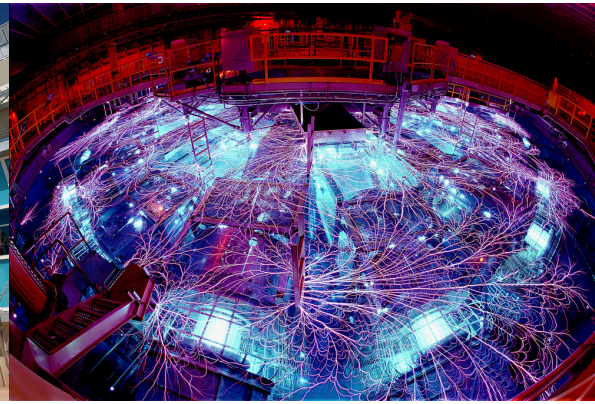
Exceptional service in the national interest

Sandia National Laboratories



Finite element methods for nonlocal problems

Theoretical background material for `PyNucleus`

Christian Glusa

github.com/sandialabs/PyNucleus

Center for Computing Research, Sandia National Laboratories

ICERM, April 12, 2024

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND NO. SAND2024-04302C

We will cover:

- Elliptic nonlocal equations
- Variational formulation
- Finite element approximation
- Matrix formats, in particular hierarchical matrices
- Linear solvers, in particular geometric multigrid

Goals

- Cover theoretical background for the hands-on examples of `PyNucleus`.
- Favor concepts over technical details.
- Questions are encouraged! Please stop me whenever I should clarify something.

Elliptic nonlocal problems

We want to numerically solve a *scalar* equation involving a *nonlocal* operator, e.g.:

$$
\begin{aligned}
-\mathcal{L}u(\boldsymbol{x}) := \text{p.v.} \int_{\Omega\cup\Omega_I} (u(\boldsymbol{x}) - u(\boldsymbol{y}))\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y} &= f(\boldsymbol{x}), && \boldsymbol{x} \in \Omega, \\
u(\boldsymbol{x}) &= g(\boldsymbol{x}), && \boldsymbol{x} \in \Omega_I.
\end{aligned}
$$

- $\Omega \subset \mathbb{R}^d$ open, bounded
- $\gamma : (\Omega \cup \Omega_I)^2 \to \mathbb{R}_{\geq 0}$ is the *kernel* of the operator
- $\Omega_I := \{\boldsymbol{y} \in \mathbb{R}^d \setminus \Omega : \gamma(\boldsymbol{x},\boldsymbol{y}) \neq 0, \quad \text{for some } \boldsymbol{x} \in \Omega\}$ is the *interaction domain*
- $u : \Omega \cup \Omega_I \to \mathbb{R}$ is the solution we want to find
- f is a forcing, g a volume condition

Nonlocality

The value of $\mathcal{L}u$ at $\boldsymbol{x}$ depends on $u(\boldsymbol{y})$ for all $\boldsymbol{y}$ in the *interaction neighborhood* $\{\boldsymbol{y} \in \Omega \cup \Omega_I : \gamma(\boldsymbol{x},\boldsymbol{z}) \neq 0\}$.
Contrast this with the PDE Laplacian $\Delta u(\boldsymbol{x}) = \sum_{j=1}^d \partial_{x_j}^2 u(\boldsymbol{x})$ which is a *local* operator.

Examples of kernel functions

1. Constant kernel, $\mathcal{X}_\bullet$ indicator function

$$\gamma(\boldsymbol{x},\boldsymbol{y}) = C_\delta \mathcal{X}_{|\boldsymbol{x}-\boldsymbol{y}|\le\delta}, \qquad \Omega_I = \text{collar of width } \delta \text{ around } \Omega$$

2. Inverse distance kernel

$$\gamma(\boldsymbol{x},\boldsymbol{y}) = \frac{\tilde{C}_\delta}{|\boldsymbol{x}-\boldsymbol{y}|}\mathcal{X}_{|\boldsymbol{x}-\boldsymbol{y}|\le\delta}, \qquad \Omega_I = \text{collar of width } \delta \text{ around } \Omega$$

3. Integral fractional Laplacian:

$$\gamma(\boldsymbol{x},\boldsymbol{y}) = \frac{C_{d,s}}{|\boldsymbol{x}-\boldsymbol{y}|^{d+2s}}, \qquad \Omega_I = \mathbb{R}^d \setminus \Omega.$$

- Normalization constants are often chosen such that the operator recovers the classical PDE Laplacian in parameter limits (e.g. $\delta \to 0$ or $s \to 1$).
- The first two kernels are *integrable* in the L^1 sense and have a finite *interaction horizon* δ.
- The fractional kernels are too singular for the integral to be taken in the usual sense: *principal value*

$$\begin{aligned}\mathcal{L}u(\boldsymbol{x}) &= \text{p.v.} \int_{\Omega\cup\Omega_I} (u(\boldsymbol{y}) - u(\boldsymbol{x}))\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y} \\ &:= \lim_{\epsilon\to 0} \int_{(\Omega\cup\Omega_I)\setminus B_\epsilon(\boldsymbol{x})} (u(\boldsymbol{y}) - u(\boldsymbol{x}))\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y}\end{aligned}$$

- These kernels just depend on $|\boldsymbol{x}-\boldsymbol{y}|$. More general kernels are allowed but often involve more work (analysis & numerics).

