k^{(\lambda)}$, satisfy [NIST DLMF, 18.8.1]

$$(1-x^2)C_k^{(\lambda)''}(x) - (2\lambda+1)xC_k^{(\lambda)'}(x) = -k(k+2\lambda)C_k^{(\lambda)}(x), \qquad x \in [-1,1].$$

The second derivative of $(1-x^2)C_k^{(\lambda)}(x)$ is given by

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(\lambda)}(x)\right] = (1-x^2)C_k^{(\lambda)''}(x) - 4xC_k^{(\lambda)'}(x) - 2C_k^{(\lambda)}(x).$$

Idea: Choose $\lambda = \frac{3}{2}$

A clever choice of basis
The ultraspherical polynomials

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(3/2)}(x)\right] = -(k(k+3)+2)C_k^{(3/2)}(x).$$

$C_k^{(3/2)}(x)$ is an eigenfunction of the differential operator $u \mapsto \frac{\partial^2}{\partial x^2}(1-x^2)u$

$$\nabla^2\left[(1-x^2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y)\right] = -(j(j+3)+2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y) - (k(k+3)+2)(1-x^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y).$$

Therefore, represent the solution in the basis

$$u(x,y) \approx \sum_{j=0}^{m-1}\sum_{k=0}^{n-1} X_{jk}(1-x^2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y), \qquad (x,y) \in [-1,1]^2.$$

A clever choice of basis
The ultraspherical polynomials

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(3/2)}(x)\right] = -(k(k+3)+2)C_k^{(3/2)}(x).$$

$C_k^{(3/2)}(x)$ is an eigenfunction of the differential operator $u \mapsto \frac{\partial^2}{\partial x^2}(1-x^2)u$

$$\nabla^2\left[(1-x^2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y)\right] = -(j(j+3)+2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y) - (k(k+3)+2)(1-x^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y).$$

Therefore, represent the solution in the basis

$$u(x,y) \approx \sum_{j=0}^{m-1}\sum_{k=0}^{n-1} X_{jk}(1-x^2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y), \qquad (x,y) \in [-1,1]^2.$$

A clever choice of basis
The ultraspherical polynomials

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(3/2)}(x)\right] = -(k(k+3)+2)C_k^{(3/2)}(x).$$

$C_k^{(3/2)}(x)$ is an eigenfunction of the differential operator $u \mapsto \frac{\partial^2}{\partial x^2}(1-x^2)u$

$$\nabla^2\left[(1-x^2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y)\right] = -(j(j+3)+2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y) - (k(k+3)+2)(1-x^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y).$$

Therefore, represent the solution in the basis

$$u(x,y) \approx \sum_{j=0}^{m-1}\sum_{k=0}^{n-1} X_{jk}(1-x^2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y), \qquad (x,y)\in[-1,1]^2.$$

A clever choice of basis
The ultraspherical polynomials

$$\frac{\partial^2}{\partial x^2}\left[(1-x^2)C_k^{(3/2)}(x)\right] = -(k(k+3)+2)C_k^{(3/2)}(x).$$

$C_k^{(3/2)}(x)$ is an eigenfunction of the differential operator $u \mapsto \frac{\partial^2}{\partial x^2}(1-x^2)u$

$$\nabla^2\left[(1-x^2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y)\right] = -(j(j+3)+2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y) - (k(k+3)+2)(1-x^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y).$$

Therefore, represent the solution in the basis

$$u(x,y) \approx \sum_{j=0}^{m-1}\sum_{k=0}^{n-1} X_{jk}(1-x^2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y), \qquad (x,y) \in [-1,1]^2.$$

A clever choice of basis
Can we “diagonalize” Poisson?

$$\nabla^2 u = f$$

We know the action of ∇^2 on this basis:

$$\nabla^2 \left[(1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = -(j(j+3)+2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) - (k(k+3)+2)(1-x^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y).$$

A clever choice of basis
Can we “diagonalize” Poisson?

$$\nabla^2 \left[\sum_{j,k} X_{jk} (1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = \sum_{j,k} F_{jk} C_j^{(3/2)}(x) C_k^{(3/2)}(y)$$

We know the action of ∇^2 on this basis:

$$\nabla^2 \left[(1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = -(j(j+3)+2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) - (k(k+3)+2)(1-x^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y).$$

A clever choice of basis
Can we “diagonalize” Poisson?

$$\nabla^2 \left[\sum_{j,k} X_{jk} (1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = \sum_{j,k} F_{jk} C_j^{(3/2)}(x) C_k^{(3/2)}(y)$$

We know the action of ∇^2 on this basis:

$$\begin{aligned} \nabla^2 \left[(1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = & -(j(j+3)+2)\,(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \\ & -(k(k+3)+2)\,(1-x^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y). \end{aligned}$$

A clever choice of basis
Can we “diagonalize” Poisson?

$$MXD^T + DXM^T = F$$

We know the action of ∇^2 on this basis:

$$\nabla^2 \left[(1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = -(j(j+3)+2)\,(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \\ -(k(k+3)+2)\,(1-x^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y).$$

A clever choice of basis
Can we “diagonalize” Poisson?

$$MXD^T + DXM^T = F$$

We know the action of ∇^2 on this basis:

$$\nabla^2 \left[(1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = \overbrace{-(j(j+3)+2)}^{\text{scale}} (1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) - (k(k+3)+2)(1-x^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y).$$

A clever choice of basis
Can we “diagonalize” Poisson?

$$MXD^T + DXM^T = F$$

We know the action of ∇^2 on this basis:

$$\nabla^2 \left[(1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = -(j(j+3)+2) \overbrace{(1-y^2)}^{\text{multiply}} C_j^{(3/2)}(x) C_k^{(3/2)}(y) - (k(k+3)+2)(1-x^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y).$$

A clever choice of basis
Can we “diagonalize” Poisson?

$$MXD^T + DXM^T = F$$

We know the action of ∇^2 on this basis:

$$\nabla^2 \left[(1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = -(j(j+3)+2)\,(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y)$$
$$\underbrace{-(k(k+3)+2)}_{\text{scale}}\,(1-x^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y).$$

A clever choice of basis
Can we “diagonalize” Poisson?

$$MXD^T + DXM^T = F$$

We know the action of ∇^2 on this basis:

$$\nabla^2 \left[(1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = -(j(j+3)+2)\,(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) - (k(k+3)+2)\,\underbrace{(1-x^2)}_{\text{multiply}} C_j^{(3/2)}(x) C_k^{(3/2)}(y).$$

A clever choice of basis
Can we “diagonalize” Poisson?

diagonal

$$MXD^T + DXM^T = F$$

pentadiagonal

[NIST DLMF, 18.9.7-8]

We know the action of ∇^2 on this basis:

$$\nabla^2 \left[(1-x^2)(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) \right] = -(j(j+3)+2)\,(1-y^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y) - (k(k+3)+2)\,(1-x^2) C_j^{(3/2)}(x) C_k^{(3/2)}(y).$$

A clever choice of basis
Can we “diagonalize” Poisson?

$$TX + XT^T = D^{-1}FD^{-1}, \qquad T = D^{-1}M$$

pentadiagonal

We know the action of ∇^2 on this basis:

$$\nabla^2\left[(1-x^2)(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y)\right] = -(j(j+3)+2)\,(1-y^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y) \\ -(k(k+3)+2)\,(1-x^2)C_j^{(3/2)}(x)C_k^{(3/2)}(y).$$

The Alternating Direction Implicit (ADI) method
(for solving matrix equations) [Wachspress, 1987]

$$TX + XT^T = F$$

- Based on **structured eigenvalues**
- Optimal parameters known [Lu & Wachspress, 1991]

still works for spectral

set $X_0 = 0$

pick shift parameters p_j

for $j = 0, \ldots, J$

solve $X_{j+1/2}(T^T + p_j I) = F - (T - p_j I)X_j$

solve $(T + p_j I)X_{j+1} = F - X_{j+1/2}(T^T - p_j I)$

} Thomas algorithm $\mathcal{O}(p^2)$

The Alternating Direction Implicit (ADI) method
(for solving matrix equations) [Wachspress, 1987]

$$TX + XT^T = F$$

- Based on **structured eigenvalues**
- Optimal parameters known [Lu & Wachspress, 1991]

still works for spectral

set $X_0 = 0$

pick shift parameters p_j

for $j = 0, \ldots, J$

solve $X_{j+1/2}(T^T + p_j I) = F - (T - p_j I)X_j$

solve $(T + p_j I)X_{j+1} = F - X_{j+1/2}(T^T - p_j I)$

} Thomas algorithm $\mathcal{O}(p^2)$

The Alternating Direction Implicit (ADI) method
(for solving matrix equations) [Wachspress, 1987]

$$TX + XT^T = F$$

- Based on **structured eigenvalues**
- Optimal parameters known [Lu & Wachspress, 1991]

still works for spectral

set $X_0 = 0$

pick shift parameters p_j

for $j = 0, \ldots, J$

solve $X_{j+1/2}(T^T + p_j I) = F - (T - p_j I)X_j$

solve $(T + p_j I)X_{j+1} = F - X_{j+1/2}(T^T - p_j I)$

} Thomas algorithm $\mathcal{O}(p^2)$

The Alternating Direction Implicit (ADI) method
(for solving matrix equations) [Wachspress, 1987]

$$TX + XT^T = F$$

- Based on **structured eigenvalues**
- Optimal parameters known [Lu & Wachspress, 1991]

still works for spectral

set $X_0 = 0$

pick shift parameters p_j

for $j = 0, \ldots, J$

solve $X_{j+1/2}(T^T + p_j I) = F - (T - p_j I)X_j$

solve $(T + p_j I)X_{j+1} = F - X_{j+1/2}(T^T - p_j I)$

Thomas algorithm
$\mathcal{O}(p^2)$

The Alternating Direction Implicit (ADI) method
(for solving matrix equations) [Wachspress, 1987]

$$TX + XT^T = F$$

- Based on **structured eigenvalues**
- Optimal parameters known [Lu & Wachspress, 1991]

still works for spectral

set $X_0 = 0$

pick shift parameters p_j

for $j = 0, \ldots, J$

solve $X_{j+1/2}(T^T + p_j I) = F - (T - p_j I)X_j$

solve $(T + p_j I)X_{j+1} = F - X_{j+1/2}(T^T - p_j I)$

} Thomas algorithm $\mathcal{O}(p^2)$

The Alternating Direction Implicit (ADI) method
(for solving matrix equations) [Wachspress, 1987]

$$TX + XT^T = F$$

- Based on **structured eigenvalues**
- Optimal parameters known [Lu & Wachspress, 1991]

still works for spectral

set $X_0 = 0$

pick shift parameters p_j

for $j = 0, \ldots, J$?

