Divide and...

Piotr Krzyżanowski

MIM Colloquium 2017/11/9

MIM UW, Institute of Applied Mathematics

System of linear equations

Problem

Given nonsingular $L \in \mathbb{R}^{N \times N}$ *and* $b \in \mathbb{R}^N$, *find* $x \in \mathbb{R}^N$ *satisfying*

$$\begin{cases} L_{11}x_1 + L_{12}x_2 + \ldots L_{1N}x_N & = b_1, \\ L_{21}x_1 + L_{22}x_2 + \ldots L_{2N}x_N & = b_2, \\ \quad \vdots & \\ L_{N1}x_1 + L_{N2}x_2 + \ldots L_{NN}x_N & = b_N, \end{cases}$$

System of linear equations

Problem

Given nonsingular $L \in \mathbb{R}^{N \times N}$ *and* $b \in \mathbb{R}^N$, *find* $x \in \mathbb{R}^N$ *satisfying*

$$\begin{cases} L_{11}x_1 + L_{12}x_2 + \ldots L_{1N}x_N & = b_1, \\ L_{21}x_1 + L_{22}x_2 + \ldots L_{2N}x_N & = b_2, \\ \quad \vdots & \\ L_{N1}x_1 + L_{N2}x_2 + \ldots L_{NN}x_N & = b_N, \end{cases}$$

Can anything be more boring?

A solved problem

Find $x \in \mathbb{R}^N$ such that

$$Lx = b$$

- Gaussian elimination known for about 2000 years; costs $O(N^3)$

A solved problem

Find $x \in \mathbb{R}^N$ such that

$$Lx = b$$

- Gaussian elimination known for about 2000 years; costs $O(N^3)$
- Cramer's rule (much) more costly: $O(N!)$

A solved problem?

Find $x \in \mathbb{R}^N$ such that

$$Lx = b$$

- Gaussian elimination known for about 2000 years; costs $O(N^3)$
- Cramer's rule (much) more costly: $O(N!)$
- Complexity: still an open question

A solved problem?

Find $x \in \mathbb{R}^N$ such that

$$Lx = b$$

- Gaussian elimination known for about 2000 years; costs $O(N^3)$
- Cramer's rule (much) more costly: $O(N!)$
- Complexity: still an open question
  - We know $O(N^\omega)$ algorithms exist with $\omega < 3$.

Strassen's matrix multiply

- Matrix–matrix multiplication $X = L \cdot B$ as complex as solving $Lx = b$

Strassen's matrix multiply

- Matrix–matrix multiplication $X = L \cdot B$ as complex as solving $Lx = b$
- **Divide and...**

$$\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Strassen's matrix multiply

- **Divide and...**

$$\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

  Naively,

$$\begin{aligned} X_{11} &= L_{11}B_{11} + L_{12}B_{21} \\ X_{12} &= L_{11}B_{12} + L_{12}B_{22} \\ X_{21} &= L_{21}B_{11} + L_{22}B_{21} \\ X_{22} &= L_{21}B_{12} + L_{22}B_{22}. \end{aligned}$$

  gives a recursive "divide-and-conquer" algorithm.

- Complexity: still $O(N^3)$.

Strassen's matrix multiply

- **Divide and... think again**:

$$\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Reduce number of matrix multiplications to seven!

$$X_{11} = P_1 + P_4 - P_5 + P_7$$
$$X_{12} = P_3 + P_5$$
$$X_{21} = P_2 + P_4$$
$$X_{22} = P_1 + P_3 - P_2 + P_6,$$

$$P_1 = (L_{11} + L_{22})(B_{11} + B_{22}),$$
$$P_2 = (L_{21} + L_{22})B_{11}$$
$$\vdots$$
$$P_7 = (L_{12} - L_{22})(B_{21} + B_{22})$$

Complexity: $O(N^{\log_2 7}) \approx O(N^{2.808\ldots})$

Strassen (1969) *Numer. Math.*

Large systems are intractable for simple Gaussian elimination

Find $x \in \mathbb{R}^N$ such that

$$Lx = b.$$

Large systems are intractable for simple Gaussian elimination

Find $x \in \mathbb{R}^N$ such that

$$Lx = b.$$

If $N = 10^6$, a PC would have

- computed the solution after 10^9 seconds

if straightforward Gaussian elimination (e.g. LAPACK's `DGESV`) was used.

Large systems are intractable for simple Gaussian elimination

Find $x \in \mathbb{R}^N$ such that

$$Lx = b.$$

If $N = 10^6$, a PC would have

- computed the solution after 10^9 seconds i.e. $\approx$ **32 years**

if straightforward Gaussian elimination (e.g. LAPACK's `DGESV`) was used.

Large systems are intractable for simple Gaussian elimination

Find $x \in \mathbb{R}^N$ such that

$$Lx = b.$$

If $N = 10^6$, a PC would have

- computed the solution after 10^9 seconds i.e. $\approx$ **32 years**
- needed 10^{13} bytes of memory

if straightforward Gaussian elimination (e.g. LAPACK's `DGESV`) was used.

Large systems are intractable for simple Gaussian elimination

Find $x \in \mathbb{R}^N$ such that

$$Lx = b.$$

If $N = 10^6$, a PC would have

- computed the solution after 10^9 seconds i.e. $\approx$ **32 years**
- needed 10^{13} bytes of memory i.e. $\approx$ **9,000 GB**

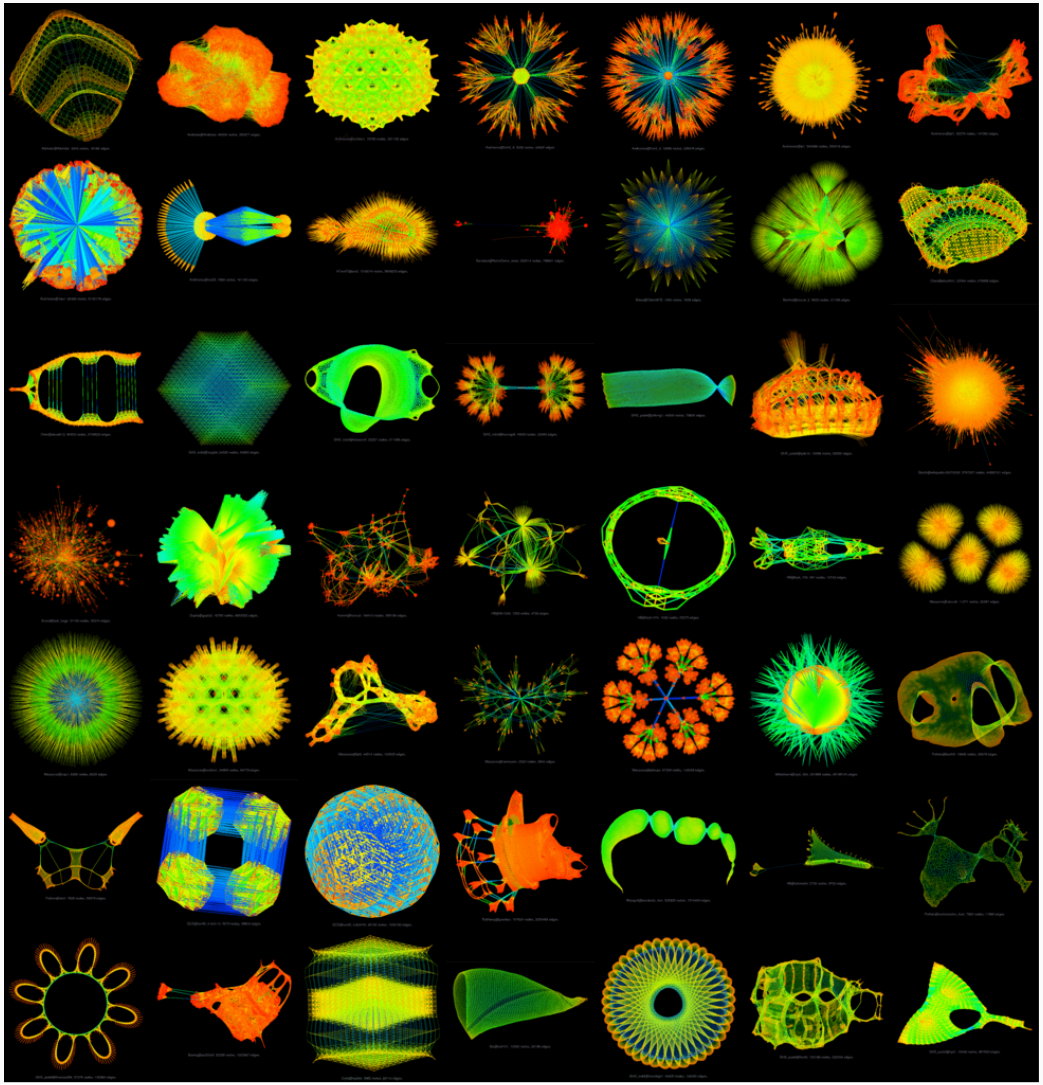
if straightforward Gaussian elimination (e.g. LAPACK's `DGESV`) was used.

Outline

1. Large systems of linear equations: where do they come from?
2. Systems with (lots of) structure: finite elements for PDEs
3. Solving large sparse systems
4. A sidenote: another class of structured sparse matrices
5. Domain decomposition for PDEs
6. Splitting equations
7. Summing up

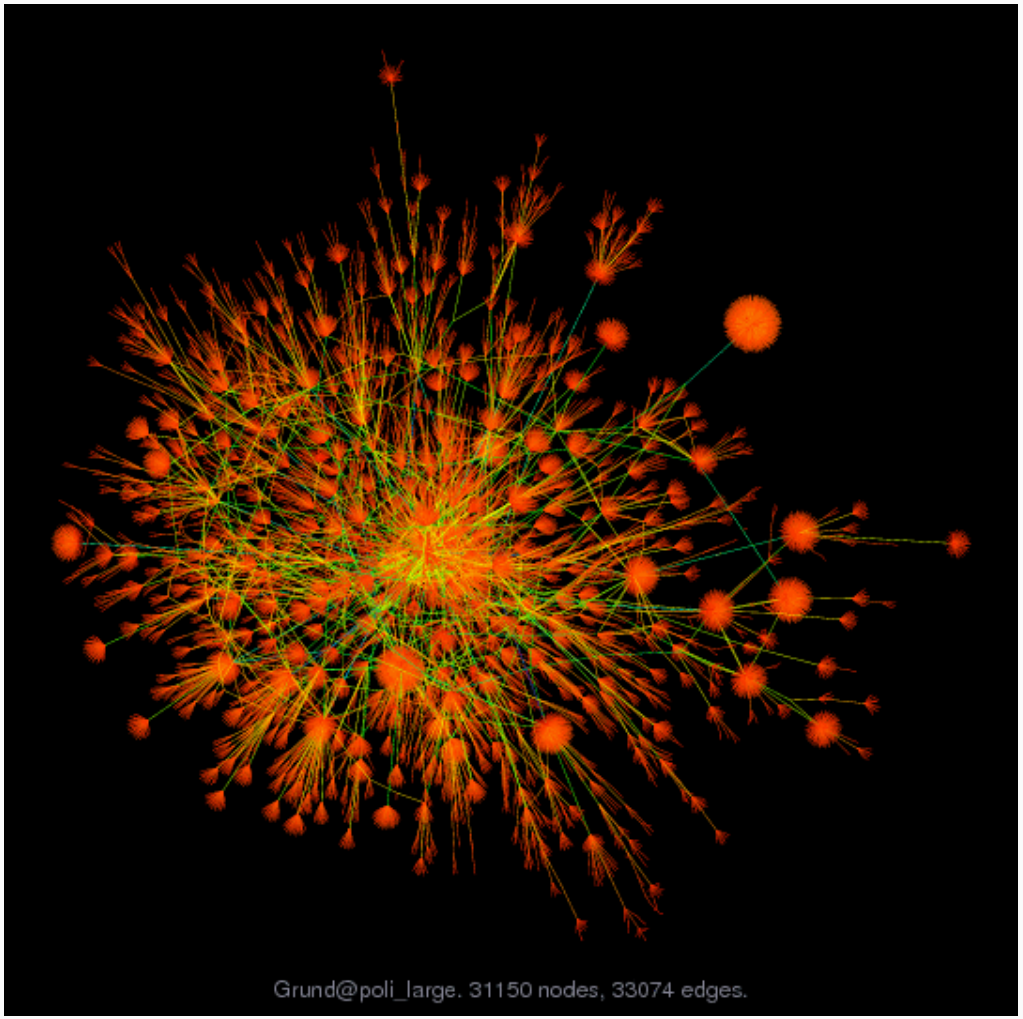
Large systems of linear equations: where do they come from?