Elliptic nonlocal operators

Currently available in `PyNucleus`

$$\gamma(\boldsymbol{x},\boldsymbol{y}) = \phi(\boldsymbol{x},\boldsymbol{y})\,|\boldsymbol{x}-\boldsymbol{y}|_p^{-\beta(\boldsymbol{x},\boldsymbol{y})}\,\mathcal{X}_{|\boldsymbol{x}-\boldsymbol{y}|_p\leq\delta}, \qquad \boldsymbol{x},\boldsymbol{y}\in\Omega\cup\Omega_I$$

- $d \in \{1,2\}$,
- $\phi(\boldsymbol{x},\boldsymbol{y}) > 0$,
- $\delta \in (0,\infty]$,
- $p \in \{1,2,\infty\}$,
- $\beta(\boldsymbol{x},\boldsymbol{y}) \in [0, d+2)$.
- Some additional conditions are required for well-posedness

What is not available at the moment:

- 3D domains
- vector-valued kernels (peridynamics!)
- kernels that correspond to derivatives wrt parameters
- kernels for non-flat spaces

Nonlocal equations

- Nonlocal Poisson equation:

$$\begin{aligned} -\mathcal{L}u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{in } \Omega_I. \end{aligned}$$

- Nonlocal heat equation:

$$\begin{aligned} u_t - \mathcal{L}u &= f \quad \text{in } (0,T) \times \Omega, \\ u &= g \quad \text{in } (0,T) \times \Omega_I, \\ u &= u_0 \quad \text{on } \{0\} \times \Omega. \end{aligned}$$

- Combinations of local and nonlocal spatial operators, e.g. local advection and nonlocal diffusion
- fractional time derivatives
- Source control:

$$\min_f \frac{1}{2} \|u - u_d\|_{L^2}^2 + \mathcal{R}(f) \quad \text{subject to nonlocal equation}$$

- Parameter learning:

$$\min_{s,\delta,\dots} \frac{1}{2} \|u - u_d\|_{L^2}^2 + \mathcal{R}(s,\delta,\dots) \quad \text{subject to nonlocal equation}$$

Variational formulation

Take $v \in C^\infty(\Omega \cup \Omega_I)$ such that $v\big|_{\Omega_I} = 0$.

$$\int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})d\boldsymbol{x} = \int_\Omega (-\mathcal{L}u(\boldsymbol{x}))v(\boldsymbol{x})d\boldsymbol{x} = \int_\Omega \int_{\Omega\cup\Omega_I} (u(\boldsymbol{x}) - u(\boldsymbol{y}))\gamma(\boldsymbol{x},\boldsymbol{y})v(\boldsymbol{x})d\boldsymbol{y}d\boldsymbol{x}$$
$$= \iint_{(\Omega\cup\Omega_I)^2} (u(\boldsymbol{x}) - u(\boldsymbol{y}))\gamma(\boldsymbol{x},\boldsymbol{y})v(\boldsymbol{x})d\boldsymbol{y}d\boldsymbol{x}$$

We can perform "integration-by-parts" by splitting into equal parts, relabeling integration variables and using symmetry of the kernel:

$$\int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})d\boldsymbol{x} = \frac{1}{2}\iint_{(\Omega\cup\Omega_I)^2} (u(\boldsymbol{x}) - u(\boldsymbol{y}))\gamma(\boldsymbol{x},\boldsymbol{y})v(\boldsymbol{x})d\boldsymbol{y}d\boldsymbol{x}$$
$$+ \frac{1}{2}\iint_{(\Omega\cup\Omega_I)^2} (u(\boldsymbol{y}) - u(\boldsymbol{x}))\gamma(\boldsymbol{y},\boldsymbol{x})v(\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y}$$
$$= \frac{1}{2}\iint_{(\Omega\cup\Omega_I)^2} (u(\boldsymbol{x}) - u(\boldsymbol{y}))(v(\boldsymbol{x}) - v(\boldsymbol{y}))\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y}d\boldsymbol{x}$$

Note: The bilinear form looks a bit different for unsymmetric kernels.