solve $X_{j+1/2}(T^T + p_j I) = F - (T - p_j I)X_j$

solve $(T + p_j I)X_{j+1} = F - X_{j+1/2}(T^T - p_j I)$

} Thomas algorithm $\mathcal{O}(p^2)$

The Alternating Direction Implicit (ADI) method
(for solving matrix equations) [Wachspress, 1987]

$$TX + XT^T = F$$

- Based on **structured eigenvalues**
- Optimal parameters known [Lu & Wachspress, 1991]

still works for spectral

set $X_0 = 0$

pick shift parameters p_j

for $j = 0, \ldots, J$?

solve $X_{j+1/2}(T^T + p_j I) = F - (T - p_j I)X_j$

solve $(T + p_j I)X_{j+1} = F - X_{j+1/2}(T^T - p_j I)$

} Thomas algorithm $\mathcal{O}(p^2)$

If eigenvalues of T lie in $[a, b]$, then for $0 < \epsilon < 1$, $\frac{\|X - X_J\|_2}{\|X\|_2} \leqslant \epsilon$ when $J > \frac{1}{\pi^2} \log \frac{4b}{a} \log \frac{4}{\epsilon}$

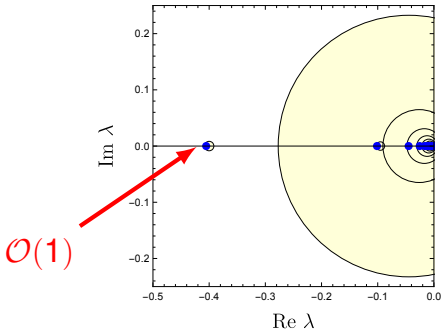
[Lu & Wachspress, 1991]

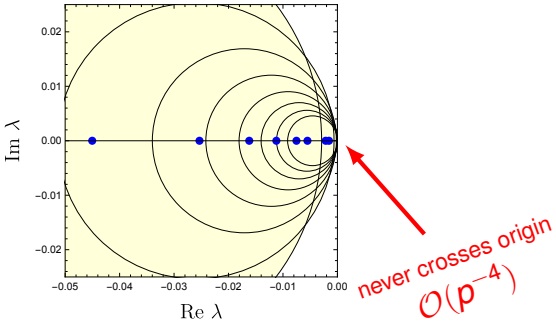
Theorem

Every eigenvalue of a complex $n \times n$ matrix A lies within at least one disc centered at a_{ii} of radius $\sum_{j \neq i} |a_{ij}|$.

Theorem

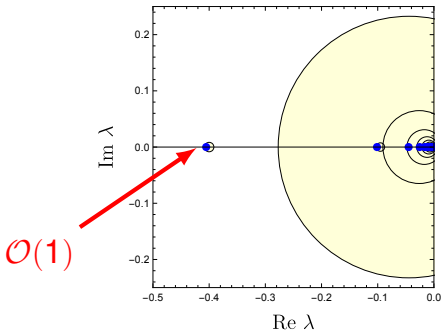
Every eigenvalue of a complex $n \times n$ matrix A lies within at least one disc centered at a_{ii} of radius $\sum_{j \neq i} |a_{ij}|$.



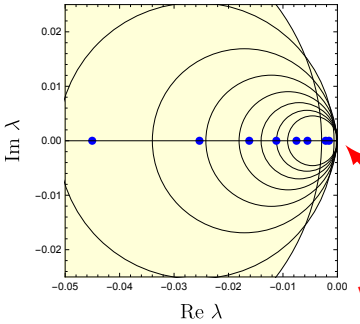


Theorem

Every eigenvalue of a complex $n \times n$ matrix A lies within at least one disc centered at a_{ii} of radius $\sum_{j \neq i} |a_{ij}|$.



$\mathcal{O}(1)$



never crosses origin

$\mathcal{O}(p^{-4})$

$$J \sim \mathcal{O}\left(\log p \log \frac{1}{\epsilon}\right)$$

A fast spectrally-accurate Poisson solver

For a given error tolerance $0 < \epsilon < 1$:

	Cost	
1. Compute $C^{(3/2)}$ coefficients of f	$\mathcal{O}(p^2(\log p)^2 \log 1/\epsilon)$	[Hale & Townsend, 2014]
2. Solve matrix equation using ADI		
▸ $\mathcal{O}(p^2)$ per iteration	$\mathcal{O}(p^2 \log p \log 1/\epsilon)$	
▸ $\mathcal{O}(\log p \log 1/\epsilon)$ iterations		
3. Convert solution to Chebyshev	$\mathcal{O}(p^2(\log p)^2 \log 1/\epsilon)$	[Hale & Townsend, 2014]
	$\mathcal{O}(p^2(\log p)^2 \log 1/\epsilon)$	

A similar method in 1979
...but a different conclusion!

"The accurate solution of Poisson's equation by expansion in Chebyshev polynomials"

[Haidvogel & Zang, 1979]

Dale Haidvogel

$$D_2 X + X D_2^T = F$$

Chebyshev differentiation

inverse is tridiagonal

- Concluded **ADI is too slow to be practical!**

A similar method in 1979
...but a different conclusion!

"The accurate solution of Poisson's equation by expansion in Chebyshev polynomials"

[Haidvogel & Zang, 1979]

Dale Haidvogel

$$D_2 X + X D_2^T = F$$

Chebyshev differentiation

no bound on iterations

- Concluded **ADI is too slow to be practical!**

A similar method in 1979
...but a different conclusion!

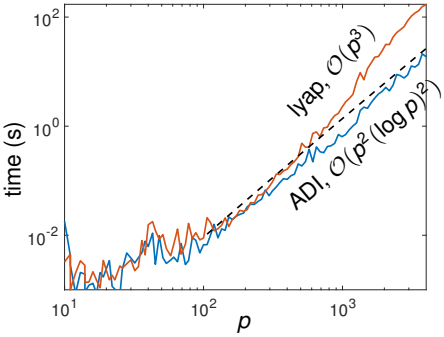
"The accurate solution of Poisson's equation by expansion in Chebyshev polynomials"
[Haidvogel & Zang, 1979]


Dale Haidvogel

$$D_2 X + X D_2^T = F$$

Chebyshev differentiation

no bound on iterations



- Concluded **ADI is too slow to be practical!**

A similar method in 1979
...but a different conclusion!

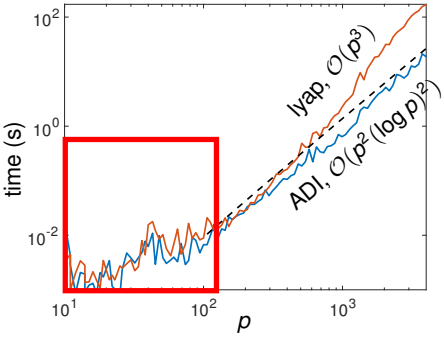
"The accurate solution of Poisson's equation by expansion in Chebyshev polynomials"
[Haidvogel & Zang, 1979]


Dale Haidvogel

$$D_2 X + X D_2^T = F$$

Chebyshev differentiation

no bound on iterations



- Concluded **ADI is too slow to be practical!**

Additional features

Our fast solver can also...

- ✓ exploit low rank right-hand sides using factored ADI
- ✓ handle arbitrary Dirichlet BCs
- ✓ handle more complex BCs (e.g. Neumann)
- ✓ apply to other strongly elliptic PDEs with nice spectra

Additional features


Alex Townsend


Heather Wilber

Our fast solver can also...

low-rank RHS $\Rightarrow$ low-rank solution

- ✓ exploit low rank right-hand sides using factored ADI
- ✓ handle arbitrary Dirichlet BCs
- ✓ handle more complex BCs (e.g. Neumann)
- ✓ apply to other strongly elliptic PDEs with nice spectra

Additional features


Alex Townsend


Heather Wilber

Our fast solver can also... low-rank RHS $\Rightarrow$ low-rank solution

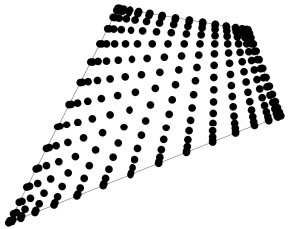
- ✓ exploit low rank right-hand sides using factored ADI
- ✓ handle arbitrary Dirichlet BCs
- ✓ handle more complex BCs (e.g. Neumann)
- ✓ apply to other strongly elliptic PDEs with nice spectra

Coming soon to Chebfun2!