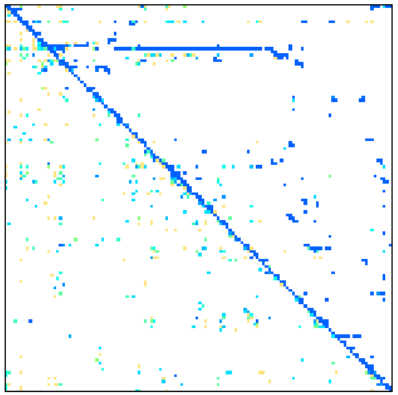
The beauty of sparse matrices



Davis, Hu (2011) *ACM Trans. Math. Softw.*

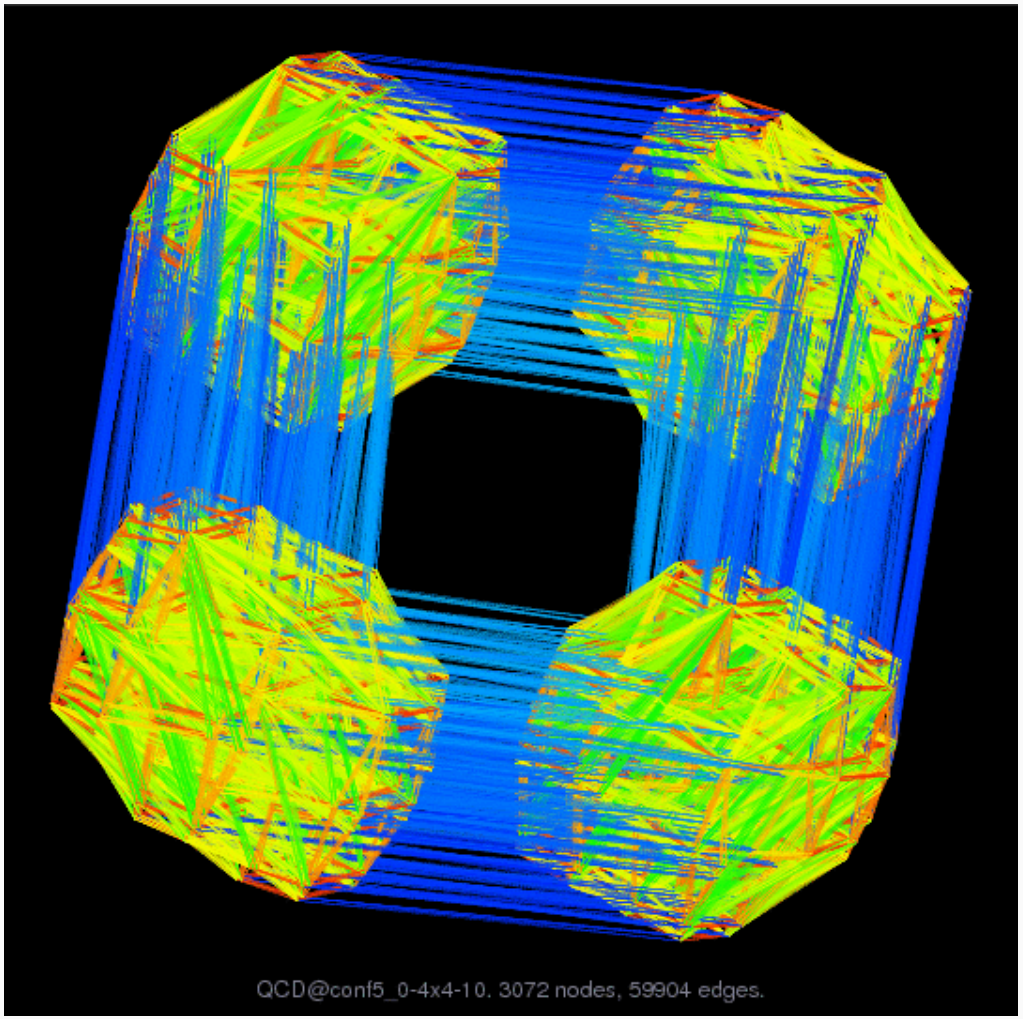
Economic problem

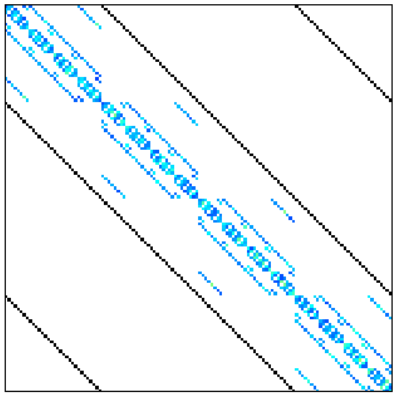




$N = 15,575$

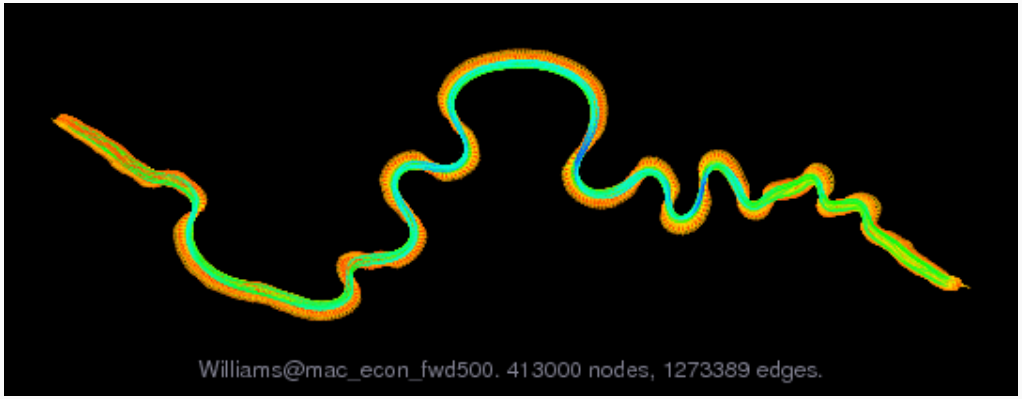
Quantum chromodynamics

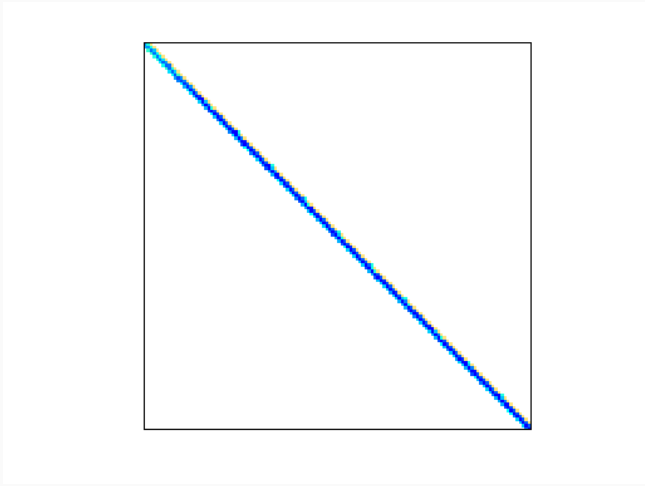




$N = 3,072$

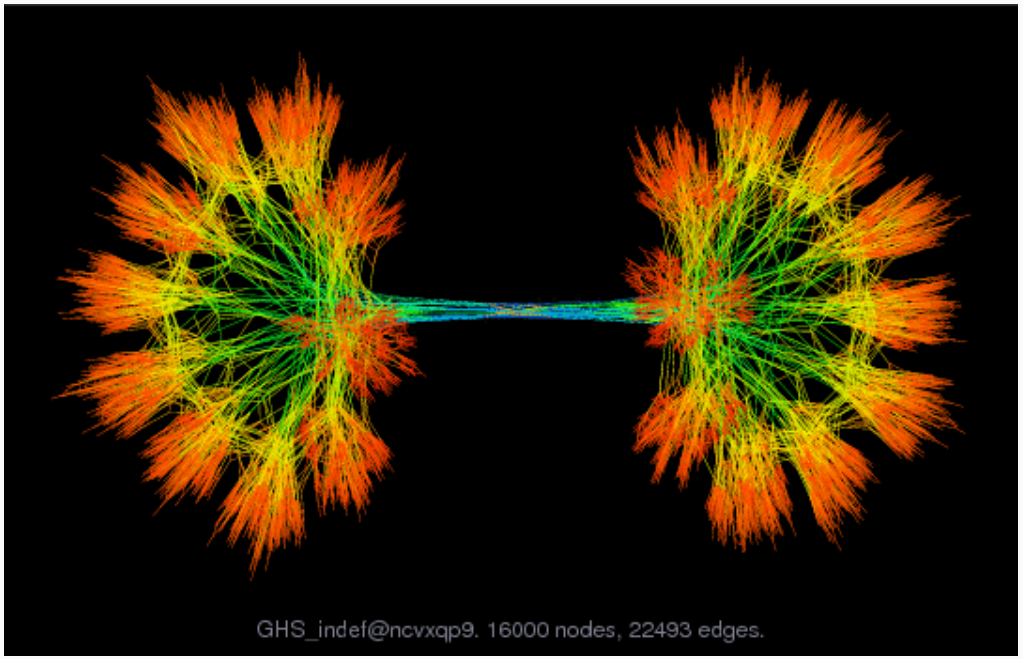
Macroeconomic problem

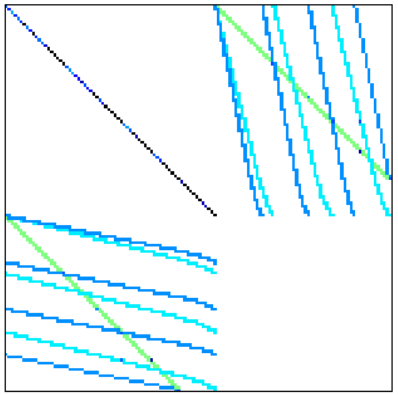




$N = 206,500$

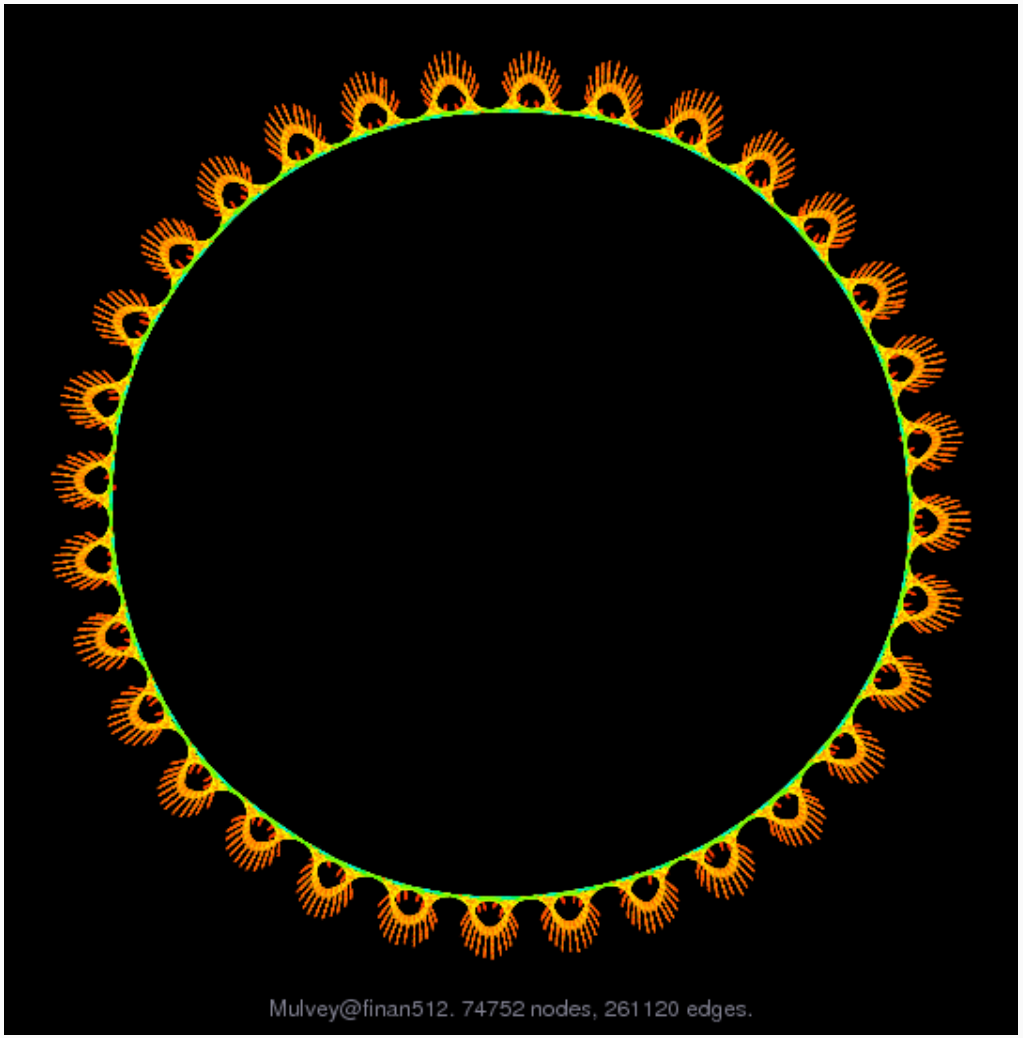
KKT system, nonconvex optimization

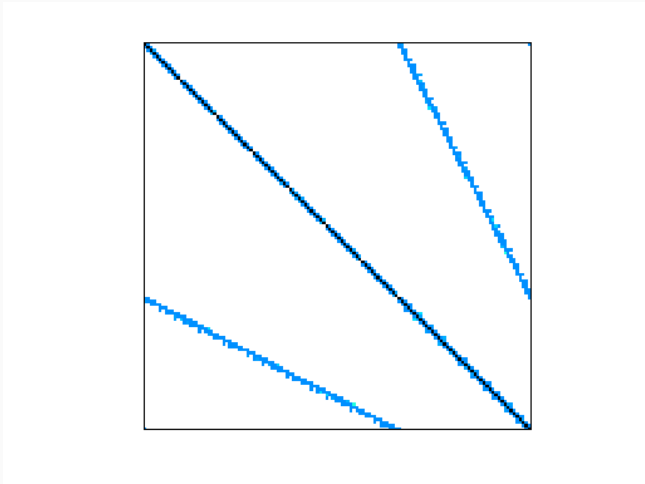




$N = 16,554$

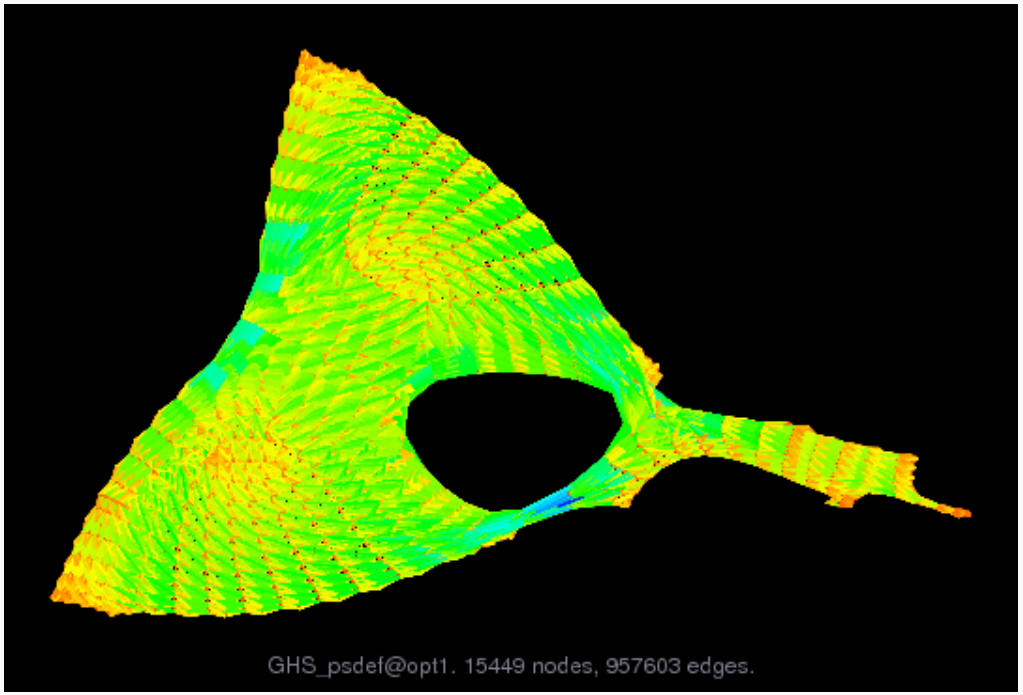
Financial portfolio optimization

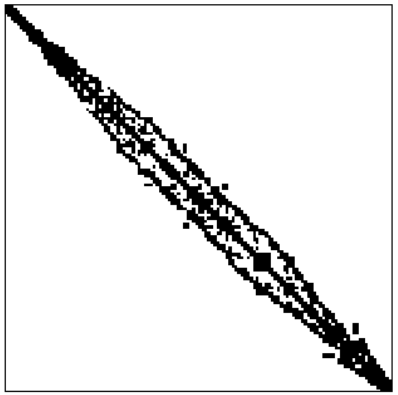




$N = 74,752$

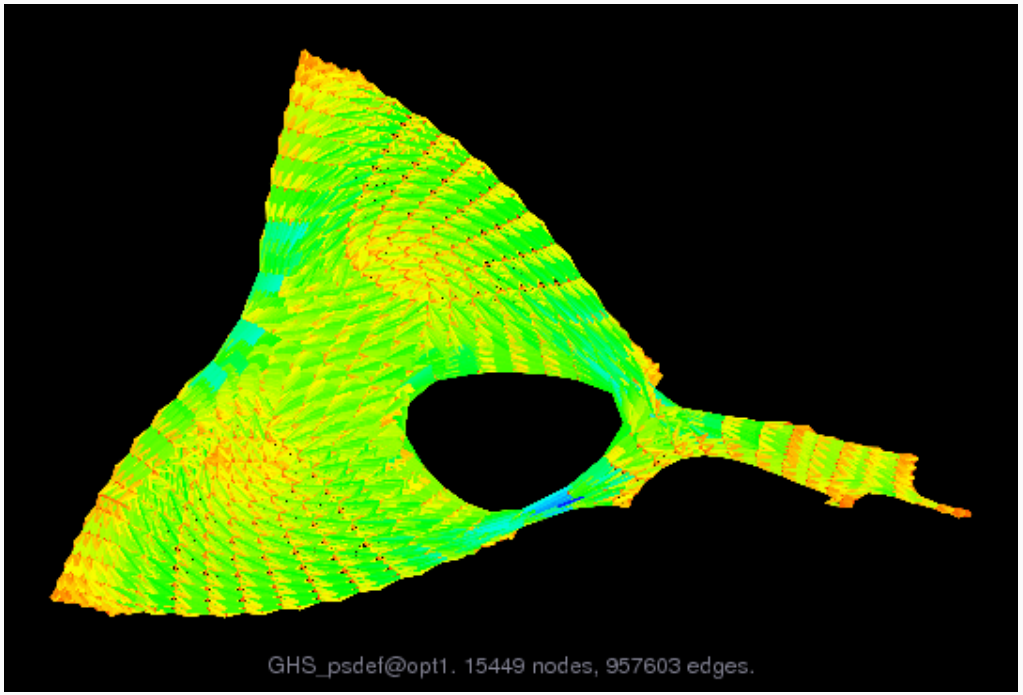
Structural engineering, finite element

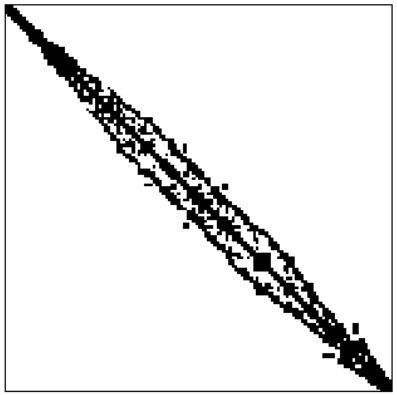




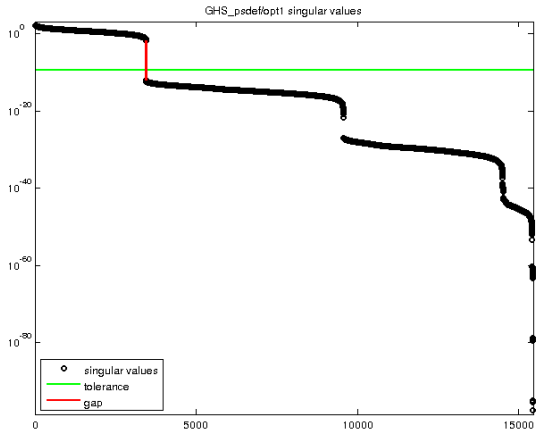
$N = 15,449$

Structural engineering, finite element





$N = 15,449$



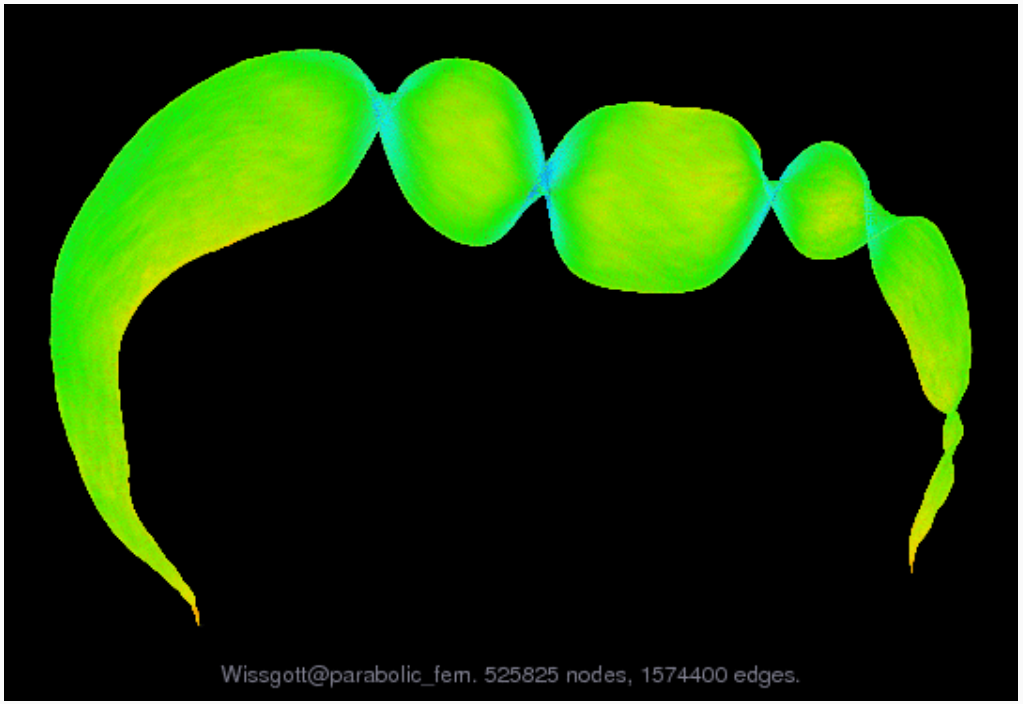
Structural engineering, finite element

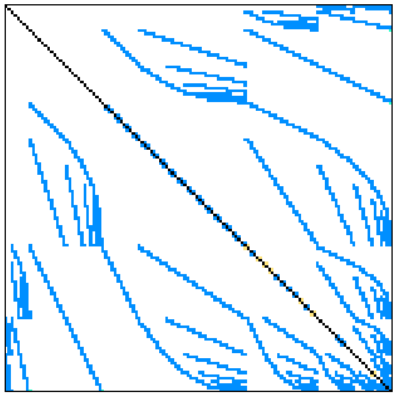
This is how a sparse matrix *really* looks like:

```
        63929           444         10024          53461
rsa                    4875          4875         160383             0
(11I7)          (16I5)          (3E23.15)
1     25     48     70    103    135    166    199    231    262    295
327    358    391    423    454    487    519    550    583    615    646
679    711    742    775    807    838    871    903    934    967    999
1030   1063   1095   1126   1147   1167   1186   1216   1245   1273   1315
1356   1396   1438   1479   1519   1561   1602   1642   1684   1725   1765
1807   1848   1888   1930   1971   2011   2053   2094   2134   2176   2217
2257   2299   2340   2380   2422   2463   2503   2545   2586   2626   2653
2679   2704   2734   2763   2791   2833   2874   2914   2956   2997   3037

 0.409672687144694E-14 -0.270324344694379E-11  0.462322806286147E-14
-0.125103474334186E-15 -0.157968969372661E-11  0.120545535566847E-14
-0.415025707341799E-14 -0.518149596242225E-12  0.850962616131678E-13
-0.209551074847814E-12 -0.107421047460559E-11  0.340867474174134E-12
-0.371312672815900E-13 -0.562016116896941E-12  0.811101830369373E-13
0.198413464034039E-12 -0.375360224439469E-12  0.141281556760829E-13
-0.805393690795621E-12  0.267683729453480E-11  0.861749802483021E-16
-0.113997243532461E-15 -0.423020243608237E-12 -0.143670337428612E-13
0.830097314588988E-12 -0.580299367504821E-12 -0.811121342725180E-13
-0.198394255770204E-12 -0.107673095703232E-11 -0.340570288494983E-12
0.124288644083145E-13 -0.501209266056193E-12 -0.852387929702163E-13
0.209644642907451E-12  0.454179546572026E-10  0.256700564221894E-11
0.248365328762197E-16  0.885615735926505E-11  0.482064121963914E-13
0.109981819279385E-14 -0.598541997591639E-11 -0.124576307872633E-11
0.463130261987941E-14 -0.198244879116781E-10 -0.708476661679811E-12
```

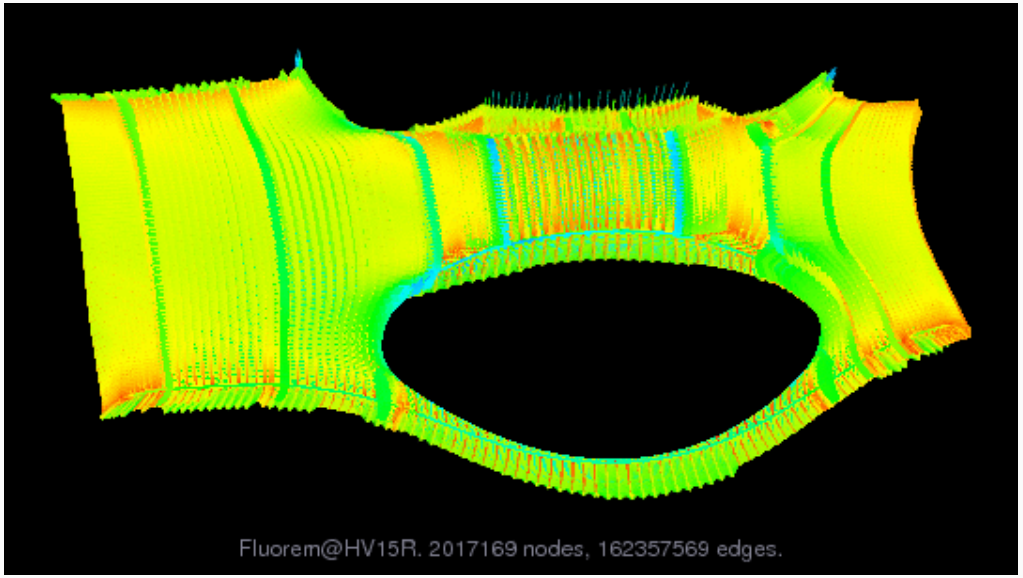
Parabolic diffusion-convection-reaction, finite element

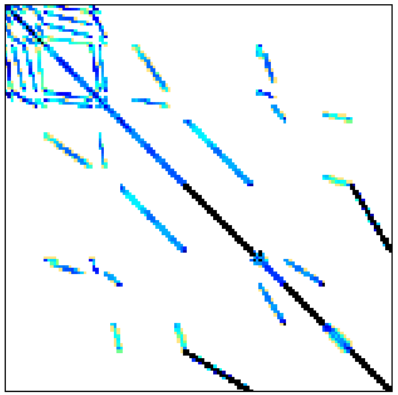




$N = 525,825$

Fluid dynamics, finite element





$N = 2,017,169$

Systems with (lots of) structure: finite elements for PDEs

A model PDE: diffusion equation

Find $u : \mathbb{R}^d \supset \Omega \to R$ satisfying

$$-\operatorname{div}(\,\rho(x)\nabla u(x)\,) = f(x) \quad \forall x \in \Omega,$$
$$u(x) = 0 \quad \forall x \in \partial\Omega.$$

For example: u — temperature, ρ — thermal conductivity, f — external heating

A model PDE: diffusion equation

Find $u : \mathbb{R}^d \supset \Omega \to R$ satisfying

$$-\operatorname{div}(\,\rho(x)\nabla u(x)\,) = f(x) \quad \forall x \in \Omega,$$
$$u(x) = 0 \quad \forall x \in \partial\Omega.$$

For example: u — temperature, ρ — thermal conductivity, f — external heating

Assume $\rho(x) = 1$.

A model PDE: diffusion equation

Find $u : \mathbb{R}^d \supset \Omega \to R$ satisfying

$$\begin{aligned} -\Delta u(x) &= f(x) \quad \forall x \in \Omega, \\ u(x) &= 0 \quad \forall x \in \partial\Omega. \end{aligned}$$

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Finite element approximation

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Finite element approximation

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Problem (discrete)

Find $u_h \in V_h \subset H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u_h \cdot \nabla v_h \, dx = \int_\Omega f \, v_h \, dx \qquad \forall v_h \in V_h.$$

Here V_h is finite dimensional. How to choose it?

Finite element approximation

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Problem (discrete)

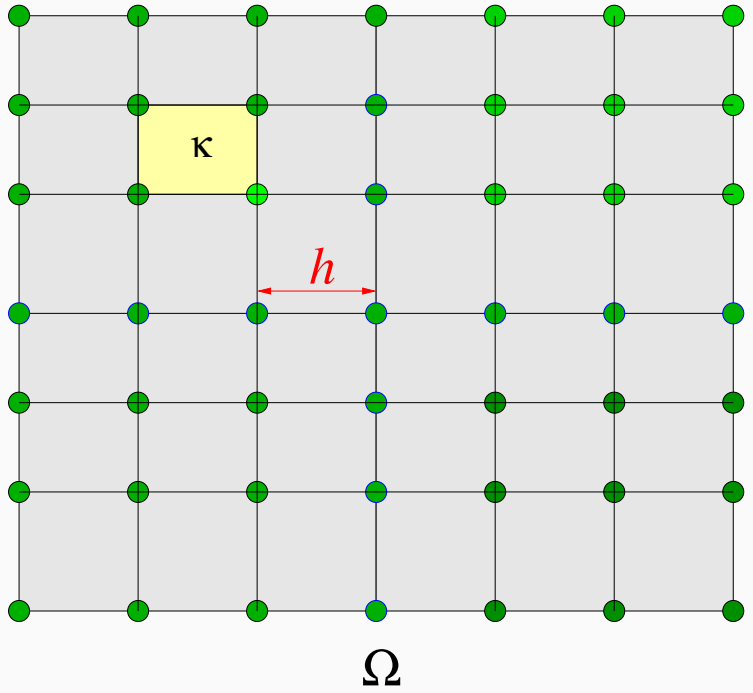
Find $u_h \in V_h \subset H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u_h \cdot \nabla v_h \, dx = \int_\Omega f \, v_h \, dx \qquad \forall v_h \in V_h.$$

Here V_h is finite dimensional. How to choose it?

Divide and... approximate wisely.

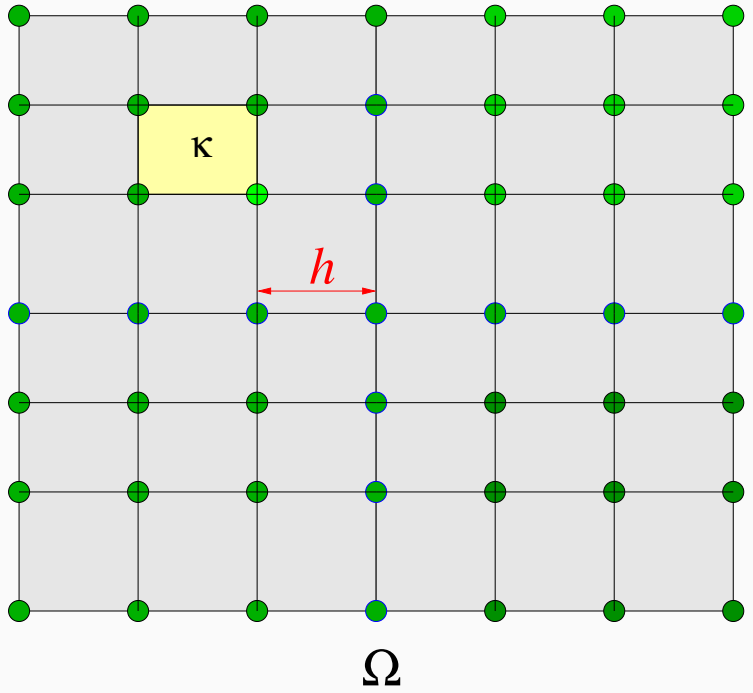
Finite elements



Divide Ω into smaller elements:

- Triangulation $\mathcal{T}_h$ consisting of elements κ.

Finite elements

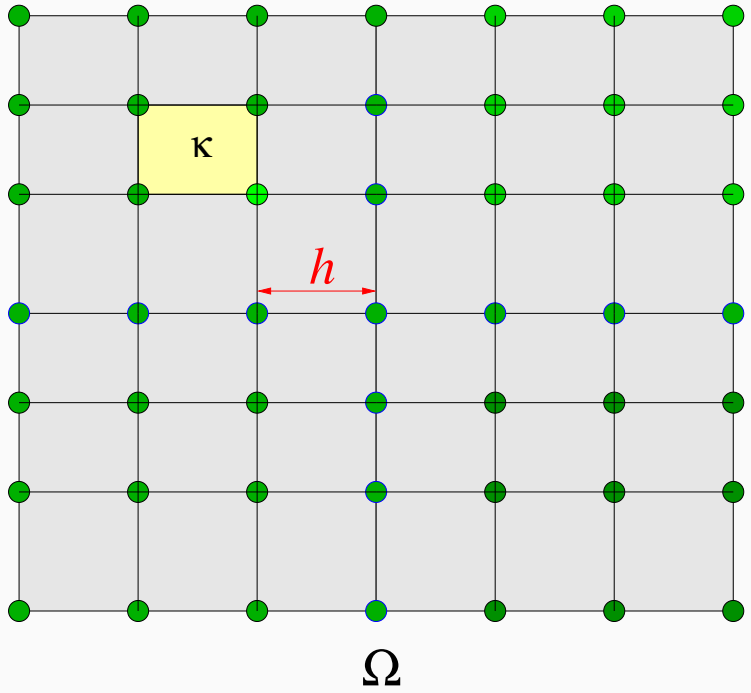


Divide Ω into smaller elements:

- Triangulation $\mathcal{T}_h$ consisting of elements κ.

$$V_h = \{v \in C(\Omega) \cap H_0^1(\Omega) : v_{|_\kappa} \in P_1(\kappa) \quad \forall \kappa \in \mathcal{T}_h\} \subset H_0^1(\Omega)$$

Finite elements



- Triangulation $\mathcal{T}_h$ consisting of elements κ.

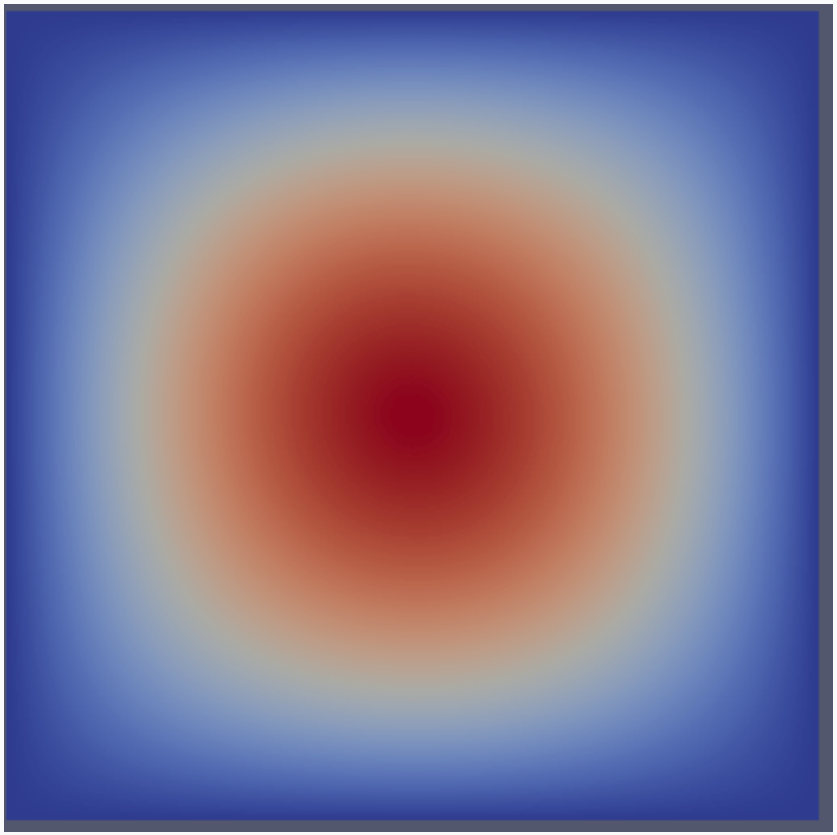
$$V_h = \{v \in C(\Omega) \cap H_0^1(\Omega) : v_{|_\kappa} \in P_1(\kappa) \quad \forall \kappa \in \mathcal{T}_h\} \subset H_0^1(\Omega)$$

More generally,

$$V_h^p = \{v \in C(\Omega) \cap H_0^1(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

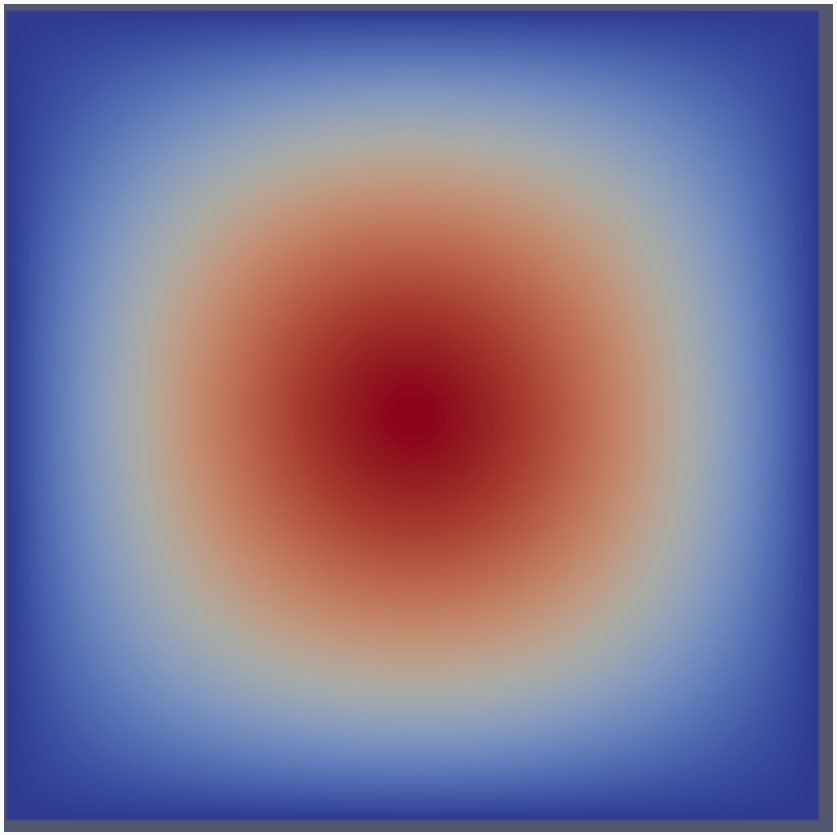
Experiment: h–approximation vs p–approximation

Consider true solution to $-\Delta u = f$:



Experiment: h–approximation vs p–approximation

Consider true solution to $-\Delta u = f$:

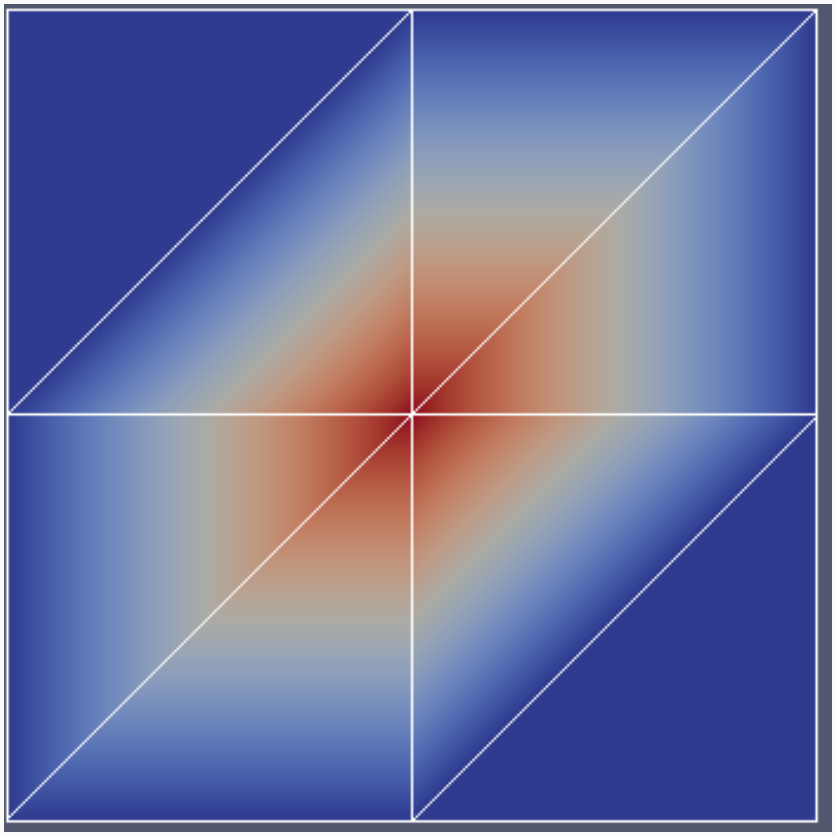


How well can it be approximated by the finite element method?