Poisson on a quadrilateral

Poisson on a quadrilateral
A change of variables

Idea: Transform to $[-1, 1]^2$ and discretize using a sparse spectral method

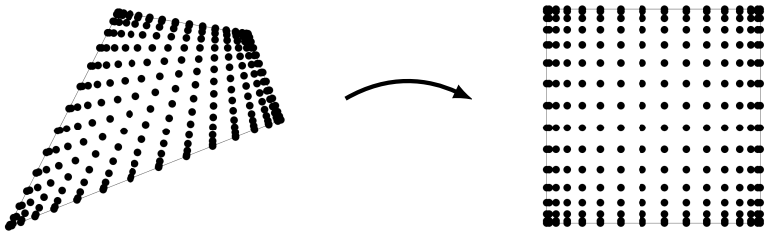


(x, y)

Poisson

Poisson on a quadrilateral
A change of variables

Idea: Transform to $[-1, 1]^2$ and discretize using a sparse spectral method

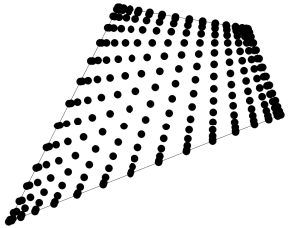


(x, y)
Poisson

(ξ, η)
not Poisson

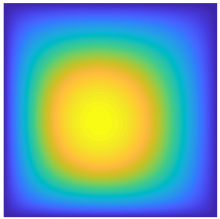
Poisson on a quadrilateral
A change of variables

Idea: Transform to $[-1, 1]^2$ and discretize using a sparse spectral method



(x, y)

Poisson

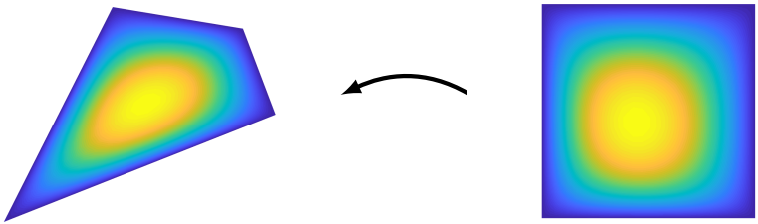


(ξ, η)

not Poisson

Poisson on a quadrilateral
A change of variables

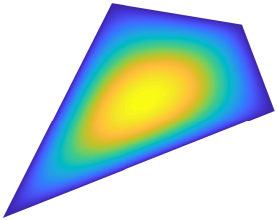
Idea: Transform to $[-1, 1]^2$ and discretize using a sparse spectral method



(x, y)
Poisson

(ξ, η)
not Poisson

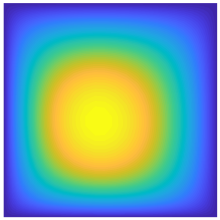
Idea: Transform to $[-1,1]^2$ and discretize using a sparse spectral method



(x, y)

Poisson

$$u_{xx} + u_{yy} = f$$



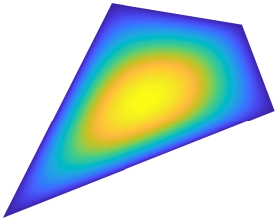
(ξ, η)

not Poisson

$$u_{\xi\xi}(\xi_x^2 + \xi_y^2) + 2u_{\xi\eta}(\xi_x\eta_x + \xi_y\eta_y) + u_{\eta\eta}(\eta_x^2 + \eta_y^2) + u_\xi(\xi_{xx} + \xi_{yy}) + u_\eta(\eta_{xx} + \eta_{yy}) = \tilde{f}$$

Poisson on a quadrilateral
A change of variables

Idea: Transform to $[-1, 1]^2$ and discretize using a sparse spectral method

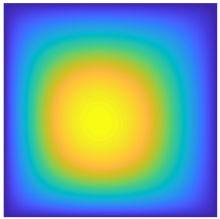


(x, y)

Poisson

splitting rank 2

$$A_1 X B_1^T + A_2 X B_2^T = F$$



(ξ, η)

not Poisson

splitting rank k

$$\sum_{i=1}^{k} A_i X B_i^T = \tilde{F}$$

We can rewrite the matrix equation

$$\sum_{i=1}^{k} A_i X B_i^T = F$$

as a system of equations by introducing constraints:

$$\sum_{i=1}^{k} A_i X_i B_i^T = F, \qquad X_1 = \cdots = X_k$$

In matrix form this is

$$\begin{bmatrix} I & -I & & \\ & \ddots & \ddots & \\ & & I & -I \\ B_1 \otimes A_1 & B_2 \otimes A_2 & \cdots & B_k \otimes A_k \end{bmatrix} \begin{bmatrix} X_1(:) \\ X_2(:) \\ \vdots \\ X_k(:) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ F(:) \end{bmatrix}$$

Note that we can multiply this matrix by a vector fast without ever forming it, since A_i and B_i are sparse, almost-banded matrices.

$$\begin{bmatrix} I & & -I & & \\ & \ddots & & \ddots & \\ & & & I & -I \\ B_1 \otimes A_1 & & B_2 \otimes A_2 & \cdots & B_k \otimes A_k \end{bmatrix} \begin{bmatrix} X_1(:) \\ X_2(:) \\ \vdots \\ X_k(:) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ F(:) \end{bmatrix}$$

This motivates us to use an iterative **Krylov method**.

To obtain convergence independent of p, we will need a good **preconditioner**.

Preconditioning

Finding a good preconditioner to solve a given sparse linear system is often viewed as a combination of art and science.

— Yousef Saad

Preconditioning

Note that the block LU decomposition of the matrix is easy to compute:

$$\begin{bmatrix} I & & -I & & \\ & \ddots & & \ddots & \\ & & & I & -I \\ B_1 \otimes A_1 & & B_2 \otimes A_2 & \cdots & B_k \otimes A_k \end{bmatrix} =$$

$$\begin{bmatrix} I & & & \\ & \ddots & & \\ & & I & \\ B_1 \otimes A_1 & \sum_{i=1}^{2} B_i \otimes A_i & \cdots & \sum_{i=1}^{k} B_i \otimes A_i \end{bmatrix} \begin{bmatrix} I & -I & & \\ & \ddots & \ddots & \\ & & \ddots & -I \\ & & & I \end{bmatrix}$$

Preconditioning

Note that the block LU decomposition of the matrix is easy to compute:

$$\begin{bmatrix} I & -I & & \\ & \ddots & \ddots & \\ & & I & -I \\ B_1 \otimes A_1 & B_2 \otimes A_2 & \cdots & B_k \otimes A_k \end{bmatrix} \approx$$

$$\begin{bmatrix} I & & & \\ & \ddots & & \\ & & I & \\ B_1 \otimes A_1 & B_1 \otimes A_1 & \cdots & B_1 \otimes A_1 \end{bmatrix} \begin{bmatrix} I & -I & & \\ & \ddots & \ddots & \\ & & \ddots & -I \\ & & & I \end{bmatrix} = P$$

$P^{-1}v$ can be computed in $\mathcal{O}(p^2k)$ time.

Preconditioning

Note that the block LU decomposition of the matrix is easy to compute:

$$\begin{bmatrix} I & -I & & \\ & \ddots & \ddots & \\ & & I & -I \\ B_1 \otimes A_1 & B_2 \otimes A_2 & \cdots & B_k \otimes A_k \end{bmatrix} \approx$$

$$\begin{bmatrix} I & & & \\ & \ddots & & \\ & & I & \\ B_1 \otimes A_1 & B_1 \otimes A_1 & \cdots & B_1 \otimes A_1 \end{bmatrix} \begin{bmatrix} I & -I & & \\ & \ddots & \ddots & \\ & & \ddots & -I \\ & & & I \end{bmatrix} = P$$

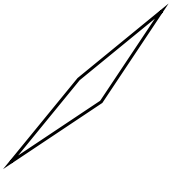
$P^{-1}v$ can be computed in $\mathcal{O}(p^2 k)$ time.

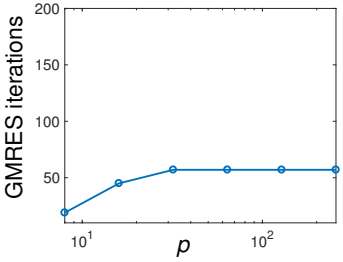
Preconditioning

Does it work?

Preconditioning

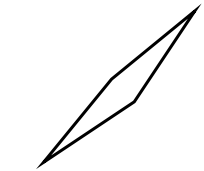
Does it work?

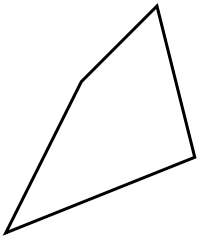


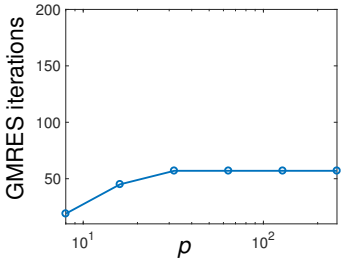


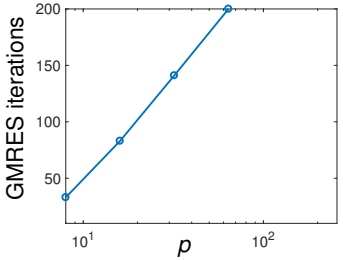
Preconditioning

Does it work?