Finite element h–approximation vs p–approximation

fixed $p = 1$

decrease h

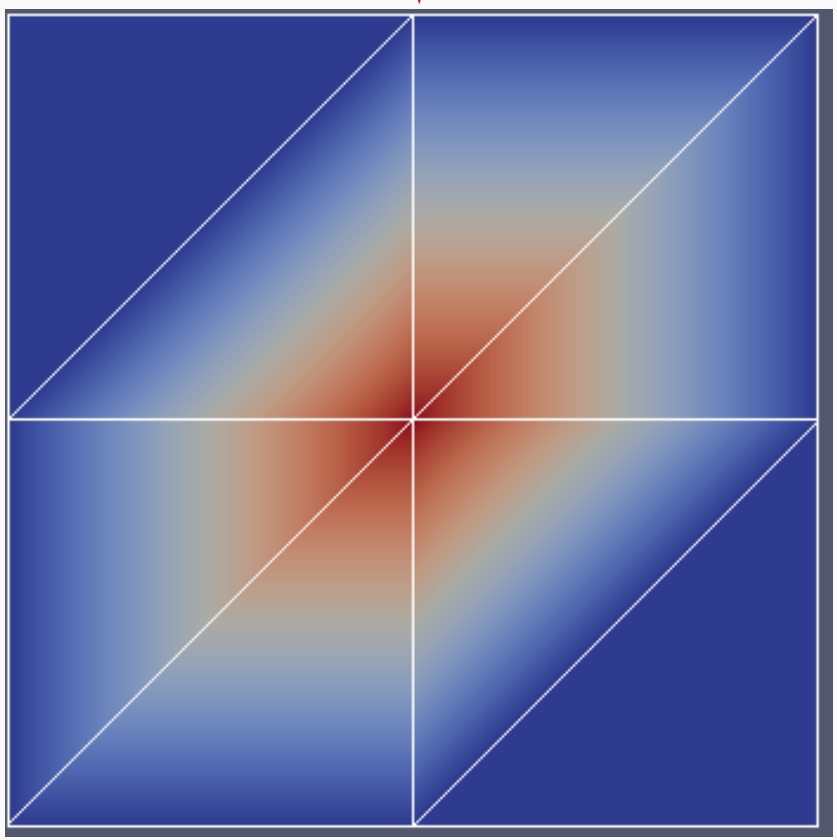


$h = 1/2$

N=9

fixed $h = 1/2$

increase p

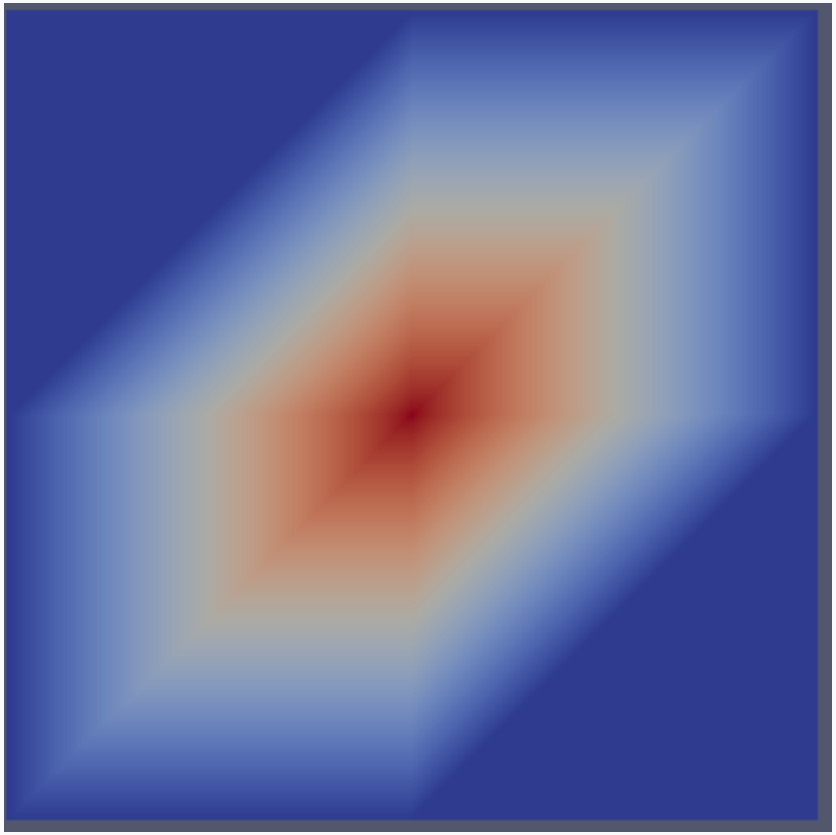


$p = 1$

N=9

h–approximation vs p–approximation

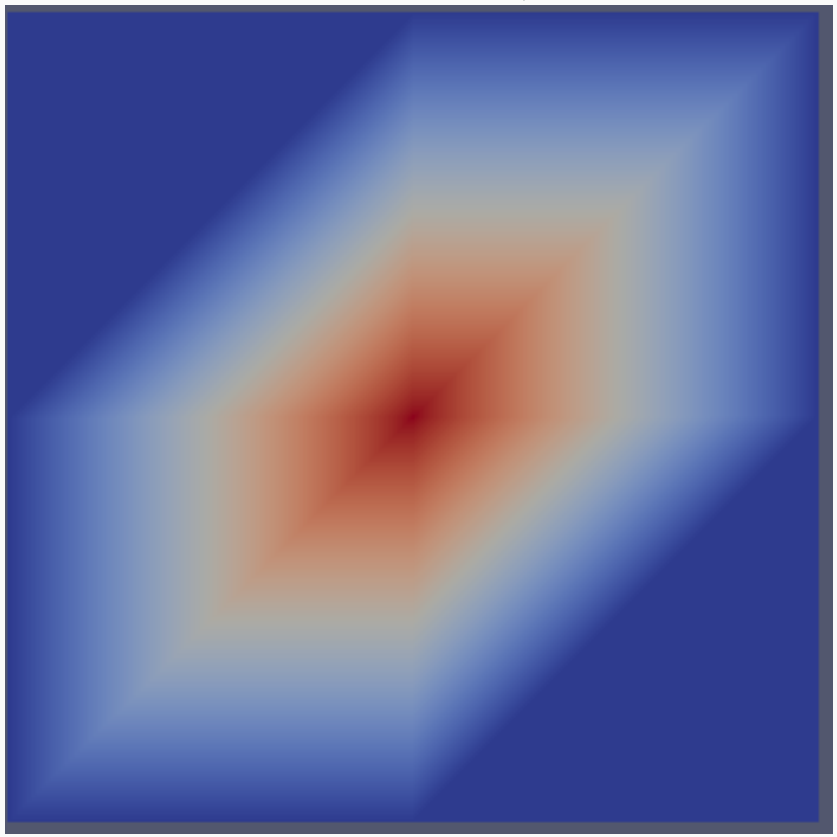
fixed $p = 1$



$h = 1/2$

N=9

fixed $h = 1/2$

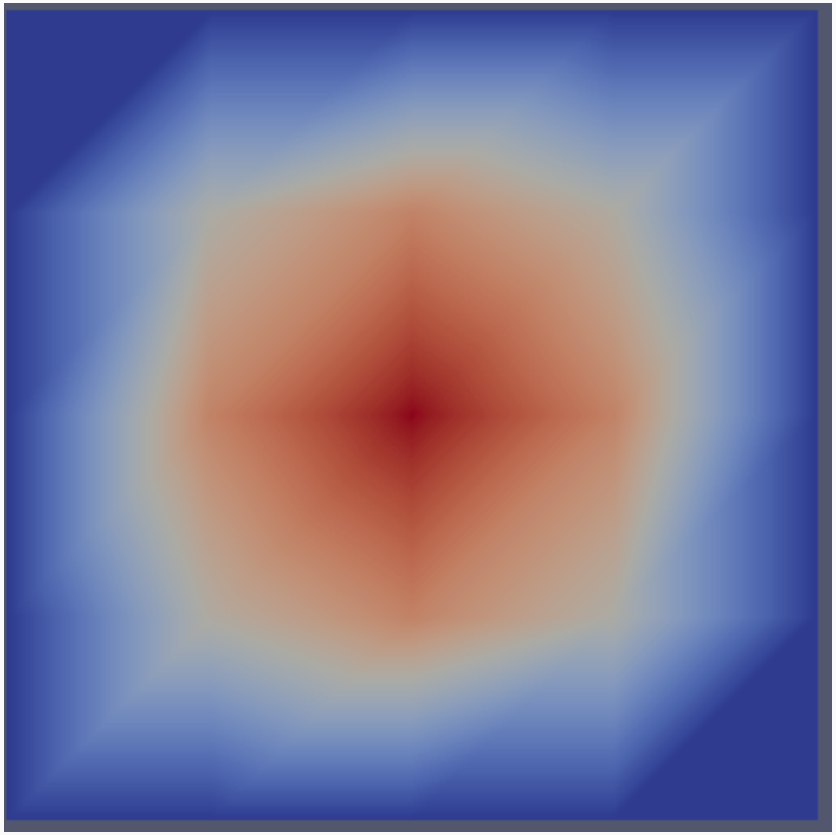


$p = 1$

N=9

h–approximation vs p–approximation

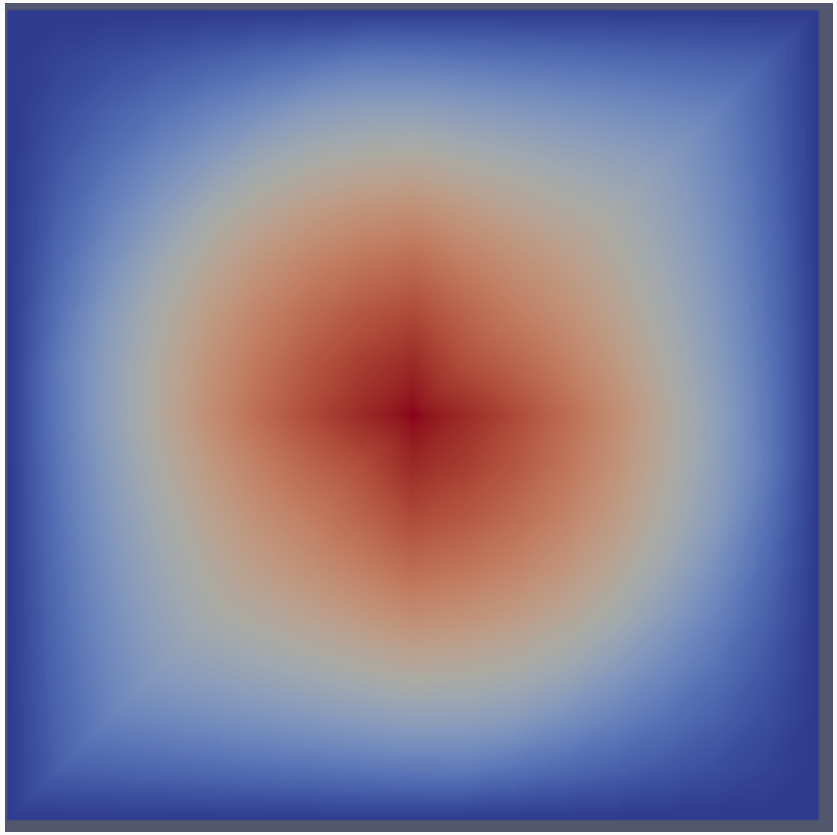
fixed $p = 1$



$h = 1/2^{\mathbf{2}}$

N=25

fixed $h = 1/2$

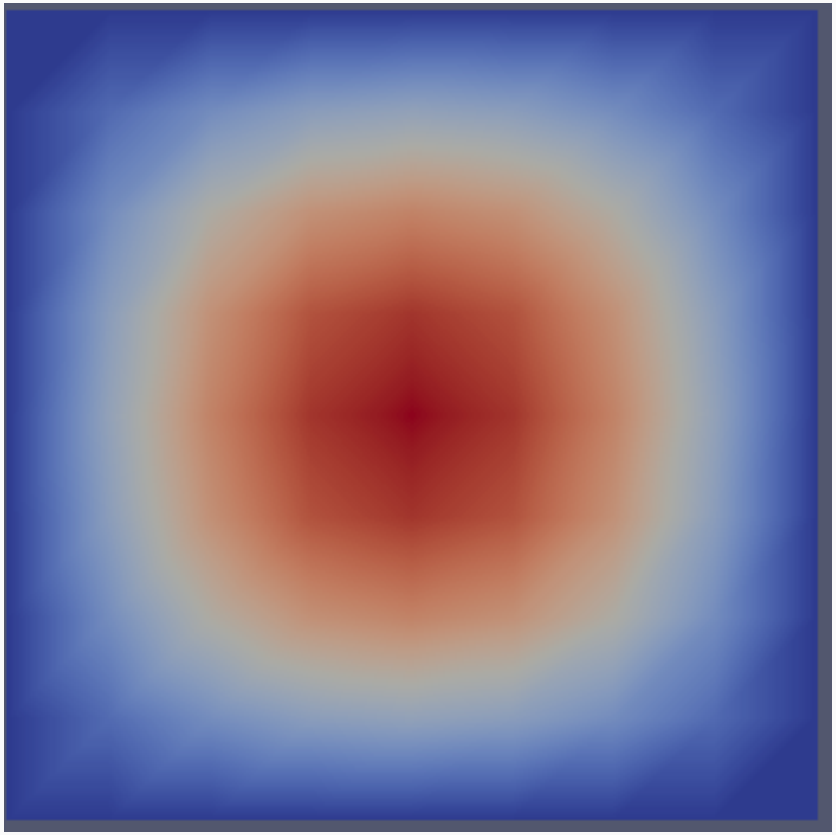


$p = \mathbf{2}$

N=25

h–approximation vs p–approximation

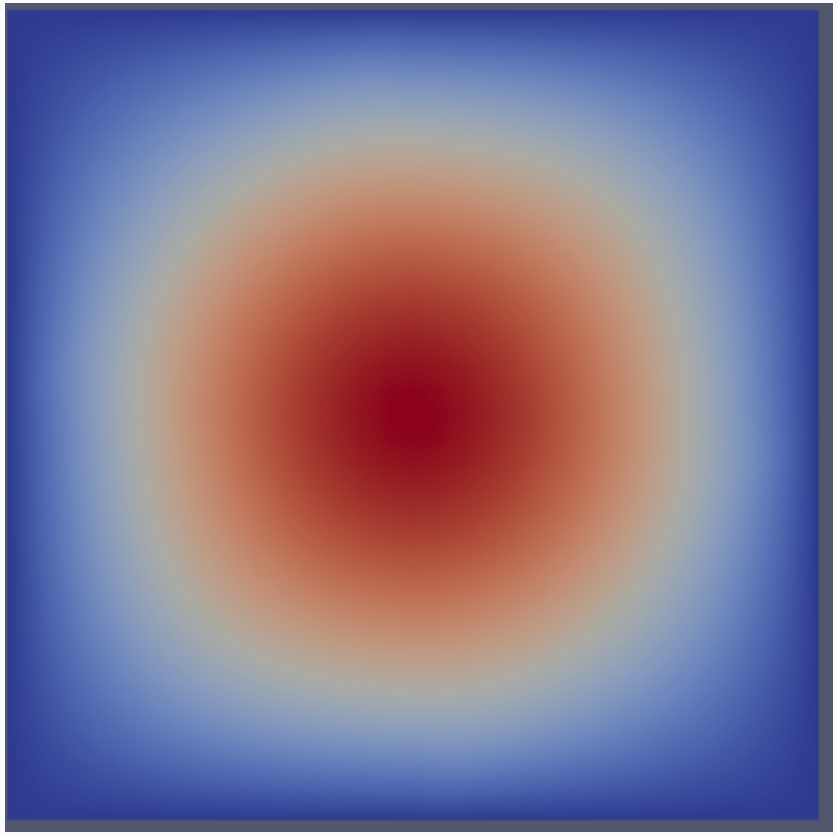
fixed $p = 1$



$h = 1/2^{\mathbf{3}}$

N=81

fixed $h = 1/2$

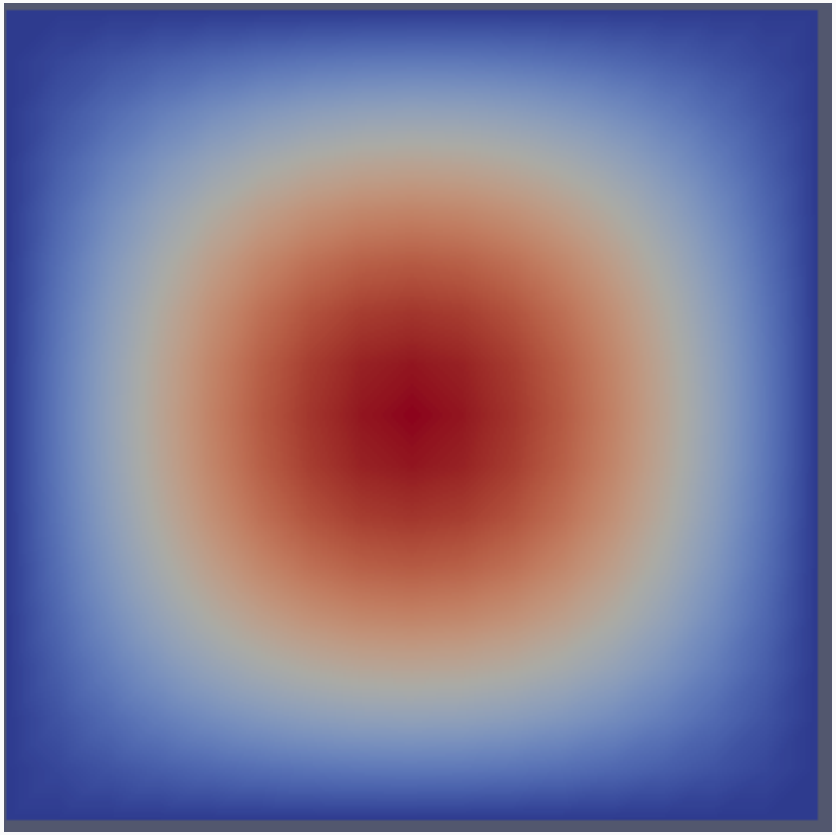


$p = \mathbf{3}$

N=49

h–approximation vs p–approximation

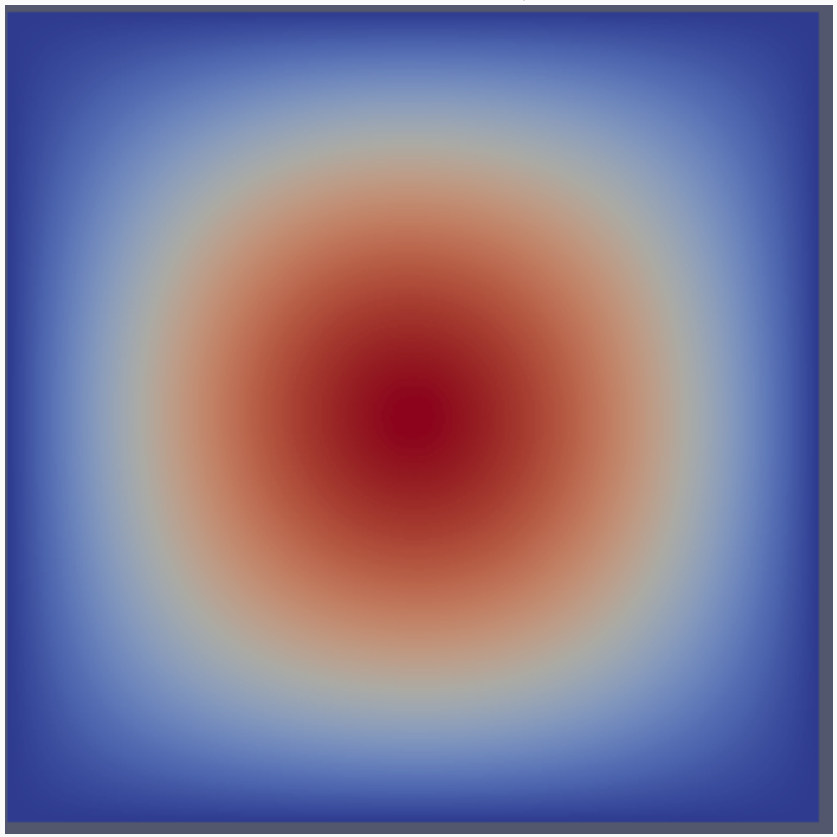
fixed $p = 1$



$h = 1/2^{4}$

N=289

fixed $h = 1/2$

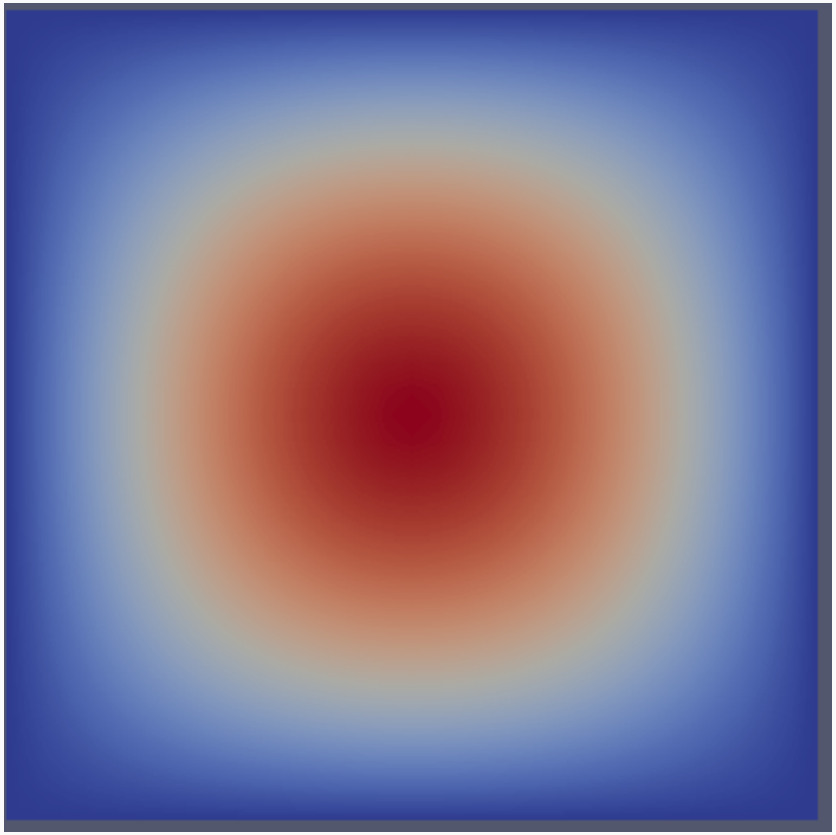


$p = 4$

N=81

h–approximation vs p–approximation

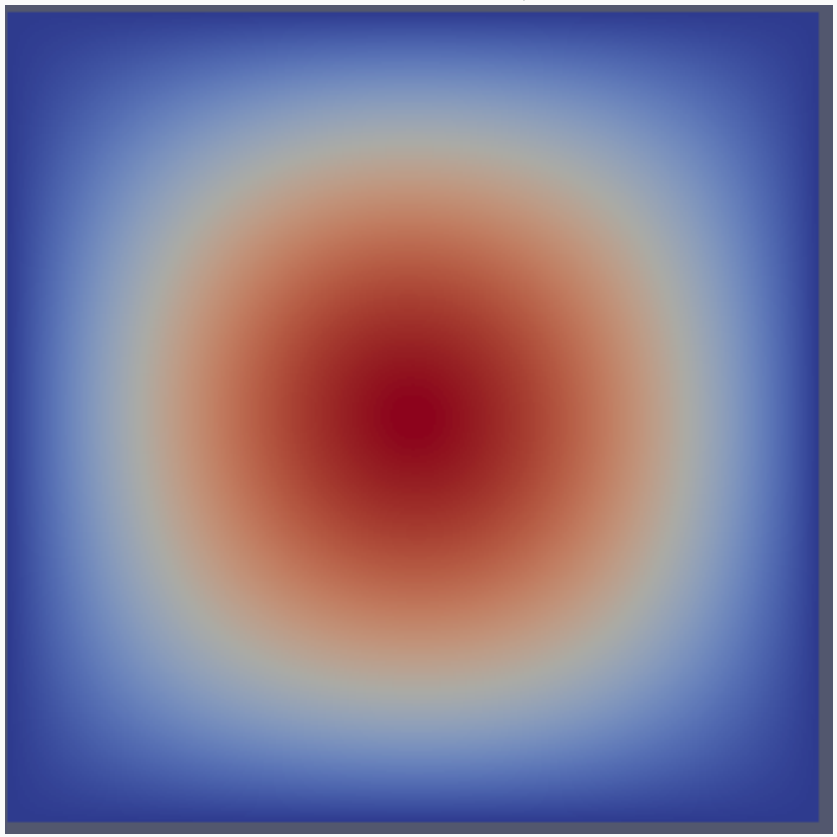
fixed $p = 1$



$h = 1/2^{\mathbf{5}}$

N=1089

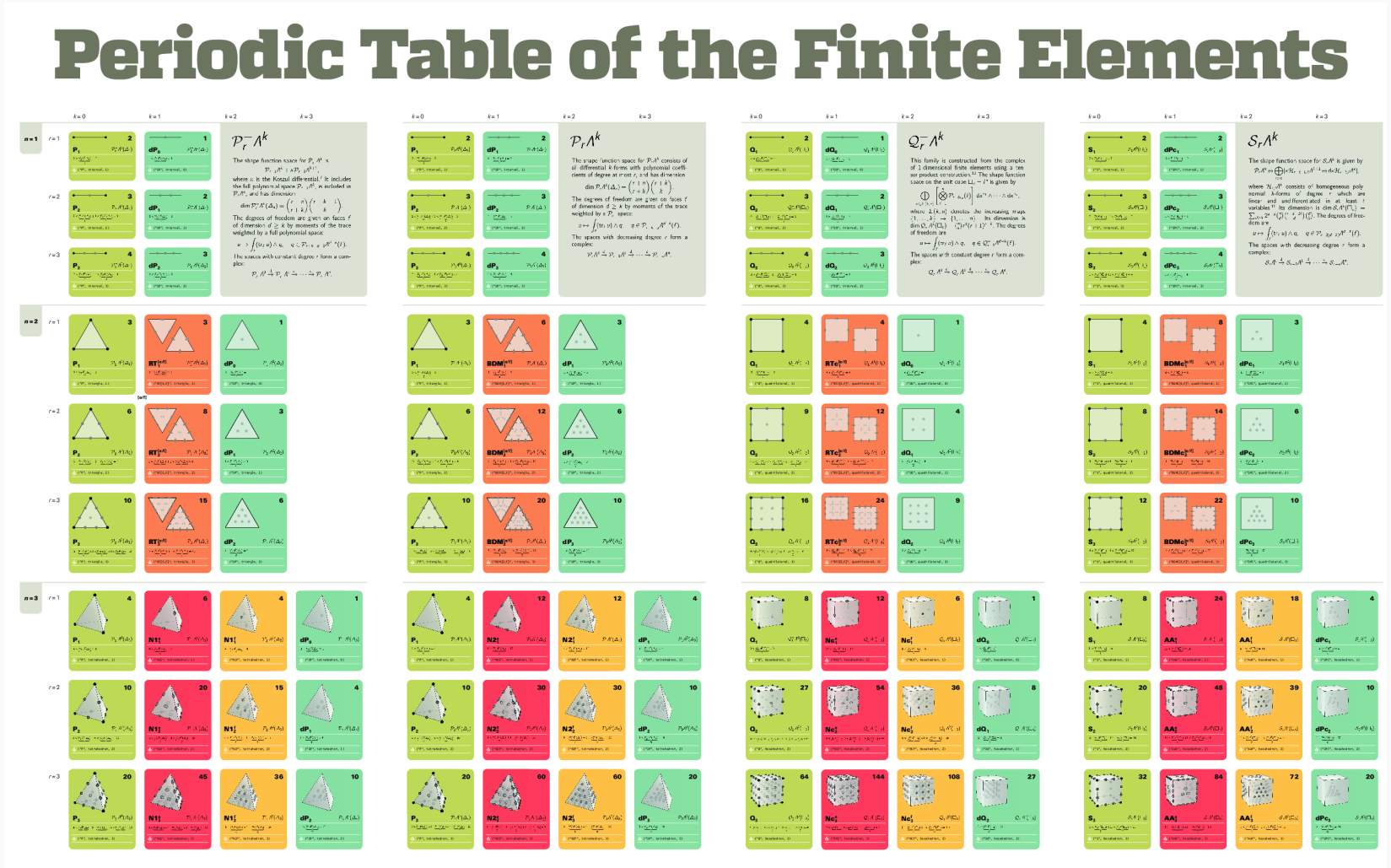
fixed $h = 1/2$



$p = \mathbf{5}$

N=121

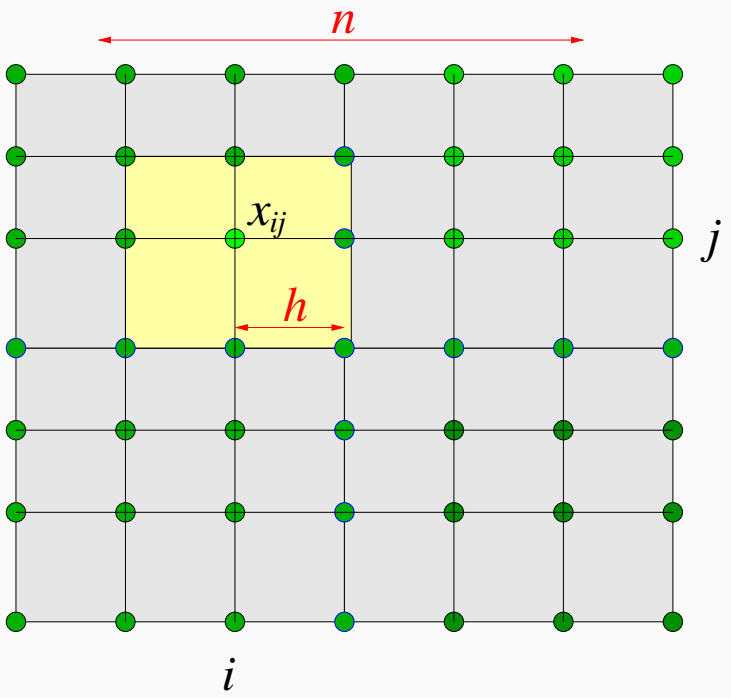
More finite elements...



Arnold, Logg (2014) *SIAM News*

What are discontinuous finite elements?

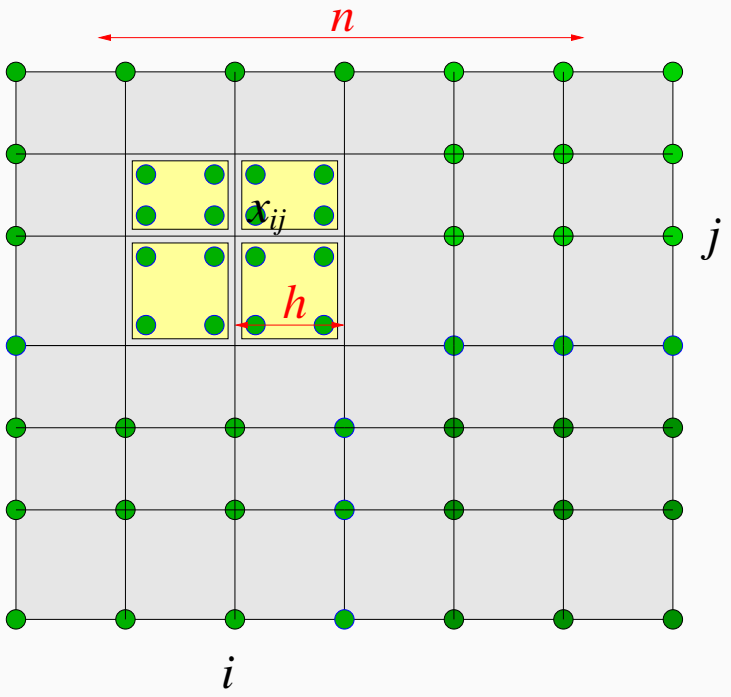
'Continuous' finite elements:



$$V_h = \{v \in C(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\} \subset H_0^1(\Omega)$$

Discontinuous finite elements

'Discontinuous' finite elements:

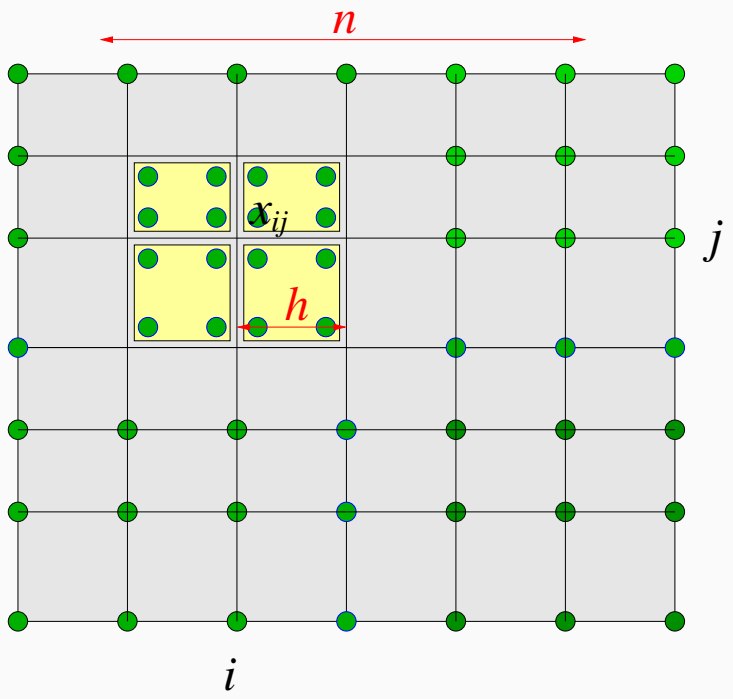


$$V_h^p = \{v \in L^2(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\} \not\subseteq H_0^1(\Omega)$$

...allow for using discontinuous basis functions.

Discontinuous finite elements

'Discontinuous' finite elements:



$$V_h^p = \{v \in L^2(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\} \not\subseteq H_0^1(\Omega)$$

...allow for using discontinuous basis functions.

More degrees of freedom, but: easy h–refinement and p–refinement (nonconforming elements allowed by design)

DGFEM approximation of the model problem

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Problem (DGFEM approximation)

$u_h, v_h \in V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}$

$$\sum_{\kappa \in \mathcal{T}_h} \int_\kappa \nabla u_h \cdot \nabla v_h \, dx$$

$$= (f, v_h)_\Omega$$

Divide and... reconnect (weakly).

Arnold (1982) *SIAM J. Numer. Anal.*

DGFEM approximation of the model problem

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Problem (DGFEM approximation)

$u_h, v_h \in V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}$

$$\sum_{\kappa \in \mathcal{T}_h} \int_\kappa \nabla u_h \cdot \nabla v_h \, dx + \sum_{e \in \mathcal{E}_h} \int_e \frac{\gamma p^2}{h} [u_h] \cdot [v_h] \, d\sigma$$