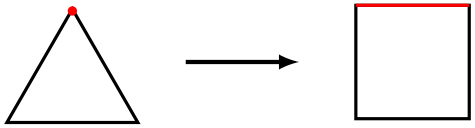






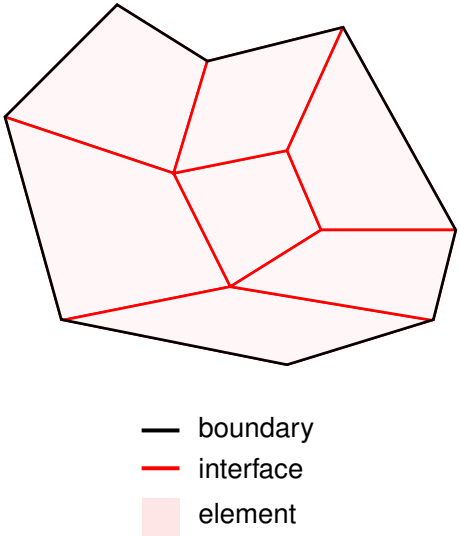


Duffy transform? Introduces singularity



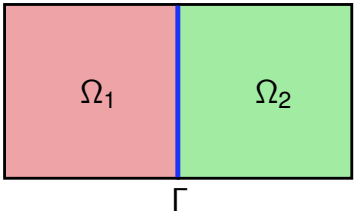
Instead, divide any convex k-polygon into k quadrilaterals

- ✓ An element solver (local)
- An interface solver (global)



The Schur complement method

Suppose we wish to solve a PDE $Au = f$ on two glued squares:



We can separate the interface unknowns from the subdomain interiors and write the PDE as

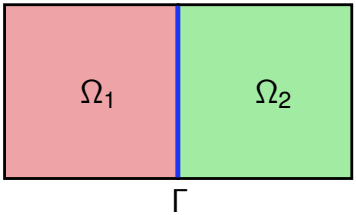
$$\begin{bmatrix} A_{11} & & A_{1\Gamma} \\ & A_{22} & A_{2\Gamma} \\ A_{\Gamma 1} & A_{\Gamma 2} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_\Gamma \end{bmatrix}$$

which ensures continuity and continuity of the derivative across Γ with

$$A_{\Gamma *} = \text{evaluate normal derivative}, \quad A_{*\Gamma} = \text{inject boundary data}$$

The Schur complement method

Suppose we wish to solve a PDE $Au = f$ on two glued squares:



We can find the values on the interface by solving the smaller system

$$\Sigma u_\Gamma = f_\Gamma - A_{\Gamma 1} A_{11}^{-1} f_1 - A_{\Gamma 2} A_{22}^{-1} f_2$$

where

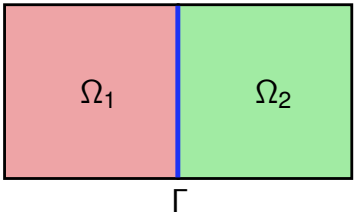
$$\Sigma = A_{\Gamma\Gamma} - A_{\Gamma 1} A_{11}^{-1} A_{1\Gamma} - A_{\Gamma 2} A_{22}^{-1} A_{2\Gamma}$$

is called the Schur complement of $A_{\Gamma\Gamma}$. Once we know u_Γ we can find the interior values by solving

$$A_{11} u_1 = f_1 - A_{1\Gamma} u_\Gamma, \qquad A_{22} u_2 = f_2 - A_{2\Gamma} u_\Gamma$$

The Schur complement method

Suppose we wish to solve a PDE $Au = f$ on two glued squares:



The inverse operator is therefore

$$A^{-1} = \begin{bmatrix} I & & -A_{11}^{-1}A_{1\Gamma} \\ & I & -A_{22}^{-1}A_{2\Gamma} \\ & & I \end{bmatrix} \begin{bmatrix} A_{11}^{-1} & & \\ & A_{22}^{-1} & \\ & & \Sigma^{-1} \end{bmatrix} \begin{bmatrix} I & & \\ & I & \\ -A_{\Gamma 1}A_{11}^{-1} & -A_{\Gamma 2}A_{22}^{-1} & I \end{bmatrix}$$

where

$$\Sigma = A_{\Gamma\Gamma} - A_{\Gamma 1}A_{11}^{-1}A_{1\Gamma} - A_{\Gamma 2}A_{22}^{-1}A_{2\Gamma}$$

The Schur complement method

The inverse can be factored as

$$
\begin{aligned}
A^{-1} &= \begin{bmatrix} A_{11}^{-1} & & \\ & A_{22}^{-1} & \\ & & I \end{bmatrix} \begin{bmatrix} I & & -A_{1\Gamma} \\ & I & -A_{2\Gamma} \\ & & I \end{bmatrix} \begin{bmatrix} A_{11} & & \\ & A_{22} & \\ & & \Sigma^{-1} \end{bmatrix} \begin{bmatrix} I & & \\ & I & \\ -A_{\Gamma 1} & -A_{\Gamma 2} & I \end{bmatrix} \begin{bmatrix} A_{11}^{-1} & & \\ & A_{22}^{-1} & \\ & & I \end{bmatrix} \\
&= \begin{bmatrix} A_{11}^{-1} & & \\ & A_{22}^{-1} & \\ & & I \end{bmatrix} \begin{bmatrix} I & & -A_{1\Gamma} \\ & I & -A_{2\Gamma} \\ & & I \end{bmatrix} \begin{bmatrix} I & & \\ & I & \\ & & \Sigma^{-1} \end{bmatrix} \left\{ I + \begin{bmatrix} I & & \\ & I & \\ -A_{\Gamma 1} & -A_{\Gamma 2} & I \end{bmatrix} \begin{bmatrix} A_{11}^{-1} & & \\ & A_{22}^{-1} & \\ & & I \end{bmatrix} \right\} \\
&\approx \begin{bmatrix} A_{11}^{\dagger} & & \\ & A_{22}^{\dagger} & \\ & & I \end{bmatrix} \begin{bmatrix} I & & -A_{1\Gamma} \\ & I & -A_{2\Gamma} \\ & & I \end{bmatrix} \begin{bmatrix} I & & \\ & I & \\ & & \Sigma^{\dagger} \end{bmatrix} \left\{ I + \begin{bmatrix} I & & \\ & I & \\ -A_{\Gamma 1} & -A_{\Gamma 2} & I \end{bmatrix} \begin{bmatrix} A_{11}^{\dagger} & & \\ & A_{22}^{\dagger} & \\ & & I \end{bmatrix} \right\}
\end{aligned}
$$

where † denotes an approximate inverse.

This extends naturally to k subdomains.

The Schur complement method
An algorithm

1. Solve subproblems:
$$\left.\begin{aligned} A_{11}\hat{u}_1 &= f_1 \\ &\vdots \\ A_{kk}\hat{u}_k &= f_k \end{aligned}\right\} \text{ zero Dirichlet BCs}$$

2. Solve interface problem: $\Sigma u_\Gamma = f_\Gamma - A_{\Gamma 1}\hat{u}_1 - \cdots - A_{\Gamma k}\hat{u}_k.$

3. Solve subproblems:
$$\left.\begin{aligned} A_{11}u_1 &= f_1 \\ &\vdots \\ A_{kk}u_k &= f_k \end{aligned}\right\} u_\Gamma \text{ Dirichlet BCs}$$

We have a fast solver for 1 & 3. Note that 1 & 3 can be parallelized as well. It remains to solve 2 fast.

The Schur complement method
An algorithm

1. Solve subproblems:
$$\left.\begin{aligned} A_{11}\hat{u}_1 &= f_1 \\ &\vdots \\ A_{kk}\hat{u}_k &= f_k \end{aligned}\right\} \text{ zero Dirichlet BCs}$$

2. Solve interface problem: $\Sigma u_\Gamma = f_\Gamma - A_{\Gamma 1}\hat{u}_1 - \cdots - A_{\Gamma k}\hat{u}_k.$

3. Solve subproblems:
$$\left.\begin{aligned} A_{11}u_1 &= f_1 \\ &\vdots \\ A_{kk}u_k &= f_k \end{aligned}\right\} u_\Gamma \text{ Dirichlet BCs}$$

We have a fast solver for 1 & 3. Note that 1 & 3 can be parallelized as well. It remains to solve 2 fast.

The Schur complement method
An algorithm

Note that we can apply Σ to a vector fast without explicitly constructing it since

$$\begin{aligned}\Sigma u_\Gamma &= (A_{\Gamma\Gamma} - A_{\Gamma 1} A_{11}^{-1} A_{1\Gamma} - \cdots - A_{\Gamma k} A_{kk}^{-1} A_{k\Gamma}) u_\Gamma \\ &= A_{\Gamma\Gamma} u_\Gamma - \mathrm{D2N}_1(u_\Gamma) - \cdots - \mathrm{D2N}_k(u_\Gamma)\end{aligned}$$

where $\mathrm{D2N}_i$ is the Dirichlet-to-Neumann map for subdomain i, which does:

- Solve $A_{ii} u_i = 0$ with u_Γ Dirichlet BC
- Evaluate the normal derivative of u_i on Γ

$\mathrm{D2N}_i(u_\Gamma)$ takes $\mathcal{O}(p^2 \log p)$ and $A_{\Gamma\Gamma} u_\Gamma$ takes $\mathcal{O}(p^2)$. We wish to solve

$$\Sigma u_\Gamma = f_\Gamma - A_{\Gamma 1} \hat{u}_1 - \cdots - A_{\Gamma k} \hat{u}_k$$

approximately via an iterative method, and use it to design a preconditioner for the **global problem**, $A^\dagger \approx A^{-1}$.