$$= (f, v_h)_\Omega$$

Divide and... reconnect (weakly).

Arnold (1982) *SIAM J. Numer. Anal.*

DGFEM approximation of the model problem

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Problem (DGFEM approximation)

$u_h, v_h \in V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}$

$$\sum_{\kappa \in \mathcal{T}_h} \int_\kappa \nabla u_h \cdot \nabla v_h \, dx + \sum_{e \in \mathcal{E}_h} \int_e \frac{\gamma p^2}{h} [u_h] \cdot [v_h] \, d\sigma$$

$$- \sum_{e \in \mathcal{E}_h} \int_e \{\nabla u_h\}_\omega \cdot [v_h] \, d\sigma$$

$$= (f, v_h)_\Omega$$

Divide and... reconnect (weakly).

Arnold (1982) *SIAM J. Numer. Anal.*

DGFEM approximation of the model problem

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Problem (DGFEM approximation)

$u_h, v_h \in V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}$

$$\sum_{\kappa \in \mathcal{T}_h} \int_\kappa \nabla u_h \cdot \nabla v_h \, dx + \sum_{e \in \mathcal{E}_h} \int_e \frac{\gamma p^2}{h} [u_h] \cdot [v_h] \, d\sigma$$
$$- \sum_{e \in \mathcal{E}_h} \int_e \{\nabla u_h\}_\omega \cdot [v_h] \, d\sigma$$
$$- \sum_{e \in \mathcal{E}_h} \int_e \{\nabla v_h\}_\omega \cdot [u_h] \, d\sigma = (f, v_h)_\Omega$$

Divide and... reconnect (weakly).

Arnold (1982) *SIAM J. Numer. Anal.*

DGFEM approximation of the model problem

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Problem (DGFEM approximation)

$u_h, v_h \in V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}$

$$\begin{aligned} \mathcal{A}_h(u_h, v_h) \equiv & \sum_{\kappa \in \mathcal{T}_h} \int_\kappa \nabla u_h \cdot \nabla v_h \, dx + \sum_{e \in \mathcal{E}_h} \int_e \frac{\gamma p^2}{h} [u_h] \cdot [v_h] \, d\sigma \\ & - \sum_{e \in \mathcal{E}_h} \int_e \{\nabla u_h\}_\omega \cdot [v_h] \, d\sigma \\ & - \sum_{e \in \mathcal{E}_h} \int_e \{\nabla v_h\}_\omega \cdot [u_h] \, d\sigma = (f, v_h)_\Omega \end{aligned}$$

Divide and... reconnect (weakly).

Arnold (1982) *SIAM J. Numer. Anal.*

DGFEM approximation of the model problem

Problem

Find $u \in H_0^1(\Omega)$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in H_0^1(\Omega).$$

Problem (DGFEM approximation)

$u_h, v_h \in V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}$

$$\mathcal{A}_h(u_h, v_h) \equiv \sum_{\kappa \in \mathcal{T}_h} \int_\kappa \nabla u_h \cdot \nabla v_h \, dx + \ \textit{...iterface terms...} \ = (f, v_h)_\Omega$$

Divide and... reconnect (weakly).

Arnold (1982) *SIAM J. Numer. Anal.*

Di Pietro, Ern (2012) *Mathematical aspects of discontinuous Galerkin methods*

FEM/DGFEM stiffness matrix

Find $u_h \in V_h^p$ such that

$$\mathcal{A}_h(u_h,\, v_h) = (f\,, v_h)_\Omega \qquad \forall v_h \in V_h^p.$$

Let $V_h^p = \text{span}\{\phi_1, \ldots, \phi_N\}$ and expand $u_h = \sum_i u_i \phi_i$.