The Schur complement method
An algorithm

Note that we can apply Σ to a vector fast without explicitly constructing it since

$$\begin{aligned}\Sigma u_\Gamma &= (A_{\Gamma\Gamma} - A_{\Gamma 1}A_{11}^{-1}A_{1\Gamma} - \cdots - A_{\Gamma k}A_{kk}^{-1}A_{k\Gamma})u_\Gamma \\ &= A_{\Gamma\Gamma}u_\Gamma - \mathrm{D2N}_1(u_\Gamma) - \cdots - \mathrm{D2N}_k(u_\Gamma)\end{aligned}$$

where $\mathrm{D2N}_i$ is the Dirichlet-to-Neumann map for subdomain i, which does:

- Solve $A_{ii}u_i = 0$ with u_Γ Dirichlet BC
- Evaluate the normal derivative of u_i on Γ

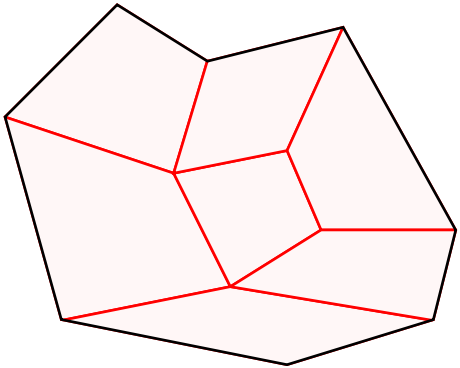
$\mathrm{D2N}_i(u_\Gamma)$ takes $\mathcal{O}(p^2 \log p)$ and $A_{\Gamma\Gamma}u_\Gamma$ takes $\mathcal{O}(p^2)$. We wish to solve

$$\Sigma u_\Gamma = f_\Gamma - A_{\Gamma 1}\hat{u}_1 - \cdots - A_{\Gamma k}\hat{u}_k$$

approximately via an iterative method, and use it to design a preconditioner for the **global problem**, $A^\dagger \approx A^{-1}$.

- ✓ An element solver (local)
- ? An interface solver (global)

Need a good preconditioner for Σ!



- Optimal complexity in h and p: $\mathcal{O}(p^2/h^2)$
- True, automatic hp-adaptivity (without concern of ill-conditioning or cost)
- Solution of uniformly elliptic PDEs with general boundary conditions
- High accuracy on elements independent of their aspect ratio
- Ability to handle unstructured meshes of arbitrary convex polygons
- High parallelizability

```
import ultraSEM

mesh = ultraSEM.mesh(pts, tri) # create mesh
pdo  = ultraSEM.pdo(1, 0, 0)   # define Poisson
f  = 1
bc = 0

S = ultraSEM.solver(mesh, pdo) # build the solver
u = S.solve(f, bc)             # solve the PDE
```

Discontinuous Galerkin methods

Discontinuous Galerkin?
What is it? Why would I use it?

Discontinuous Galerkin?
What is it? Why would I use it?

"finite volume"
Discontinuous

Discontinuous Galerkin?
What is it? Why would I use it?

"finite volume"
Discontinuous

+

"finite element"
Galerkin

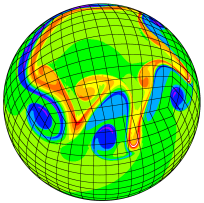
Discontinuous Galerkin?
What is it? Why would I use it?

"finite volume" Discontinuous + "finite element" Galerkin

	Complex geometry	High-order accuracy and *hp*-adaptivity	Explicit semi-discrete form	Stability for conservation laws	Elliptic problems
FD	×	✓	✓	✓	✓
FV	✓	×	✓	✓	(✓)
FEM	✓	✓	×	×	✓
DG	✓	✓	✓	✓	(✓)

The natural choices for fluid and solid representation conflict:

Fluid simulation



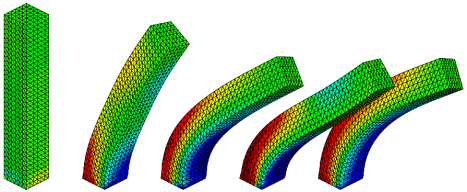
Interested in **velocity**

↓

Track on fixed grid

Eulerian

Solid simulation



Interested in **displacement**

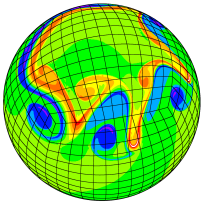
↓

Move mesh with material

Lagrangian

The natural choices for fluid and solid representation conflict:

Fluid simulation



Interested in **velocity**

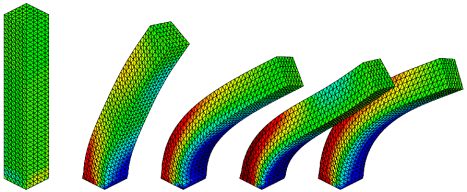
↓

Track on fixed grid

Eulerian

?

Solid simulation



Interested in **displacement**

↓

Move mesh with material

Lagrangian

The motion function χ maps the undeformed reference state to the deformed state:

$$\boldsymbol{x} = \chi(\boldsymbol{X}, t)$$

Eulerian conservation of mass:

$$\rho_t = \boldsymbol{u} \cdot \nabla \rho - \rho \nabla \cdot \boldsymbol{u}$$

Euler conservation of momentum:

$$\boldsymbol{u}_t = -\boldsymbol{u} \cdot \nabla \boldsymbol{u} - \frac{\nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{g}}{\rho}$$

The deformation gradient is then

$$\boldsymbol{F}(\boldsymbol{X}, t) = \frac{\partial \boldsymbol{\chi}(\boldsymbol{X}, t)}{\partial \boldsymbol{X}}$$

Define the **reference map** ξ to map the deformed state to the reference state:

$$\boldsymbol{X} = \boldsymbol{\xi}(\boldsymbol{x}, t) = \chi^{-1}(\boldsymbol{x}, t)$$

Then we can write the deformation gradient as

$$\boldsymbol{F}\left(\boldsymbol{\xi}(\boldsymbol{x}, t), t\right) = \left(\nabla \boldsymbol{\xi}(\boldsymbol{x}, t)\right)^{-1}$$

Large deformation constitutive relations can be simulated since $\boldsymbol{\xi} \rightarrow \boldsymbol{F} \rightarrow \boldsymbol{\sigma}$.

The original location of a material point never changes, so ξ obeys

$$\boxed{\boldsymbol{\xi}_t + \boldsymbol{u} \cdot \nabla \boldsymbol{\xi} = 0}$$

Interface between solid and fluid is tracked by a level set but phase change is blurred. A transition zone between the two allows for simpler computations.

Eulerian fluid-structure interaction
Current limitations

- finite differences
  - high-order accuracy possible but not practical as stencil size increases
- interface is blurred
  - relies on grid refinement for accuracy
- requires extrapolation of reference map outside of the solid
  - can lead to artifacts

Eulerian fluid-structure interaction
Current limitations

- finite differences
  - high-order accuracy possible but not practical as stencil size increases
- interface is blurred
  - relies on grid refinement for accuracy
- requires extrapolation of reference map outside of the solid
  - can lead to artifacts

Can we use discontinuous Galerkin methods instead?

Implicit mesh DG
A new hope

Traditional DG methods rely on an unstructured mesh to capture geometry.

For an Eulerian method, we'd like to:

- store unknowns on a **fixed background grid**
- represent geometry using implicitly defined **level sets**

Can DG do this?

Implicit mesh DG
A new hope

Traditional DG methods rely on an unstructured mesh to capture geometry.

For an Eulerian method, we'd like to:

- store unknowns on a **fixed background grid**
- represent geometry using implicitly defined **level sets**

Can DG do this? **Yes!**

Robert Saye

Implicit mesh DG
A new hope

Traditional DG methods rely on an unstructured mesh to capture geometry.

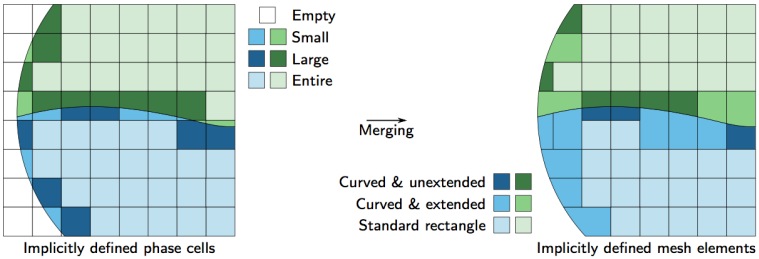
For an Eulerian method, we'd like to:

- store unknowns on a **fixed background grid**
- represent geometry using implicitly defined **level sets**

Robert Saye

Can DG do this? **Yes!**

Elements are defined based on level sets



[Saye, 2016]

Implicit mesh DG
A new hope

Traditional DG methods rely on an unstructured mesh to capture geometry.

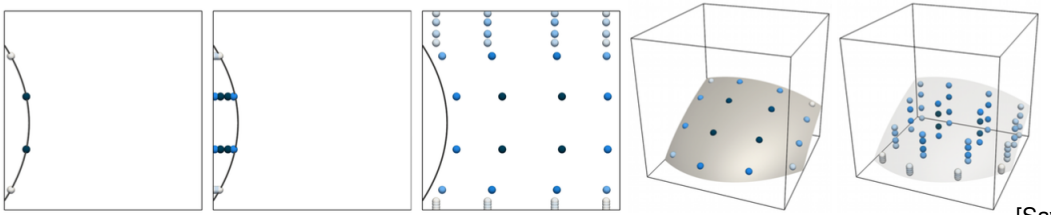
For an Eulerian method, we'd like to:

- store unknowns on a **fixed background grid**
- represent geometry using implicitly defined **level sets**

Can DG do this? **Yes!**

Robert Saye

High-order quadrature rules are computed based on 1D root-finding



[Saye, 2015]

Given a mesh $\mathcal{T}_h$, we introduce the following approximation spaces:

$$W_h^p = \left\{ w \in L^2(\mathcal{T}_h) \quad : w|_K \in \mathcal{P}^p(K) \qquad \forall\ K \in \mathcal{T}_h \right\}$$
$$\boldsymbol{V}_h^p = \left\{ \boldsymbol{v} \in \left[L^2(\mathcal{T}_h)\right]^d : \boldsymbol{v}|_K \in \left[\mathcal{P}^p(K)\right]^d \ \forall\ K \in \mathcal{T}_h \right\}$$

The L^2 inner products over an element K are given by

$$(w, v)_K = \int_K wv, \qquad (\boldsymbol{w}, \boldsymbol{v})_K = \sum_{i=1}^{d} (w_i, v_i)_K, \qquad \langle \eta, \mu \rangle_{\partial K} = \int_{\partial K} \eta\mu,$$

and we define inner products over the mesh as

$$(w, v)_{\mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} (w, v)_K, \qquad (\boldsymbol{w}, \boldsymbol{v})_{\mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} (\boldsymbol{w}, \boldsymbol{v})_K, \qquad \langle \eta, \mu \rangle_{\mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} \langle \eta, \mu \rangle_{\partial K}.$$

Given a mesh $\mathcal{T}_h$, we introduce the following approximation spaces:

Lagrange basis

$$W_h^p = \left\{ w \in L^2(\mathcal{T}_h) \quad : w|_K \in \mathcal{P}^p(K) \quad \forall\ K \in \mathcal{T}_h \right\}$$
$$\boldsymbol{V}_h^p = \left\{ \boldsymbol{v} \in \left[L^2(\mathcal{T}_h)\right]^d : \boldsymbol{v}|_K \in \left[\mathcal{P}^p(K)\right]^d \ \forall\ K \in \mathcal{T}_h \right\}$$

The L^2 inner products over an element K are given by

$$(w, v)_K = \int_K wv, \qquad (\boldsymbol{w}, \boldsymbol{v})_K = \sum_{i=1}^{d} (w_i, v_i)_K, \qquad \langle \eta, \mu \rangle_{\partial K} = \int_{\partial K} \eta\mu,$$

and we define inner products over the mesh as

$$(w, v)_{\mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} (w, v)_K, \qquad (\boldsymbol{w}, \boldsymbol{v})_{\mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} (\boldsymbol{w}, \boldsymbol{v})_K, \qquad \langle \eta, \mu \rangle_{\mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} \langle \eta, \mu \rangle_{\partial K}.$$

We wish to solve the reference map equation

$$\boldsymbol{\xi}_t + \boldsymbol{u} \cdot \nabla \boldsymbol{\xi} = 0 \tag{1}$$

for $\boldsymbol{\xi}(\boldsymbol{x}, t)$ using DG, where for now we assume that $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$ is given. We can write (1) in conservative form as

$$\boldsymbol{\xi}_t + \nabla \cdot (\boldsymbol{u}\boldsymbol{\xi}) = (\nabla \cdot \boldsymbol{u})\boldsymbol{\xi}$$

or, more explicity,

$$\frac{\partial \xi_1}{\partial t} + \nabla \cdot (\boldsymbol{u}\xi_1) = (\nabla \cdot \boldsymbol{u})\xi_1$$

$$\frac{\partial \xi_2}{\partial t} + \nabla \cdot (\boldsymbol{u}\xi_2) = (\nabla \cdot \boldsymbol{u})\xi_2$$

Let K be an element in a mesh $\mathcal{T}_h$. To derive the weak form, we multiply by a test function $w \in W_h^p$ and integrate by parts to obtain:

$$\left(\frac{\partial \xi_1}{\partial t}, w\right)_K - (\boldsymbol{u}\xi_1, \nabla w)_K + \langle \boldsymbol{u}\xi_1 \cdot \boldsymbol{n}, w\rangle_{\partial K} = ((\nabla \cdot \boldsymbol{u})\xi_1, w)_K$$

$$\left(\frac{\partial \xi_2}{\partial t}, w\right)_K - (\boldsymbol{u}\xi_2, \nabla w)_K + \langle \boldsymbol{u}\xi_2 \cdot \boldsymbol{n}, w\rangle_{\partial K} = ((\nabla \cdot \boldsymbol{u})\xi_2, w)_K$$

We now make the following approximations:

- Replace $\boldsymbol{\xi}$ with $\boldsymbol{\xi}_h \in V_h^p$ in the bulk
- Replace $\boldsymbol{u\xi}$ with a *numerical flux* $\widehat{\boldsymbol{u\xi}_h}$ on the boundary
- Define $\widehat{\boldsymbol{u\xi}_h}$ in terms of $\boldsymbol{\xi}_h$
- Sum over all elements $K \in \mathcal{T}_h$

Find $\boldsymbol{\xi}_h \in \boldsymbol{V}_h^p$ such that

$$\left(\frac{\partial \xi_h^1}{\partial t}, w\right)_{\mathcal{T}_h} - \left(\boldsymbol{u}\xi_h^1, \nabla w\right)_{\mathcal{T}_h} + \left\langle \widehat{\boldsymbol{u}\xi_h^1} \cdot \boldsymbol{n}, w \right\rangle_{\partial\mathcal{T}_h} = \left((\nabla \cdot \boldsymbol{u})\xi_h^1, w\right)_{\mathcal{T}_h}$$

$$\left(\frac{\partial \xi_h^2}{\partial t}, w\right)_{\mathcal{T}_h} - \left(\boldsymbol{u}\xi_h^2, \nabla w\right)_{\mathcal{T}_h} + \left\langle \widehat{\boldsymbol{u}\xi_h^2} \cdot \boldsymbol{n}, w \right\rangle_{\partial\mathcal{T}_h} = \left((\nabla \cdot \boldsymbol{u})\xi_h^2, w\right)_{\mathcal{T}_h}$$

for all $w \in W_h^p$. To complete the method, we still need to define the numerical flux $\widehat{\boldsymbol{u}\xi_h}$. A natural choice for linear convection is the **upwind flux**:

$$\widehat{\boldsymbol{u}\xi_h} = \frac{1}{2}(\boldsymbol{u} \cdot \boldsymbol{n})(\xi_h^+ + \xi_h^-) + \frac{1}{2}|\boldsymbol{u} \cdot \boldsymbol{n}|(\xi_h^+ - \xi_h^-)$$

where $\xi_h^\pm$ denotes the solution on neighboring elements $K^\pm$ of each face in $\partial\mathcal{T}_h$.

Consider the incompressible velocity field

$$\boldsymbol{u}(\boldsymbol{x}, t) = \begin{pmatrix} b \sin ax \cos by \sin ct \\ -a \cos ax \sin by \sin ct \end{pmatrix}$$

Since $\boldsymbol{u}$ is incompressible, the right-hand side of (1) drops out. So we wish to solve

$$\boldsymbol{\xi}_t + \nabla \cdot (\boldsymbol{u}\boldsymbol{\xi}) = 0$$

with initial condition $\boldsymbol{\xi}(\boldsymbol{x}, 0) = \boldsymbol{x}$.

Eulerian fluid-structure interaction
An incompressible test case

It can be shown that an exact solution is given by

$$\boldsymbol{\xi}(\boldsymbol{x}, t) = \begin{pmatrix} \frac{1}{a}\cos^{-1}\left(k(\boldsymbol{x})\,\mathrm{cd}(\psi(\boldsymbol{x}, t))\right) \\ \frac{1}{b}\cos^{-1}\left(k(\boldsymbol{x})\,\mathrm{sn}(\psi(\boldsymbol{x}, t))\right) \end{pmatrix}$$

where $\psi = F(\phi, k) + \frac{ab(1-\cos ct)}{c}$, $k = \sqrt{1 - \sin^2 ax \sin^2 by}$, $\phi = \sin^{-1}\frac{\cos by}{k}$, cd & sn are Jacobi elliptic functions, and F is the incomplete elliptic integral of the first kind.

Eulerian fluid-structure interaction
An incompressible test case

Exact solution

Computed solution

Incompressible Navier-Stokes:

$$\rho\left(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \mu\nabla^2\boldsymbol{u} + \boldsymbol{f}, \quad \text{in } \Omega, \tag{2}$$

$$\nabla\cdot\boldsymbol{u} = 0, \quad \text{in } \Omega. \tag{3}$$

We'd like to advance (2) in time while maintaining (3).

Projection method:

1. Compute an intermediate velocity $\boldsymbol{u}^*$: $\dfrac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -(\boldsymbol{u}^n\cdot\nabla)\boldsymbol{u}^n + \mu\nabla^2\boldsymbol{u} + \dfrac{\boldsymbol{f}}{\rho}$
2. Solve for the pressure that will maintain (3): $\nabla\cdot\left(\dfrac{\nabla p^{n+1}}{\rho}\right) = \dfrac{\nabla\cdot\boldsymbol{u}^*}{\Delta t}$
3. Compute $\boldsymbol{u}^{n+1}$ to be divergence-free: $\dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\dfrac{\nabla p^{n+1}}{\rho}$

Eulerian fluid-structure interaction
What about the fluid?

Incompressible Navier-Stokes:

solid coupling

$$\rho\left(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \mu\nabla^2\boldsymbol{u} + \boldsymbol{f}, \quad \text{in } \Omega, \tag{2}$$

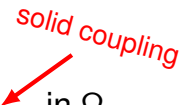
$$\nabla\cdot\boldsymbol{u} = 0, \quad \text{in } \Omega. \tag{3}$$

We'd like to advance (2) in time while maintaining (3).

Projection method:

1. Compute an intermediate velocity $\boldsymbol{u}^*$: $\dfrac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -(\boldsymbol{u}^n\cdot\nabla)\boldsymbol{u}^n + \mu\nabla^2\boldsymbol{u} + \dfrac{\boldsymbol{f}}{\rho}$
2. Solve for the pressure that will maintain (3): $\nabla\cdot\left(\dfrac{\nabla p^{n+1}}{\rho}\right) = \dfrac{\nabla\cdot\boldsymbol{u}^*}{\Delta t}$
3. Compute $\boldsymbol{u}^{n+1}$ to be divergence-free: $\dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\dfrac{\nabla p^{n+1}}{\rho}$

Incompressible Navier-Stokes:



$$\rho\left(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \mu\nabla^2\boldsymbol{u} + \boldsymbol{f}, \quad \text{in } \Omega, \tag{2}$$

$$\nabla\cdot\boldsymbol{u} = 0, \quad \text{in } \Omega. \tag{3}$$

We'd like to advance (2) in time while maintaining (3).