FEM/DGFEM stiffness matrix

Find $u_h \in V_h^p$ such that

$$\mathcal{A}_h(u_h,\, v_h) = (f\,, v_h)_\Omega \qquad \forall v_h \in V_h^p.$$

Let $V_h^p = \text{span}\{\phi_1, \ldots, \phi_N\}$ and expand $u_h = \sum_i u_i \phi_i$.

Then $u = [u_1, \ldots, u_N] \in \mathbb{R}^N$ satisfies

$$Lu = b$$

where

$$L_{ij} = \mathcal{A}_h(\phi_i,\, \phi_j), \qquad i, j = 1, \ldots, N.$$

FEM/DGFEM stiffness matrix

Find $u_h \in V_h^p$ such that

$$\mathcal{A}_h(u_h,\, v_h) = (f\,, v_h)_\Omega \qquad \forall v_h \in V_h^p.$$

Let $V_h^p = \text{span}\{\phi_1, \ldots, \phi_N\}$ and expand $u_h = \sum_i u_i \phi_i$.

Then $u = [u_1, \ldots, u_N] \in \mathbb{R}^N$ satisfies

$$Lu = b$$

where

$$L_{ij} = \mathcal{A}_h(\phi_i,\, \phi_j), \qquad i, j = 1, \ldots, N.$$

Properties of stiffness matrix L:

- symmetric and positive definite: $L = L^T > 0$
- N can be as **large** as one can afford ($h \searrow 0$, $p \nearrow$ large)
- **sparse**: each row has only a few nonzero elements

Solving large sparse systems

No need for Gaussian elimination

Approximate solution to $Lx = b$ is a **reasonable choice**.

No need for Gaussian elimination

Approximate solution to $Lx = b$ is a **reasonable choice**.

Model iteration:

$$x_{n+1} = x_n + \tau D^{-1}(b - Lx_n) \qquad \textit{(damped Jacobi iteration)}$$

$$L = D - A, \qquad \textit{(D is the diagonal of L)}$$

No need for Gaussian elimination

Approximate solution to $Lx = b$ is a **reasonable choice**.

Model iteration:

$$x_{n+1} = x_n + \tau D^{-1}(b - Lx_n) \qquad \textit{(damped Jacobi iteration)}$$

$$L = D - A, \qquad \textit{(D is the diagonal of L)}$$

Divide and... be patient: for $L = L^T > 0$,

- with optimal damping τ, convergence driven by the **condition number**

$$\kappa = \frac{\lambda_{\max}(D^{-1}L)}{\lambda_{\min}(D^{-1}L)}$$

- error reduction:

$$\|x_{n+1} - x\| \lesssim \underbrace{\frac{\kappa - 1}{\kappa + 1}}_{=\gamma} \|x_n - x\|$$

Iterative solution of $Lx = b$

- Model iterative method:

$$x_{n+1} = x_n + \tau D^{-1}(b - Lx_n)$$

Iterative solution of $Lx = b$

- Model iterative method:

$$x_{n+1} = x_n + \tau D^{-1}(b - Lx_n)$$

- error reduction factor $\gamma = \dfrac{\kappa - 1}{\kappa + 1}$ depends on $\kappa = \dfrac{\lambda_{\max}(D^{-1}L)}{\lambda_{\min}(D^{-1}L)}$

Iterative solution of $Lx = b$

- Model iterative method:

$$x_{n+1} = x_n + \tau D^{-1}(b - Lx_n)$$

- error reduction factor $\gamma = \dfrac{\kappa - 1}{\kappa + 1}$ depends on $\kappa = \dfrac{\lambda_{\max}(D^{-1}L)}{\lambda_{\min}(D^{-1}L)}$

Iterative solution of $Lx = b$

- Model iterative method:

$$x_{n+1} = x_n + \tau D^{-1}(b - Lx_n)$$

- error reduction factor $\gamma = \dfrac{\kappa - 1}{\kappa + 1}$ depends on $\kappa = \dfrac{\lambda_{\max}(D^{-1}L)}{\lambda_{\min}(D^{-1}L)}$

Problem

If $D^{-1}L$ is ill–conditioned: $\kappa \gg 1 \quad \Longrightarrow \quad \gamma \approx 1.$

Iterative solution of $Lx = b$

- Model iterative method:

$$x_{n+1} = x_n + \tau D^{-1}(b - Lx_n)$$

- error reduction factor $\gamma = \dfrac{\kappa - 1}{\kappa + 1}$ depends on $\kappa = \dfrac{\lambda_{\max}(D^{-1}L)}{\lambda_{\min}(D^{-1}L)}$

Problem

If $D^{-1}L$ is ill–conditioned: $\kappa \gg 1 \quad \Longrightarrow \quad \gamma \approx 1$.
Our L from finite element method is ill–conditioned: $p \nearrow \infty$, $h \searrow 0$ and

$$\kappa(L) = O(p^4/h^2)$$

$D^{-1}L$ is ill–conditioned, too.

Iterative solution of $Lx = b$

- Model iterative method:

$$x_{n+1} = x_n + \tau P^{-1}(b - Lx_n)$$

- error reduction factor $\gamma = \dfrac{\kappa - 1}{\kappa + 1}$ depends on $\kappa = \dfrac{\lambda_{\max}(P^{-1}L)}{\lambda_{\min}(P^{-1}L)}$

Problem

If $P^{-1}L$ is ill–conditioned: $\kappa \gg 1 \quad \Longrightarrow \quad \gamma \approx 1$.
Our L from finite element method is ill–conditioned: $p \nearrow \infty$, $h \searrow 0$ and

$$\kappa(L) = O(p^4/h^2)$$

$D^{-1}L$ is ill–conditioned, too.

Divide and... use a good preconditioner P.
If $P^{-1}L$ is well–conditioned: $\kappa \approx 1 \quad \Longrightarrow \quad \gamma \ll 1$.

What makes a good preconditioner?

Simple **preconditioned** iteration:

$$x_{n+1} = x_n + P^{-1}(b - Lx_n)$$

Ideally, P should:

- be easy to construct,
- be easy to invert (i.e. solving a system with P is cheap),
- reduce the condition number: $\kappa(P^{-1}L) \ll \kappa(L)$.

These rules apply when simple iteration is replaced with a better method (e.g. Conjugate Gradients).

What makes a good preconditioner?

Simple **preconditioned** iteration:

$$x_{n+1} = x_n + P^{-1}(b - Lx_n)$$

Ideally, P should:

- be easy to construct,
- be easy to invert (i.e. solving a system with P is cheap),
- reduce the condition number: $\kappa(P^{-1}L) \ll \kappa(L)$.

These rules apply when simple iteration is replaced with a better method (e.g. Conjugate Gradients).

Extreme case: $P = I$ does not satisfy *all* requirements.

What makes a good preconditioner?

Simple **preconditioned** iteration:

$$x_{n+1} = x_n + P^{-1}(b - Lx_n)$$

Ideally, P should:

- be easy to construct,
- be easy to invert (i.e. solving a system with P is cheap),
- reduce the condition number: $\kappa(P^{-1}L) \ll \kappa(L)$.

These rules apply when simple iteration is replaced with a better method (e.g. Conjugate Gradients).

Extreme case: $P = L$ does not satisfy *all* requirements as well.

Guidelines for choosing efficient P

$L = L^T > 0$, so choose $P = P^T > 0$.

- Impose *spectral equivalence*: if exist $C_0,\ C_1 > 0$ *independent* of $h,\ p,\ \ldots$, such that

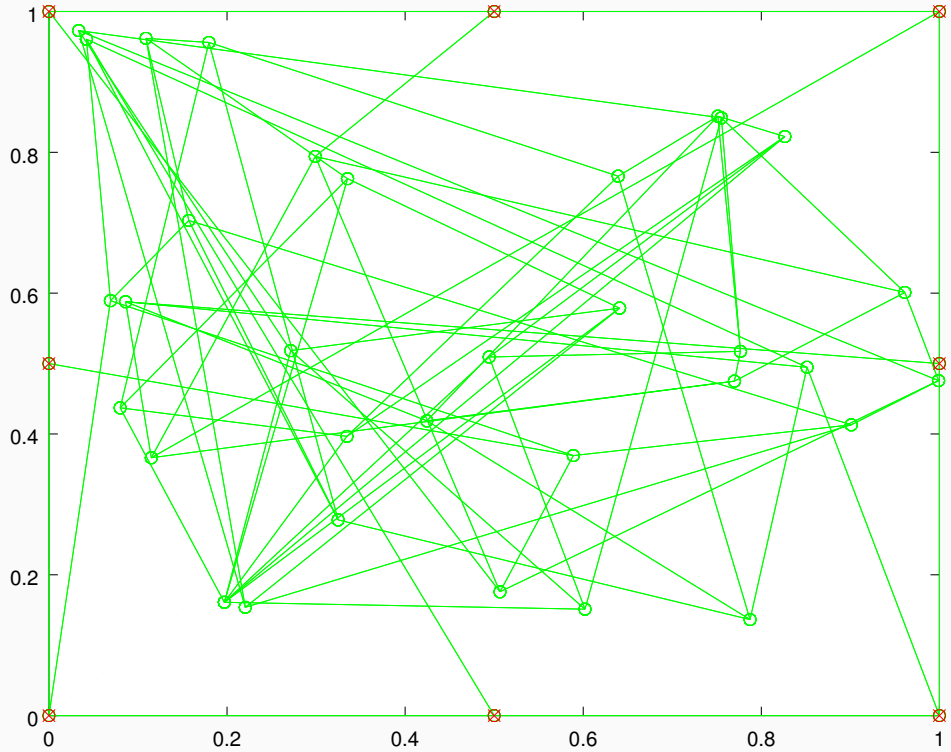
$$C_0\, x^T P x \leq x^T L x \leq C_1\, x^T P x \qquad \Longrightarrow \qquad \kappa(P^{-1}L) \leq \frac{C_1}{C_0}.$$

  $\rightarrow$ This makes the number of iterations independent of problem size.

- *Think globally, act locally*: embrace parallelism.
  $\rightarrow$ This makes each iteration fast.

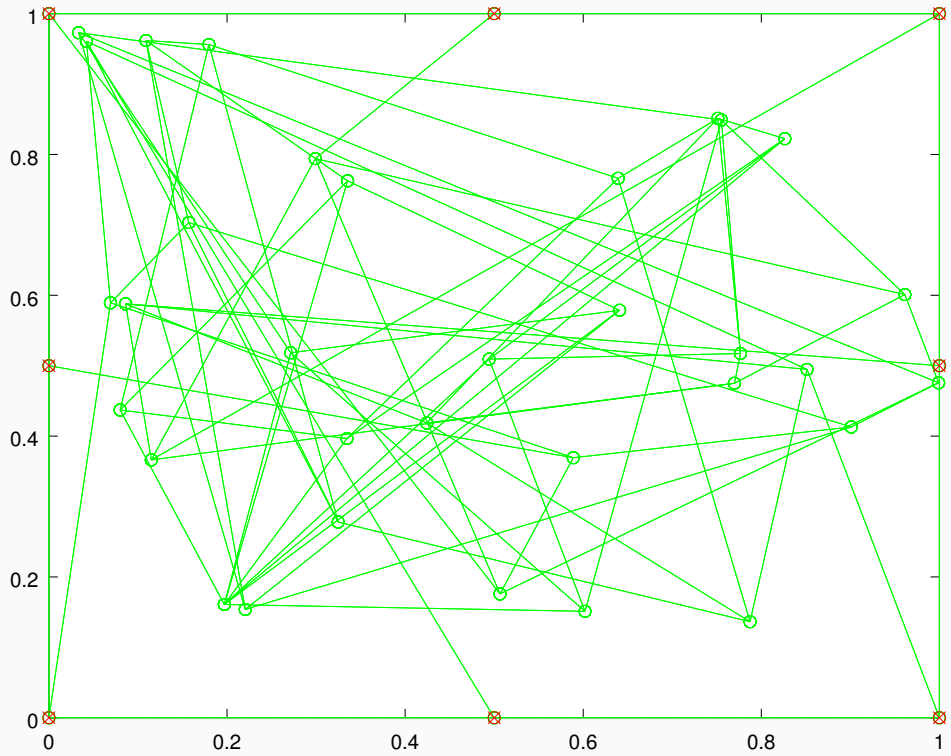
A sidenote: another class of structured sparse matrices

Pretty drawing graphs



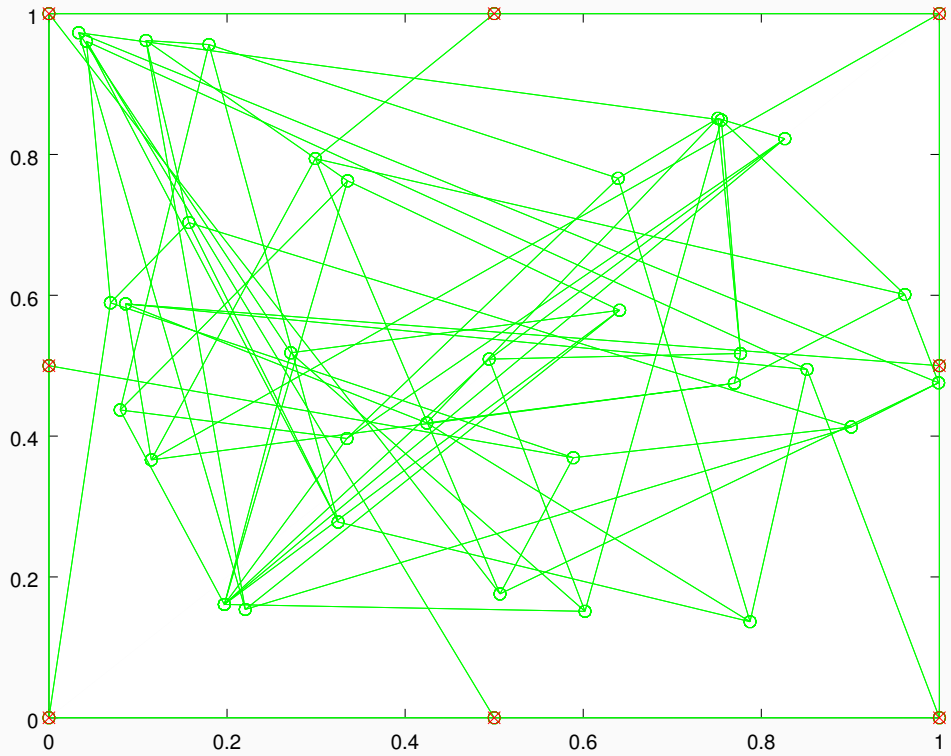
Spider's messy net: how to draw it *nicely*?

Graph Laplacians: pretty drawing graphs



- Assume edges are elastic threads, obeying (linear!) Hooke's law

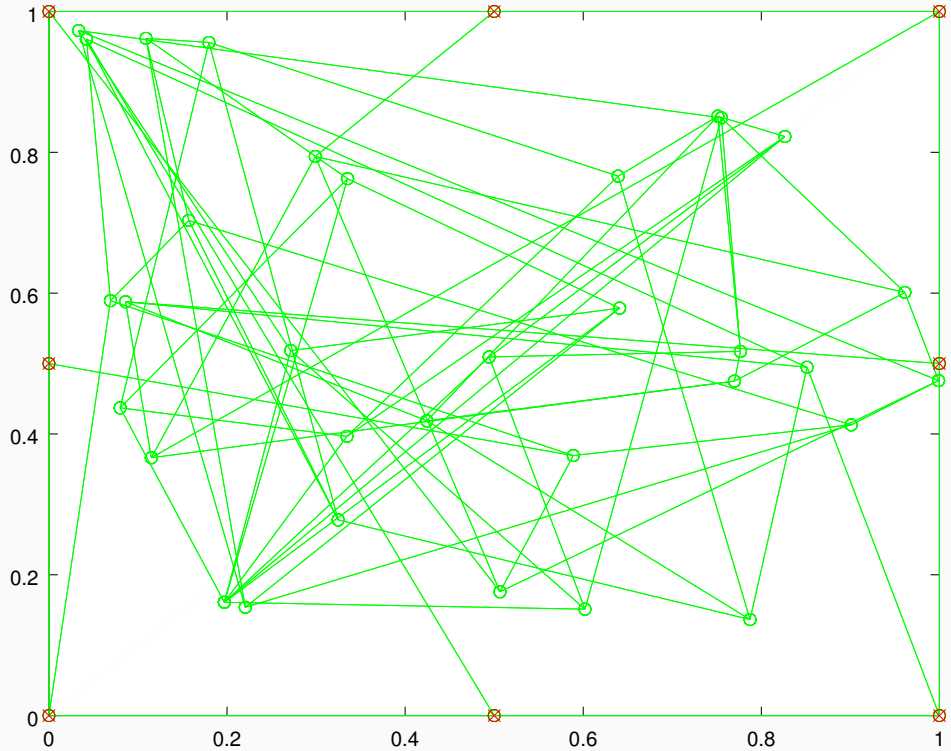
Graph Laplacians: pretty drawing graphs



- Assume edges are elastic threads, obeying (linear!) Hooke's law
- Fix positions of some nodes

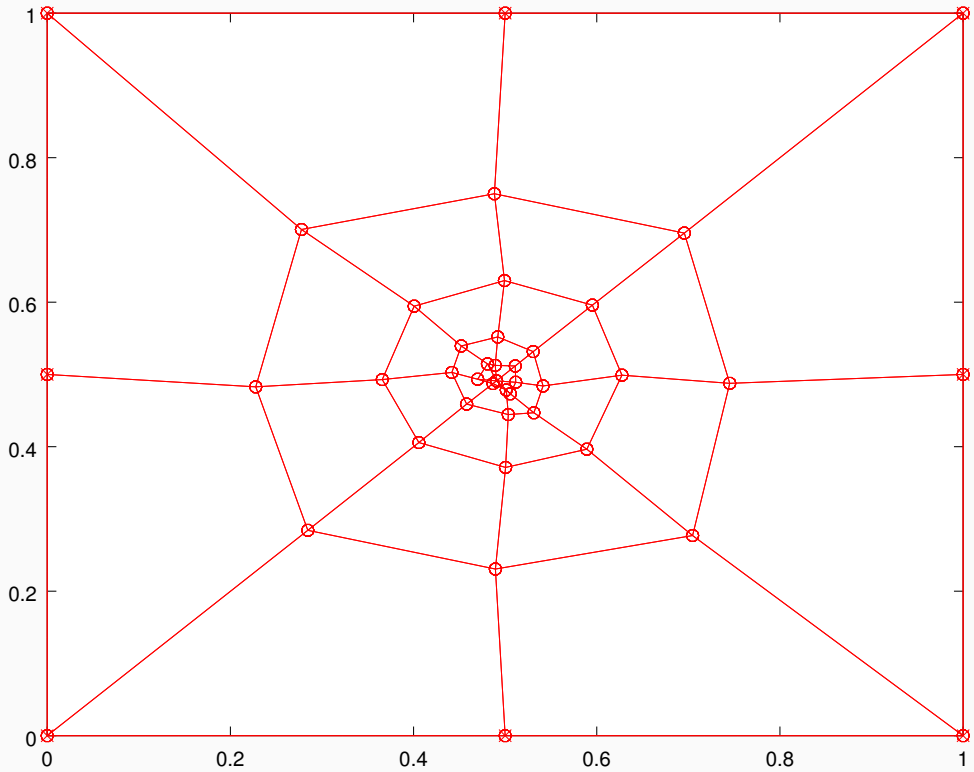
Graph Laplacians: pretty drawing graphs

- Assume edges are identical elastic threads, obeying (linear!) Hooke's law
- Fix positions of some nodes
- **Solve** for other positions:



Graph Laplacians: pretty drawing graphs

- Assume edges are identical elastic threads, obeying (linear!) Hooke's law
- Fix positions of some nodes
- **Solve** for other positions:

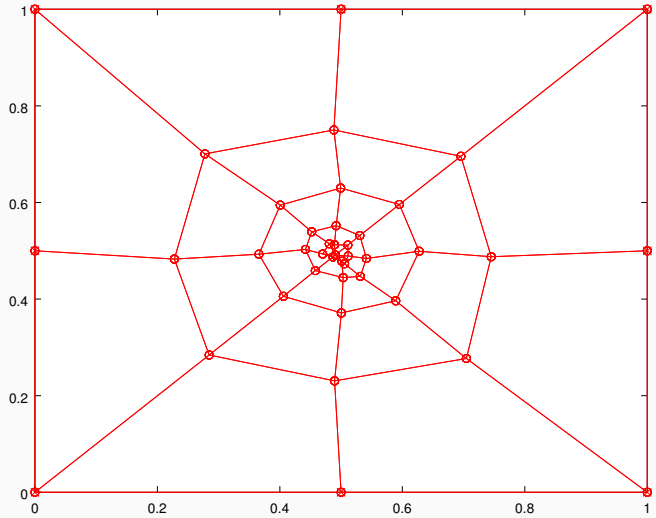


Graph Laplacian

Simple, unidirected, weighted **graph** (V, E) (e.g. social network, transport network, electric circuit, ...)