Projection method:

1. Compute an intermediate velocity $\boldsymbol{u}^*$: $\dfrac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -(\boldsymbol{u}^n\cdot\nabla)\boldsymbol{u}^n + \mu\nabla^2\boldsymbol{u} + \dfrac{\boldsymbol{f}}{\rho}$
2. Solve for the pressure that will maintain (3): $\nabla\cdot\left(\dfrac{\nabla p^{n+1}}{\rho}\right) = \dfrac{\nabla\cdot\boldsymbol{u}^*}{\Delta t}$
3. Compute $\boldsymbol{u}^{n+1}$ to be divergence-free: $\dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\dfrac{\nabla p^{n+1}}{\rho}$

Eulerian fluid-structure interaction
What about the fluid?

We'd like to maintain high-order accuracy when coupling to a solid. However, fractional stepping of $\boldsymbol{u}$...

- creates nonphysical coupling between velocity, pressure, and interface
  - can limit the order of accuracy
- assumes evolution in time at a fixed point in space is smooth
  - not true where interface changes

Gauge method: reformulate (2) & (3) to solve for a scalar field ϕ and an auxiliary vector field $\boldsymbol{m}$ whose divergence-free component is $\boldsymbol{u}$.

$$\begin{aligned} \rho\left(\boldsymbol{m}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) &= \mu\nabla^2\boldsymbol{m} + \boldsymbol{f}, && \text{in } \Omega \\ \boldsymbol{u} &= \boldsymbol{m} - \nabla\phi, && \text{in } \Omega \\ \nabla^2\phi &= \nabla\cdot\boldsymbol{m}, && \text{in } \Omega \end{aligned}$$

$\boldsymbol{u}$ is recovered at every instant in time from $\boldsymbol{m}$ and ϕ → weaker coupling!

Eulerian fluid-structure interaction
What about the fluid?

We'd like to maintain high-order accuracy when coupling to a solid. However, fractional stepping of $\boldsymbol{u}$...

- creates nonphysical coupling between velocity, pressure, and interface
  - can limit the order of accuracy
- assumes evolution in time at a fixed point in space is smooth
  - not true where interface changes

Gauge method: reformulate (2) & (3) to solve for a scalar field ϕ and an auxiliary vector field $\boldsymbol{m}$ whose divergence-free component is $\boldsymbol{u}$.

$$\begin{aligned} \rho\left(\boldsymbol{m}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) &= \mu\nabla^2\boldsymbol{m} + \boldsymbol{f}, && \text{in } \Omega \\ \boldsymbol{u} &= \boldsymbol{m} - \nabla\phi, && \text{in } \Omega \\ \nabla^2\phi &= \nabla\cdot\boldsymbol{m}, && \text{in } \Omega \end{aligned}$$

$\boldsymbol{u}$ is recovered at every instant in time from $\boldsymbol{m}$ and ϕ → weaker coupling!

Eulerian fluid-structure interaction
What about the fluid?

We'd like to maintain high-order accuracy when coupling to a solid. However, fractional stepping of $\boldsymbol{u}$...

- creates nonphysical coupling between velocity, pressure, and interface
  - can limit the order of accuracy
- assumes evolution in time at a fixed point in space is smooth
  - not true where interface changes

Gauge method: reformulate (2) & (3) to solve for a scalar field ϕ and an auxiliary vector field $\boldsymbol{m}$ whose divergence-free component is $\boldsymbol{u}$.

$$\begin{aligned} \rho\left(\boldsymbol{m}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) &= \mu\nabla^2\boldsymbol{m} + \boldsymbol{f}, && \text{in } \Omega \\ \boldsymbol{u} &= \boldsymbol{m} - \nabla\phi, && \text{in } \Omega \\ \nabla^2\phi &= \nabla\cdot\boldsymbol{m}, && \text{in } \Omega \end{aligned}$$

$\boldsymbol{u}$ is recovered at every instant in time from $\boldsymbol{m}$ and ϕ → weaker coupling!

Eulerian fluid-structure interaction
What about the fluid?

We'd like to maintain high-order accuracy when coupling to a solid. However, fractional stepping of $\boldsymbol{u}$...

- creates nonphysical coupling between velocity, pressure, and interface
  - can limit the order of accuracy
- assumes evolution in time at a fixed point in space is smooth
  - not true where interface changes

Gauge method: reformulate (2) & (3) to solve for a scalar field ϕ and an auxiliary vector field $\boldsymbol{m}$ whose divergence-free component is $\boldsymbol{u}$.

$$\begin{aligned} \rho\left(\boldsymbol{m}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) &= \mu\nabla^2\boldsymbol{m} + \boldsymbol{f}, && \text{in } \Omega \\ \boldsymbol{u} &= \boldsymbol{m} - \nabla\phi, && \text{in } \Omega \\ \nabla^2\phi &= \nabla\cdot\boldsymbol{m}, && \text{in } \Omega \end{aligned}$$

$\boldsymbol{u}$ is recovered at every instant in time from $\boldsymbol{m}$ and $\phi \rightarrow$ weaker coupling!

Where's the pressure?

If $\boldsymbol{m}$ and ϕ solve the gauge formulation, then $\boldsymbol{u}$ solves

$$\begin{aligned}(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}) &= -\nabla\left(\rho\phi_t - \mu\nabla^2\phi\right) + \mu\nabla^2\boldsymbol{u} + \boldsymbol{f}, && \text{in } \Omega\\ \nabla\cdot\boldsymbol{u} &= 0, && \text{in } \Omega\end{aligned}$$

So $\boldsymbol{u}$ solves Navier-Stokes with pressure identified (up to a constant) as

$$p = \rho\phi_t - \mu\nabla^2\phi.$$

Boundary conditions are imposed as

$$\begin{aligned}\boldsymbol{m}\cdot\boldsymbol{n} &= \boldsymbol{u}_D\cdot\boldsymbol{n}, && \text{on } \partial\Omega\\ \boldsymbol{m}\cdot\boldsymbol{\tau} &= \boldsymbol{u}_D\cdot\boldsymbol{\tau} + \nabla\phi\cdot\boldsymbol{\tau}, && \text{on } \partial\Omega\\ \nabla\phi\cdot\boldsymbol{n} &= 0, && \text{on } \partial\Omega.\end{aligned}$$

A multistep method is used to impose the boundary conditions on $\boldsymbol{m}$ and ϕ.

Where's the pressure?

If $\boldsymbol{m}$ and ϕ solve the gauge formulation, then $\boldsymbol{u}$ solves

$$\begin{aligned}(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}) &= -\nabla\left(\rho\phi_t - \mu\nabla^2\phi\right) + \mu\nabla^2\boldsymbol{u} + \boldsymbol{f}, && \text{in } \Omega\\ \nabla\cdot\boldsymbol{u} &= 0, && \text{in } \Omega\end{aligned}$$

So $\boldsymbol{u}$ solves Navier-Stokes with pressure identified (up to a constant) as

$$p = \rho\phi_t - \mu\nabla^2\phi.$$

Boundary conditions are imposed as

$$\begin{aligned}\boldsymbol{m}\cdot\boldsymbol{n} &= \boldsymbol{u}_D\cdot\boldsymbol{n}, && \text{on } \partial\Omega\\ \boldsymbol{m}\cdot\boldsymbol{\tau} &= \boldsymbol{u}_D\cdot\boldsymbol{\tau} + \nabla\phi\cdot\boldsymbol{\tau}, && \text{on } \partial\Omega\\ \nabla\phi\cdot\boldsymbol{n} &= 0, && \text{on } \partial\Omega.\end{aligned}$$

A multistep method is used to impose the boundary conditions on $\boldsymbol{m}$ and ϕ.

Both projection and gauge methods need a way to solve Poisson problems.

The DG framework naturally allows for this!

Discontinuous Galerkin for elliptic problems

Consider Poisson's equation on a domain Ω,

$$\begin{aligned} -\nabla^2 u &= f, \quad \text{in } \Omega \\ u &= 0, \quad \text{on } \partial\Omega \end{aligned}$$

We can rewrite this as a first-order system:

$$\begin{aligned} \boldsymbol{q} - \nabla u &= 0, \quad \text{in } \Omega \\ -\nabla \cdot \boldsymbol{q} &= f, \quad \text{in } \Omega \\ u &= 0, \quad \text{in } \partial\Omega \end{aligned}$$

A DG method aims to find functions u_h and $\boldsymbol{q}_h$ in some polynomial space which approximate the solutions u and $\boldsymbol{q}$ on each element.

Let K be an element in $\mathcal{T}_h$ and consider the Poisson system over K,

$$\begin{aligned} \boldsymbol{q} - \nabla u &= 0, \quad \text{in } K \\ -\nabla \cdot \boldsymbol{q} &= f, \quad \text{in } K \end{aligned}$$

To obtain the weak form, we multiply by test functions $(\boldsymbol{v}, w) \in \boldsymbol{V}_h^p \times W_h^p$ and integrate by parts to obtain:

$$\begin{aligned} (\boldsymbol{q}, \boldsymbol{v})_K + (u, \nabla \cdot \boldsymbol{v})_K - \langle u, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} &= 0 \\ (\boldsymbol{q}, \nabla w)_K - \langle \boldsymbol{q} \cdot \boldsymbol{n}, w \rangle_{\partial K} &= (f, w)_K \end{aligned}$$

We now make the following approximations:

- Replace $(\boldsymbol{q}, u)$ with $(\boldsymbol{q}_h, u_h) \in \boldsymbol{V}_h^p \times W_h^p$ in the bulk
- Replace $(\boldsymbol{q}, u)$ with $(\widehat{\boldsymbol{q}}_h, \widehat{u}_h)$ on the boundary
- Define $(\widehat{\boldsymbol{q}}_h, \widehat{u}_h)$ in terms of $(\boldsymbol{q}_h, u_h)$

Let K be an element in $\mathcal{T}_h$ and consider the Poisson system over K,

$$\begin{aligned} \boldsymbol{q} - \nabla u &= 0, \quad \text{in } K \\ -\nabla \cdot \boldsymbol{q} &= f, \quad \text{in } K \end{aligned}$$

To obtain the weak form, we multiply by test functions $(\boldsymbol{v}, w) \in \boldsymbol{V}_h^p \times W_h^p$ and integrate by parts to obtain:

$$\begin{aligned} (\boldsymbol{q}, \boldsymbol{v})_K + (u, \nabla \cdot \boldsymbol{v})_K - \langle u, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} &= 0 \\ (\boldsymbol{q}, \nabla w)_K - \langle \boldsymbol{q} \cdot \boldsymbol{n}, w \rangle_{\partial K} &= (f, w)_K \end{aligned}$$

We now make the following approximations:

- Replace $(\boldsymbol{q}, u)$ with $(\boldsymbol{q}_h, u_h) \in \boldsymbol{V}_h^p \times W_h^p$ in the bulk
- Replace $(\boldsymbol{q}, u)$ with $(\widehat{\boldsymbol{q}}_h, \widehat{u}_h)$ on the boundary
- Define $(\widehat{\boldsymbol{q}}_h, \widehat{u}_h)$ in terms of $(\boldsymbol{q}_h, u_h)$

Find $(\boldsymbol{q}_h, u_h)$ such that

$$(\boldsymbol{q}_h, \boldsymbol{v})_K + (u_h, \nabla \cdot \boldsymbol{v})_K - \langle \widehat{u}_h, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} = 0$$
$$(\boldsymbol{q}_h, \nabla w)_K - \langle \widehat{\boldsymbol{q}}_h \cdot \boldsymbol{n}, w \rangle_{\partial K} = (f, w)_K$$

for all $(\boldsymbol{v}, w)$ and for all $K \in \mathcal{T}_h$. Summing over all elements K yields

$$\boxed{\begin{aligned}(\boldsymbol{q}_h, \boldsymbol{v})_{\mathcal{T}_h} + (u_h, \nabla \cdot \boldsymbol{v})_{\mathcal{T}_h} - \langle \widehat{u}_h, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial \mathcal{T}_h} &= 0 \\ (\boldsymbol{q}_h, \nabla w)_{\mathcal{T}_h} - \langle \widehat{\boldsymbol{q}}_h \cdot \boldsymbol{n}, w \rangle_{\partial \mathcal{T}_h} &= (f, w)_{\mathcal{T}_h}\end{aligned}}$$

for all $(\boldsymbol{v}, w) \in \boldsymbol{V}_h^p \times W_h^p$.

It remains to determine what $(\widehat{q}_h, \widehat{u}_h)$ should be

Find $(\boldsymbol{q}_h, u_h)$ such that

$$(\boldsymbol{q}_h, \boldsymbol{v})_K + (u_h, \nabla \cdot \boldsymbol{v})_K - \langle \widehat{u}_h, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} = 0$$
$$(\boldsymbol{q}_h, \nabla w)_K - \langle \widehat{\boldsymbol{q}}_h \cdot \boldsymbol{n}, w \rangle_{\partial K} = (f, w)_K$$

for all $(\boldsymbol{v}, w)$ and for all $K \in \mathcal{T}_h$. Summing over all elements K yields

$$\boxed{\begin{aligned}(\boldsymbol{q}_h, \boldsymbol{v})_{\mathcal{T}_h} + (u_h, \nabla \cdot \boldsymbol{v})_{\mathcal{T}_h} - \langle \widehat{u}_h, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial \mathcal{T}_h} &= 0 \\ (\boldsymbol{q}_h, \nabla w)_{\mathcal{T}_h} - \langle \widehat{\boldsymbol{q}}_h \cdot \boldsymbol{n}, w \rangle_{\partial \mathcal{T}_h} &= (f, w)_{\mathcal{T}_h}\end{aligned}}$$

for all $(\boldsymbol{v}, w) \in \boldsymbol{V}_h^p \times W_h^p$.

It remains to determine what $(\widehat{\boldsymbol{q}}_h, \widehat{u}_h)$ should be

It can be shown that a good choice for the numerical flux is to "upwind" $\boldsymbol{q}$ and u in opposite directions:

$$\widehat{\boldsymbol{q}}_h = \{\boldsymbol{q}_h\} - c_{11}[u_h \boldsymbol{n}] + \boldsymbol{c}_{12}[\boldsymbol{q}_h \cdot \boldsymbol{n}]$$
$$\widehat{u}_h = \{u_h\} - \boldsymbol{c}_{12} \cdot [u_h \boldsymbol{n}] - c_{22}[\boldsymbol{q}_h \cdot \boldsymbol{n}]$$

on internal faces and

$$\widehat{\boldsymbol{q}}_h = \boldsymbol{q}_h - c_{11} u_h \boldsymbol{n}$$
$$\widehat{u}_h = 0$$

on boundary faces, where $c_{11}, c_{22} \geqslant 0$. Here $\{v\} = (v^+ + v^-)/2$ and $[v\boldsymbol{n}] = v^+ \boldsymbol{n}^+ + v^- \boldsymbol{n}^-$. The **local discontinuous Galerkin** method chooses

$$c_{11} = \mathcal{O}(1/h), \qquad \boldsymbol{c}_{12} = \boldsymbol{n}/2, \qquad c_{22} = 0.$$

With this choice, we can write the Poisson system as

$$a(\boldsymbol{q}_h, \boldsymbol{v}) + b(u_h, \boldsymbol{v}) = 0$$
$$-b^T(\boldsymbol{q_h}, w) + c(u_h, w) = \ell(w)$$

where

$$\begin{aligned} a(\boldsymbol{q}, \boldsymbol{v}) &= (\boldsymbol{q}, \boldsymbol{v})_{\mathcal{T}_h} \\ b(u, \boldsymbol{v}) &= (u, \nabla \cdot \boldsymbol{v})_{\mathcal{T}_h} - \langle \{u_h\} - \boldsymbol{c}_{12} \cdot [u_h \boldsymbol{n}], [\boldsymbol{v} \cdot \boldsymbol{n}] \rangle_{\mathcal{E}_h} \\ b^T(\boldsymbol{q}, w) &= -(\boldsymbol{q}_h, \nabla w)_{\mathcal{E}_h} + \langle \{\boldsymbol{q}_h\} + \boldsymbol{c}_{12}[\boldsymbol{q}_h \cdot \boldsymbol{n}], [w \boldsymbol{n}] \rangle_{\mathcal{E}_h} \\ c(u, w) &= (c_{11}[u\boldsymbol{n}], [w\boldsymbol{n}])_{\mathcal{E}_h} \\ \ell(w) &= (f, w)_{\mathcal{T}_h} \end{aligned}$$

Or in matrix form as

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

The LDG choice makes A block-diagonal, and thus easy to invert. We can therefore eliminate Q by taking a Schur complement to obtain the reduced system

$$KU = L$$

where $K = C + B^T A^{-1} B$.

Upcoming work: Can **multigrid** be effectively applied to solve this system?

Idea: Coarsen Q and U systems separately

Or in matrix form as

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

The LDG choice makes A block-diagonal, and thus easy to invert. We can therefore eliminate Q by taking a Schur complement to obtain the reduced system

$$KU = L$$

where $K = C + B^T A^{-1} B$.

Upcoming work: Can **multigrid** be effectively applied to solve this system?

Idea: Coarsen Q and U systems separately

Or in matrix form as

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

The LDG choice makes A block-diagonal, and thus easy to invert. We can therefore eliminate Q by taking a Schur complement to obtain the reduced system

$$KU = L$$

where $K = C + B^T A^{-1} B$.

Upcoming work: Can **multigrid** be effectively applied to solve this system?

Idea: Coarsen Q and U systems separately

Goals and future work
(aka my PhD)

- Create an optimal complexity spectral element method
- Integrate Eulerian fluid-structure interaction into the implicit mesh DG framework, in 2D and 3D
- Develop an effective way to apply multigrid to DG

Goals and future work
(aka my PhD)

- Create an optimal complexity spectral element method
- Integrate Eulerian fluid-structure interaction into the implicit mesh DG framework, in 2D and 3D
- Develop an effective way to apply multigrid to DG

Goals and future work
(aka my PhD)

- Create an optimal complexity spectral element method
- Integrate Eulerian fluid-structure interaction into the implicit mesh DG framework, in 2D and 3D
- Develop an effective way to apply multigrid to DG

Thank you

Thanks for listening!

Thanks to: Chris Rycroft, Alex Townsend, Robert Saye, Jaime Peraire, Sheehan Olver, Heather Wilber, & Haixiang Liu.

Thomas algorithm

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

First compute

$$c_i' = \begin{cases} \frac{c_i}{b_i} & i = 1 \\ \frac{c_i}{b_i - a_i c_{i-1}'} & i = 2, \ldots, n-1 \end{cases} \qquad d_i' = \begin{cases} \frac{d_i}{b_i} & i = 1 \\ \frac{d_i - a_i d_{i-1}'}{b_i - a_i c_{i-1}'} & i = 2, \ldots, n \end{cases}$$

Then compute x by backsubstitution:

$$\begin{aligned} x_n &= d_n' \\ x_i &= d_i' - c_i' x_{i+1}, \quad i = n-1, \ldots, 1. \end{aligned}$$