- Vertices $V = \{1, \ldots, N\}$
- Edge between $i, j \in V$ denoted (i, j); the set of all edges: E;
- **Degree** of vertex i is

$$D_{ii} = \sum_{j:(i,j)\in E} w_{ij}.$$



- Adjacency matrix: $A_{ij} = w_{ij}$ if $(i, j) \in E$; zero otherwise.
- **Graph Laplacian:** $L = D - A$; equivalently

$$x^T L x = \sum_{(i,j)\in E} w_{ij}(x_i - x_j)^2.$$

$$L = L^T \geq 0$$

Graph Laplacian

Reasons to solve systems $Lx = b$ with graph Laplacian:

- drawing pretty graphs

Graph Laplacian

Reasons to solve systems $Lx = b$ with graph Laplacian:

- drawing pretty graphs (or improving quality of finite element meshes)

Graph Laplacian

Reasons to solve systems $Lx = b$ with graph Laplacian:

- drawing pretty graphs (or improving quality of finite element meshes)
- finding voltages in a resistor network, with some input/output voltages fixed

Graph Laplacian

Reasons to solve systems $Lx = b$ with graph Laplacian:

- drawing pretty graphs (or improving quality of finite element meshes)
- finding voltages in a resistor network, with some input/output voltages fixed
- finding Fiedler vector of the graph (using inverse power iteration)

Graph Laplacian

Reasons to solve systems $Lx = b$ with graph Laplacian:

- drawing pretty graphs (or improving quality of finite element meshes)
- finding voltages in a resistor network, with some input/output voltages fixed
- finding Fiedler vector of the graph (using inverse power iteration) (e.g. for mesh partitioning)

Graph Laplacian

Reasons to solve systems $Lx = b$ with graph Laplacian:

- drawing pretty graphs (or improving quality of finite element meshes)
- finding voltages in a resistor network, with some input/output voltages fixed
- finding Fiedler vector of the graph (using inverse power iteration) (e.g. for mesh partitioning)

Some graphs have **very large** number of vertices N. But then usually every node is connected to only a few others: the graph is **sparse**:

$$\forall i \qquad L_{ij} \neq 0 \quad \text{only for several j}$$

We experienced this browsing through the *Sparse Matrix Collection*!

Domain decomposition for PDEs

What makes a good preconditioner? (replay)

We are solving

$$Lx = b.$$

with $L = L^T > 0$.
Simple **preconditioned** iteration:

$$x_{n+1} = x_n + P^{-1}(b - Lx_n)$$

P must:

- be easy to construct,
- be easy to invert (i.e. solve a system with P),
- reduce the condition number: $\kappa(P^{-1}L) \ll \kappa(L)$.

These rules apply when simple iteration is replaced with a better method (e.g. Conjugate Gradients).

Guidelines for choosing efficient P (replay)

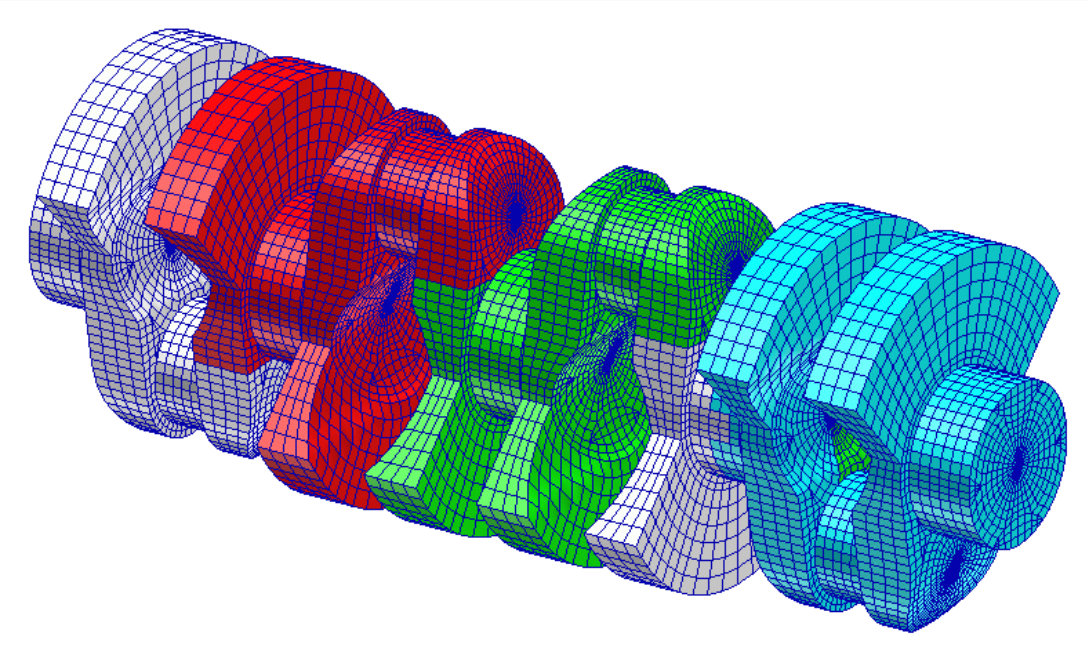
$L = L^T > 0$, choose $P = P^T > 0$.

- Impose spectral equivalence:

$$C_0\, x^T P x \leq x^T L x \leq C_1\, x^T P x \qquad \Longrightarrow \qquad \kappa(P^{-1}L) \leq \frac{C_1}{C_0}.$$

- Use full processing power: embrace **parallelism**.

Domain decomposition



Source: MSC/PARASOL

Divide and... solve smaller problems in parallel.

Then ,,glue" them together.

Additive Schwarz method

Problem

Find $u_h \in V_h$ *such that*

$$\mathcal{A}_h(u_h,\, v_h) = (f, v_h) \qquad \forall v_h \in V_h.$$

Additive Schwarz method

Problem

Find $u_h \in V_h$ *such that*

$$\mathcal{A}_h(u_h,\, v_h) = (f, v_h) \qquad \forall v_h \in V_h.$$

Divide and... add:

- Space decomposition:

$$V_h = V_0 + V_1 + \ldots + V_N.$$

Additive Schwarz method

Problem

Find $u_h \in V_h$ such that

$$\mathcal{A}_h(u_h,\, v_h) = (f, v_h) \qquad \forall v_h \in V_h.$$

Divide and... add, and solve in parallel.

- Space decomposition:

$$V_h = V_0 + V_1 + \ldots + V_N.$$

- Local solution operators $T_i : V_h \to V_i$ such that

$$A_h(T_i u_i,\, v_i) = \mathcal{A}_h(u_i,\, v_i) \qquad \forall v_i \in V_i.$$

Additive Schwarz method

Theorem (Divide and... maintain stability)

Let $T = T_0 + T_1 + \ldots + T_N$. Suppose that the following hold:

Stable decomposition: $\exists\, C > 0 \quad \exists\, u_i \in V_i,\ u = \sum_i u_i$

$$\sum_i A_h(u_i,\, u_i) \le C\, \mathcal{A}_h(u,\, u) \quad \forall\, u \in V_h$$

Strengthened Cauchy–Schwarz ineq.: $\exists\, 0 \le \mathcal{E}_{ij} \le 1\ \forall\, 1 \le i, j \le N$

$$\mathcal{A}_h(u_i,\, u_j) \le \mathcal{E}_{ij} \cdot \mathcal{A}_h(u_i,\, u_i)^{1/2} \cdot \mathcal{A}_h(u_j,\, u_j)^{1/2} \quad \forall\, u_i \in V_i,\ u_j \in V_j,$$

Local stability: $\exists\, \omega > 0\ \forall\, 0 \le i \le N$

$$\mathcal{A}_h(u_i,\, u_i) \le \omega\, A_h(u_i,\, u_i) \quad \forall\, u_i \in V_i$$

Then

$$\kappa(T) \le C\, \omega\, (\rho(\mathcal{E}) + 1).$$

Dryja, Widlund (1990) "Towards a unified theory of domain decomposition algorithms for elliptic problems"

Additive/Multiplicative Schwarz method

30 years of successful applications:

- overlapping domain decomposition
- substructuring domain decomposition
- multigrid
- building block of PETSc parallel linear solvers library

Smith, Bjørstad, Gropp (1996) *Domain decomposition*

Toselli, Widlund (2005) *Domain decomposition methods—algorithms and theory*

Mathew (2008) *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*

Xu (1992) *SIAM Rev.*

Balay (1995–) *PETSc Users Manual*

ACM Gordon Bell prize 2016

"Chinese Research Team that Employs High Performance Computing to Understand Weather Patterns Wins 2016 ACM Gordon Bell Prize"

- World's fastest supercomputer Sunway TaihuLight, 10M cores

`http://awards.acm.org/bell`

ACM Gordon Bell prize 2016

"Chinese Research Team that Employs High Performance Computing to Understand Weather Patterns Wins 2016 ACM Gordon Bell Prize"

- World's fastest supercomputer Sunway TaihuLight, 10M cores
- System of equations with **770,000,000** unknowns

`http://awards.acm.org/bell`

ACM Gordon Bell prize 2016

"Chinese Research Team that Employs High Performance Computing to Understand Weather Patterns Wins 2016 ACM Gordon Bell Prize"

- World's fastest supercomputer Sunway TaihuLight, 10M cores
- System of equations with **770,000,000** unknowns
- *"In the solver, we propose a highly efficient* **domain-decomposed multigrid preconditioner** *that can greatly accelerate the convergence rate at the extreme scale. For solving the* **overlapped subdomain** *problems, a geometry-based pipelined* **incomplete LU factorization** *method is designed to further exploit the* **on-chip** *fine-grained* **concurrency**."

`http://awards.acm.org/bell`

ACM Gordon Bell prize 2016

"Chinese Research Team that Employs High Performance Computing to Understand Weather Patterns Wins 2016 ACM Gordon Bell Prize"

- World's fastest supercomputer Sunway TaihuLight, 10M cores
- System of equations with **770,000,000** unknowns
- *"In the solver, we propose a highly efficient* **domain-decomposed multigrid preconditioner** *that can greatly accelerate the convergence rate at the extreme scale. For solving the* **overlapped subdomain** *problems, a geometry-based pipelined* **incomplete LU factorization** *method is designed to further exploit the* **on-chip** *fine-grained* **concurrency**."
- Additive Schwarz at the core of computation

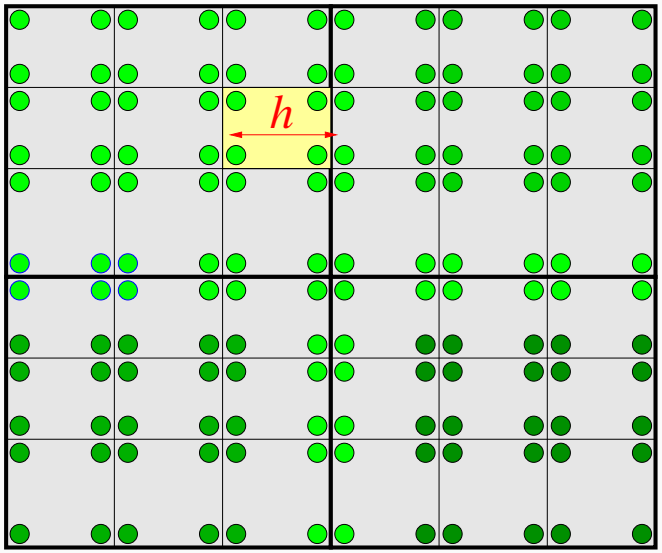
`http://awards.acm.org/bell`

Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$



Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

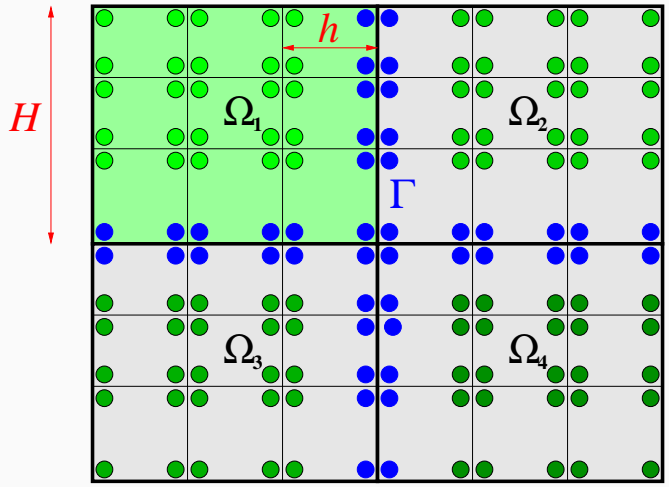
Decomposition:

$$V_h^p = \sum_{i=1}^{N} V_i$$

where

$$V_i = \{v \in V_h^p : v = 0 \text{ on } \Omega_j, \quad j \neq i\}$$

Divide and... aggregate.



Is there no overlap between subdomains?

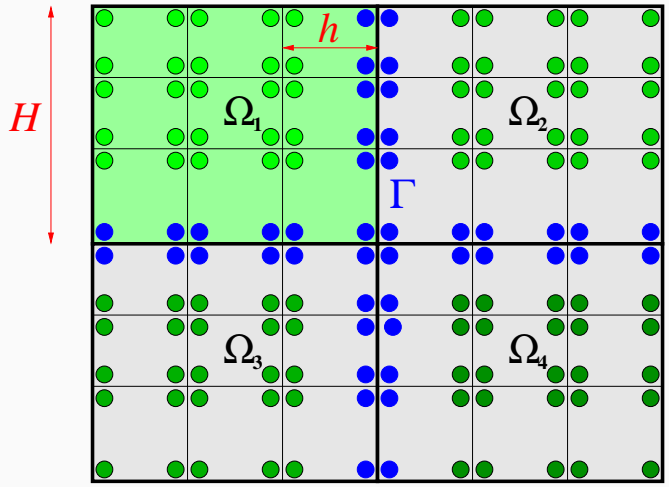
Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

Decomposition:

Divide and... aggregate.

$$V_h^p = \sum_{i=1}^{N} V_i$$



where

$$V_i = \{v \in V_h^p : v = 0 \text{ on } \Omega_j, \quad j \neq i\}$$

Is there no overlap between subdomains? Not really:

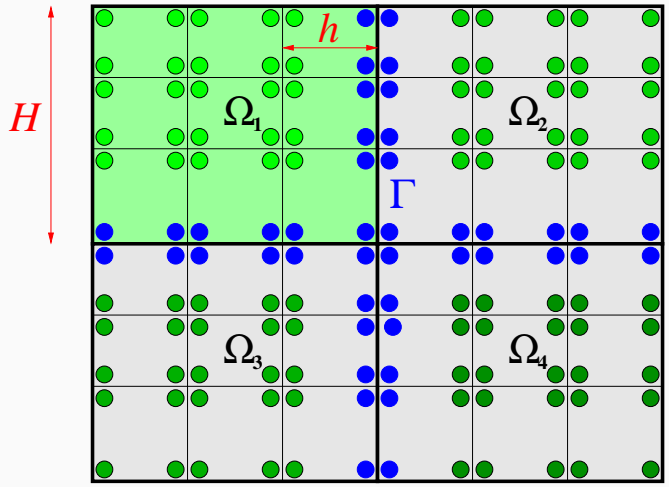
Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

Decomposition:

Divide and... aggregate.

$$V_h^p = \sum_{i=1}^{N} V_i$$



where

$$V_i = \{v \in V_h^p : v = 0 \text{ on } \Omega_j, \quad j \neq i\}$$

Is there no overlap between subdomains? Not really:

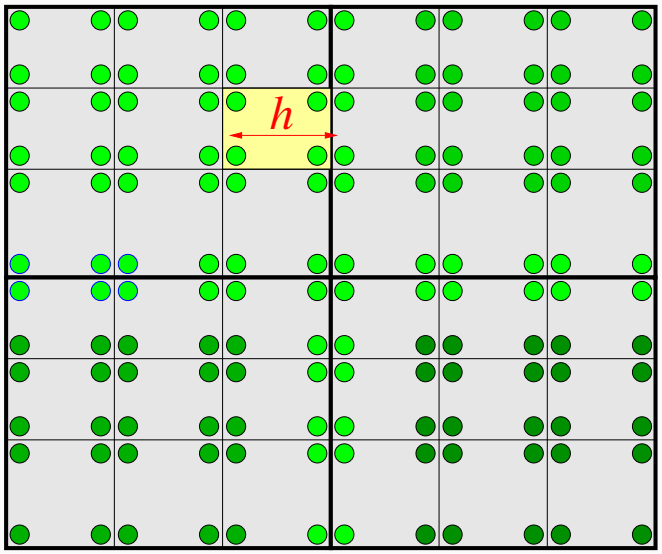
$$\mathcal{A}_h(u, v) \equiv \sum_{\kappa \in \mathcal{T}_h} \int_\kappa \nabla u \cdot \nabla v \, dx + \int_\Gamma \frac{\gamma p^2}{h} [u][v] \, d\sigma + \ldots \text{ etc.}$$

Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$



Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

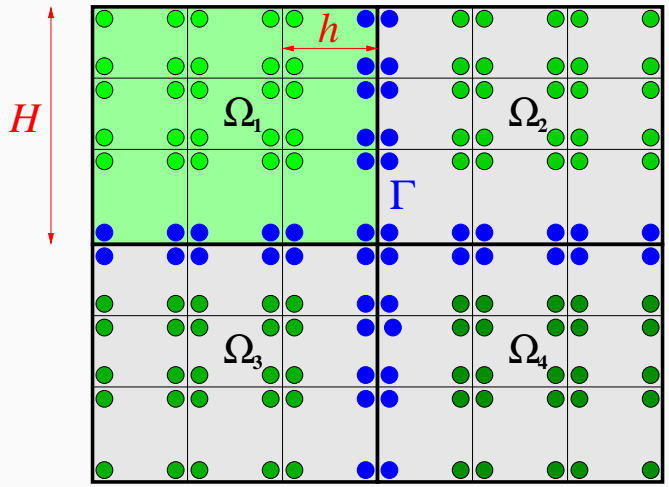
Decomposition:

$$V_h^p = \sum_{i=1}^{N} V_i$$

where

$$V_i = \{v \in V_h^p : v = 0 \text{ on } \Omega_j, \quad j \neq i\}$$

Divide and... aggregate.



Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

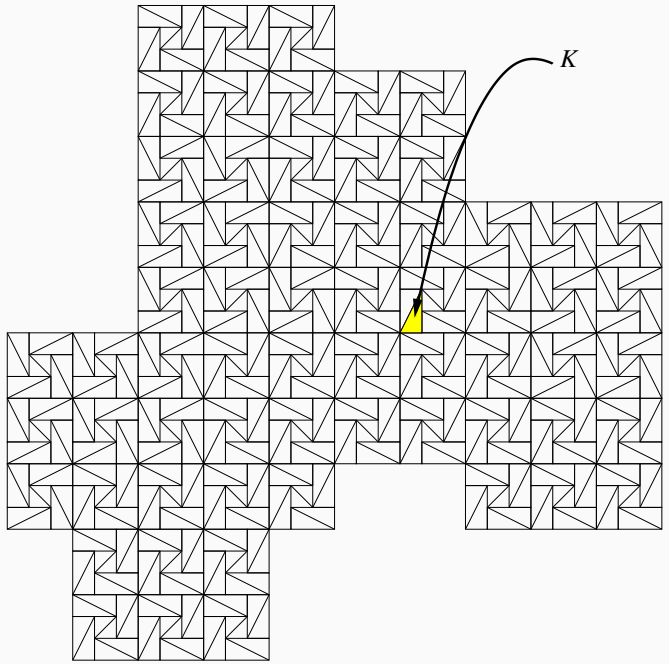
Decomposition:

$$V_h^p = \sum_{i=1}^{N} V_i$$

where

$$V_i = \{v \in V_h^p : v = 0 \text{ on } \Omega_j, \quad j \neq i\}$$

Divide and... aggregate.



Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

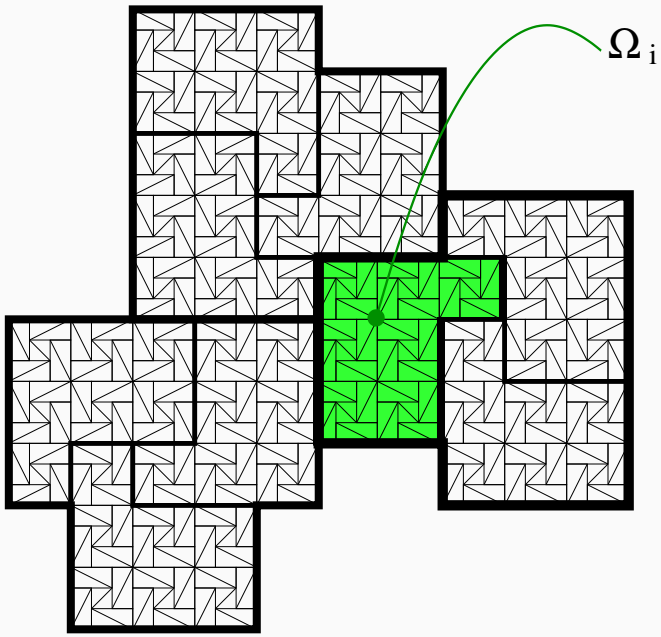
Decomposition:

$$V_h^p = \sum_{i=1}^{N} V_i$$

where

$$V_i = \{v \in V_h^p : v = 0 \text{ on } \Omega_j, \quad j \neq i\}$$

Divide and... aggregate.



N subdomains

Non–overlapping domain decomposition for DGFEM

$$V_h^p = \{v \in L^2(\Omega) : v_{|_\kappa} \in P_p(\kappa) \quad \forall \kappa \in \mathcal{T}_h\}.$$

Decomposition:

$$V_h^p = \sum_{i=1}^{N} V_i$$

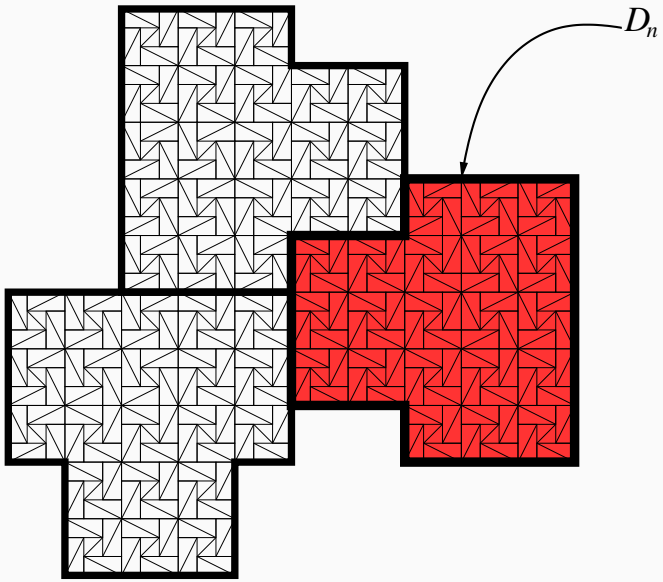
where

$$V_i = \{v \in V_h^p : v = 0 \text{ on } \Omega_j, \quad j \neq i\}$$

Coarse space:

$$V_0 = V_{\mathcal{H}}^q, \qquad \text{where } \mathcal{H} \geq H, \quad q \leq p.$$

Divide and... aggregate.



N subdomains, $\mathcal{M}$ coarse space cells.

DGFEM additive Schwarz condition estimate

Theorem

Let $T = T_0 + \sum_{i=1}^{N} T_i$ *be the preconditioned operator. Then*

$$\kappa(T) = O(\frac{\mathcal{H}^2}{hH} \cdot \frac{p^2}{\max\{q, 1\}})$$

Bound independent of discontinuities in the coefficient, extended to nonconforming meshes and varying polynomial degree.

Antonietti, Houston (2011) *J. Sci. Comput.*
K. (2016) *Num. Meth. PDEs*
Antonietti, Houston, Smears (2016) *Int. J. Numer. Anal. Model.*

DGFEM additive Schwarz condition estimate

Key condition for the coarse space V_0:

Divide and... **maintain approximation**:

$\forall u \in V_h \quad \exists u_0 \in V_0$:

$$\sum_{n=1}^{\mathcal{M}} \left(\frac{q_n^2}{\mathcal{H}_n^2} \|u - u_0\|_{0,D_n}^2 + \|u - u_0\|_{D_n}^2 \right) \leq \text{Const} \cdot \mathcal{A}_h(u,\, u).$$

K. (2016) *Num. Meth. PDEs*

Antonietti, Houston, Smears (2016) *Int. J. Numer. Anal. Model.*

DGFEM additive Schwarz condition estimate

Key condition for the coarse space V_0:

Divide and... maintain approximation:

$\forall u \in V_h \quad \exists u_0 \in V_0$:

$$\sum_{n=1}^{\mathcal{M}} \left(\frac{q_n^2}{\mathcal{H}_n^2} \|u - u_0\|_{0,D_n}^2 + \|u - u_0\|_{D_n}^2 \right) \leq \text{Const} \cdot \mathcal{A}_h(u,\, u).$$

Open questions:

- Optimal balance between $\mathcal{H}$ and H? p and q?
- How does it depend on the computer architecture?

K. (2016) *Num. Meth. PDEs*

Antonietti, Houston, Smears (2016) *Int. J. Numer. Anal. Model.*

What kind of parallelism?

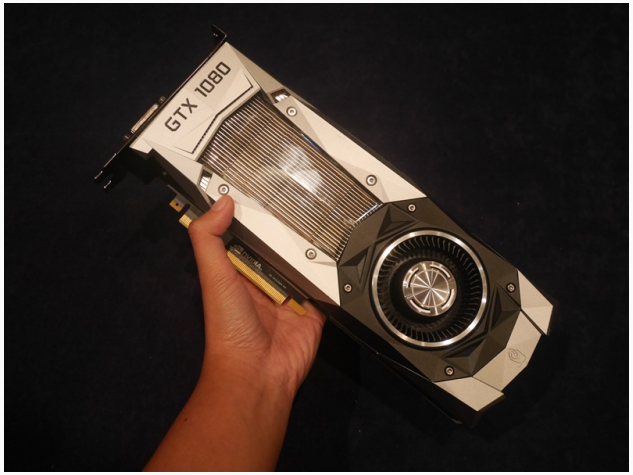
(24 cores, 2.6 GHz, 128GB) $\times$ 1084 nodes (Cray XC40, at ICM UW)

or...

What kind of parallelism?

(24 cores, 2.6 GHz, 128GB) $\times$ 1084 nodes (Cray XC40, at ICM UW)

or...



2560 cores, 1.6 GHz, 8 GB (NVIDIA GTX 1080, in your PC)

Extreme parallelism

Suppose subdomain = single finite element.

Then # parallel tasks = # subdomains = # finite elements = N.

Theorem

Let $T = T_0 + \sum_{i=1}^{N} T_i$ be the preconditioned operator. Then

$$\kappa(T) \lesssim \max_{n=1,\ldots,\mathcal{M}} \left\{ \frac{\mathcal{H}_n^2}{\min_{\kappa \in \mathcal{T}_h(D_n)} h_\kappa^2} \right\}.$$

Bound independent of discontinuities in the coefficient (under certain assumptions).

Dryja, K. (2015) *Num. Math.*

Extreme parallelism

Suppose subdomain = single finite element.

Then # parallel tasks = # subdomains = # finite elements = N.

Theorem

Let $T = T_0 + \sum_{i=1}^{N} T_i$ be the preconditioned operator. Then

$$\kappa(T) \lesssim \frac{\mathcal{H}^2}{h^2}.$$

Bound independent of discontinuities in the coefficient (under certain assumptions).

Dryja, K. (2015) *Num. Math.*

Splitting equations

Block systems

System with nonsinglar, symmetric 2×2 block matrix:

$$\mathcal{L} \begin{bmatrix} u \\ p \end{bmatrix} \equiv \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}.$$

Block systems

System with nonsinglar, symmetric 2×2 block matrix:

$$\mathcal{L} \begin{bmatrix} u \\ p \end{bmatrix} \equiv \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}.$$

Examples of "natural" block decomposition:

- $A > 0$, $C = 0$
  - Stokes equations,
  - mixed methods for elliptic PDEs,
- $A > 0$, $C < 0$
  - structured methods for elliptic PDEs:
- $A > 0$, $C > 0$
  - linear elasticity mixed discretization
  - stabilized mixed methods
- A indefinite, $C > 0$
  - time harmonic Maxwell equations

A family of preconditioners

For ill-conditioned $\mathcal{L}$, use preconditioner $\mathcal{P}$, and solve iteratively

$$\mathcal{P}^{-1} \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \mathcal{P}^{-1} \begin{bmatrix} F \\ G \end{bmatrix}$$

Benzi, Golub, Liesen (2005) *Acta Numer.*
K. (2011) *Efficient preconditioned [...] PDEs*
Brown (2012) *Intl. Symp. Para. Distr. Comp.*

A family of preconditioners

For ill-conditioned $\mathcal{L}$, use preconditioner $\mathcal{P}$, and solve iteratively

$$\mathcal{P}^{-1} \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \mathcal{P}^{-1} \begin{bmatrix} F \\ G \end{bmatrix}$$

Divide and... follow this decomposition!

$$\mathcal{P}_1 = \begin{bmatrix} I & \\ c\,B\,A_0^{-1} & I \end{bmatrix} \begin{bmatrix} A_0 & \\ & S_0 \end{bmatrix} \begin{bmatrix} I & d\,A_0^{-1}\,B^T \\ & I \end{bmatrix}$$

Benzi, Golub, Liesen (2005) *Acta Numer.*
K. (2011) *Efficient preconditioned [...] PDEs*
Brown (2012) *Intl. Symp. Para. Distr. Comp.*

A family of preconditioners

For ill-conditioned $\mathcal{L}$, use preconditioner $\mathcal{P}$, and solve iteratively

$$\mathcal{P}^{-1} \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \mathcal{P}^{-1} \begin{bmatrix} F \\ G \end{bmatrix}$$

Divide and... follow this decomposition!

$$\mathcal{P}_1 = \begin{bmatrix} I & \\ c\,B\,A_0^{-1} & I \end{bmatrix} \begin{bmatrix} A_0 & \\ & S_0 \end{bmatrix} \begin{bmatrix} I & d\,A_0^{-1}\,B^T \\ & I \end{bmatrix}$$

or

$$\mathcal{P}_2 = \begin{bmatrix} I & d\,B^T\,S_0^{-1} \\ & I \end{bmatrix} \begin{bmatrix} A_0 & \\ & S_0 \end{bmatrix} \begin{bmatrix} I & \\ c\,S_0^{-1}\,B & I \end{bmatrix},$$

Benzi, Golub, Liesen (2005) *Acta Numer.*
K. (2011) *Efficient preconditioned [...] PDEs*
Brown (2012) *Intl. Symp. Para. Distr. Comp.*

A family of preconditioners

For ill-conditioned $\mathcal{L}$, use preconditioner $\mathcal{P}$, and solve iteratively

$$\mathcal{P}^{-1} \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \mathcal{P}^{-1} \begin{bmatrix} F \\ G \end{bmatrix}$$

Divide and... follow this decomposition!

$$\mathcal{P}_1 = \begin{bmatrix} I & \\ c\,B\,A_0^{-1} & I \end{bmatrix} \begin{bmatrix} A_0 & \\ & S_0 \end{bmatrix} \begin{bmatrix} I & d\,A_0^{-1}\,B^T \\ & I \end{bmatrix}$$

or

$$\mathcal{P}_2 = \begin{bmatrix} I & d\,B^T\,S_0^{-1} \\ & I \end{bmatrix} \begin{bmatrix} A_0 & \\ & S_0 \end{bmatrix} \begin{bmatrix} I & \\ c\,S_0^{-1}\,B & I \end{bmatrix},$$

Some implemented in PETSc as `PCFIELDSPLIT` type preconditioners.

Benzi, Golub, Liesen (2005) *Acta Numer.*
K. (2011) *Efficient preconditioned [...] PDEs*
Brown (2012) *Intl. Symp. Para. Distr. Comp.*

Choosing the ingredients: c, d parameters

Type	Form of $\mathcal{P}$	c	d
block-diagonal	$\begin{bmatrix} A_0 & \\ & S_0 \end{bmatrix}$	0	0
block-triangular	$\begin{bmatrix} A_0 & \\ B & -S_0 \end{bmatrix}$	1	0
block symmetric indefinite	$\begin{bmatrix} A_0 & B^T \\ B & BA_0^{-1}B^T - S_0 \end{bmatrix}$	1	1
primal-based penalty	$\begin{bmatrix} A_0 - B^T S_0^{-1} B & B^T \\ B & -S_0 \end{bmatrix}$	1	1

Choosing the ingredients: A_0, S_0 preconditioners

Let us define a block diagonal matrix and a norm

$$\mathcal{J} = \begin{bmatrix} A_0 & \\ & S_0 \end{bmatrix}, \qquad \| \begin{bmatrix} u \\ p \end{bmatrix} \|^2_{\mathcal{J}} = \|u\|^2_{A_0} + \|p\|^2_{S_0}$$
$$= u^T A_0 u + p^T S_0 p.$$

Divide and... keep balance:

stability and continuity
$$\exists_{m_0, m_1 > 0} \quad m_0 \|x\|_{\mathcal{J}} \leq \|\mathcal{L}x\|_{\mathcal{J}^{-1}} \leq m_1 \|x\|_{\mathcal{J}} \qquad \forall x,$$

mixed continuity $\exists_{b_0 > 0} \quad |p^T B u| \leq b_0 \|u\|_{A_0} \|p\|_{S_0} \qquad \forall u, \ \forall p,$

inner product definiteness $\mathcal{H} > 0$

spectral equivalence $\exists_{h_0, h_1 > 0} \quad h_0 \|x\|_{\mathcal{H}} \leq \|x\|_{\mathcal{J}} \leq h_1 \|x\|_{\mathcal{H}}, \qquad \forall x.$

Eigenvalue estimates and PCR convergence

It is known that the convergence speed of PCR iteration depends on

$$\kappa = \frac{\max |\lambda(\mathcal{P}^{-1}\mathcal{L})|}{\min |\lambda(\mathcal{P}^{-1}\mathcal{L})|}.$$

Theorem

If λ is an eigenvalue of $\mathcal{P}^{-1}\mathcal{L}$, then

$$\frac{1}{2m_0(1+b_0^2)} \leq |\lambda| \leq 2m_1(1+b_0^2).$$

This has direct implications to preconditioning Stokes equation or certain multiphysics systems of PDEs.

Klawonn (1998) *SIAM J. Sci. Comput.*
K. (2011) *Numer. Linear Algebra Appl.*
Notay (2014) *SIAM J. Matrix Anal. & Appl.*
Smears (2017) *IMA Journal of Numerical Analysis*

Summing up

Divide and... ?

Divide and...

- reconnect wisely

Divide and...

- reconnect wisely
- solve parts in parallel

Divide and...

- reconnect wisely
- solve parts in parallel
- keep balance

Divide and...

- reconnect wisely
- solve parts in parallel
- keep balance
- maintain **stability** or **approximation**

Selected active research areas

- preconditioners for nonstandard finite elements
- algorithms for new computer architectures
- communication avoiding parallel methods/preconditioners
- domain decomposition for nonlinear problems
- nonsymmetric/indefinite linear systems
- robust methods for graph Laplacians