Variational formulation

Instead of solving in the *strong form*

$$\begin{aligned} -\mathcal{L}u(\boldsymbol{x}) &= f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega_I. \end{aligned}$$

we are solving in *weak form*

$$\text{Find } u \in V \text{ such that } u\big|_{\Omega_I} = g \text{ and}$$

$$a(u, v) = L(v) \qquad \forall v \in V_0,$$

where

$$a(u, v) = \frac{1}{2} \iint_{(\Omega \cup \Omega_I)^2} (u(\boldsymbol{x}) - u(\boldsymbol{y}))(v(\boldsymbol{x}) - v(\boldsymbol{y}))\gamma(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} d\boldsymbol{x},$$

$$L(v) = \int_{\Omega} f(\boldsymbol{x}) v(\boldsymbol{x}) d\boldsymbol{x}.$$

and

$$V := \left\{ u \in L^2\left(\mathbb{R}^d\right) \mid \|u\|_V < \infty \right\}, \qquad V_0 := \{u \in V \mid u = 0 \text{ in } \Omega_I\}$$

and

$$\|u\|_V^2 = \|u\|_{L^2}^2 + \iint_{(\Omega \cup \Omega_I)^2} (u(\boldsymbol{x}) - u(\boldsymbol{y}))^2 \gamma(\boldsymbol{x}, \boldsymbol{y}).$$

V reduces to $L^2(\Omega \cup \Omega_I)$ for integrable kernels and to $H^s(\Omega \cup \Omega_I)$ for constant order fractional kernels.

Infinite horizon kernels

Integration over unbounded domains is computationally not tractable.

$$a(u,v) = \frac{1}{2}\iint_{(\Omega\cup\Omega_I)^2} (u(\boldsymbol{x}) - u(\boldsymbol{y}))(v(\boldsymbol{x}) - v(\boldsymbol{y}))\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y}d\boldsymbol{x}$$

is problematic when $\Omega_I = \mathbb{R}^d \setminus \Omega$.
Need some additional restriction / approximation, e.g.

- u is compactly supported or even $u = 0$ in Ω_I,
- u has sufficient decay so that it can be approximated.

When $u = 0$ on Ω_I, we can play tricks with the Gauss theorem:

- Write $\gamma(\boldsymbol{x},\boldsymbol{y}) = \nabla_{\boldsymbol{y}} \cdot \boldsymbol{\Gamma}(\boldsymbol{x},\boldsymbol{y})$ for some vectorial kernel $\boldsymbol{\Gamma}$.
- Split (remember $v\big|_{\mathbb{R}^d\setminus\Omega} = 0$)

$$\begin{aligned}
a(u,v) &= \frac{1}{2}\iint_{\Omega^2} (u(\boldsymbol{x}) - u(\boldsymbol{y}))(v(\boldsymbol{x}) - v(\boldsymbol{y}))\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y}d\boldsymbol{x} \\
&\quad + \int_\Omega \int_{\mathbb{R}^d\setminus\Omega} (u(\boldsymbol{x}) - u(\boldsymbol{y}))(v(\boldsymbol{x}) - v(\boldsymbol{y}))\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y}d\boldsymbol{x} \\
&= \frac{1}{2}\iint_{\Omega^2} (u(\boldsymbol{x}) - u(\boldsymbol{y}))(v(\boldsymbol{x}) - v(\boldsymbol{y}))\gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y}d\boldsymbol{x} \\
&\quad + \int_\Omega u(\boldsymbol{x})v(\boldsymbol{x}) \underbrace{\int_{\mathbb{R}^d\setminus\Omega} \gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y}}_{\int_{\partial\Omega} \boldsymbol{n}_{\boldsymbol{y}}\cdot\Gamma(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y}} d\boldsymbol{x}
\end{aligned}$$

Finite element approximation

- V is an infinite dimensional space $\rightarrow$ cannot be represented on a computer.
- Take a finite dimensional sub-space $V_h \subset V$ and solve

$$\text{Find } u_h \in V_h \text{ such that } u_h\big|_{\Omega_I} = g_h \text{ and}$$
$$a(u_h, v_h) = L(v_h) \qquad \forall v_h \in V_{h,0},$$

- Let $\mathcal{T}_h$ be a simplical mesh for $\Omega \cup \Omega_I$.
- Set $V_h := \{v \in C^q(\Omega \cup \Omega_I) \mid v\big|_K \in \mathbb{P}_k(K)\ \forall K \in \mathcal{T}_h\}$.
  Here: $\mathbb{P}_k(K)$ are the polynomials of degree up to k on K.
- In practice:
  - continuous piecewise linears ($q = 0$ and $k = 1$) or
  - piecewise constant discontinuous space $q = -1$ and $k = 0$.
- Basis functions $\phi_i(\boldsymbol{x})$ span the space V_h, obtain linear system

$$\begin{pmatrix} A_{\Omega,\Omega} & A_{\Omega,\Omega_I} \\ & I_{\Omega_I} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_\Omega \\ \boldsymbol{u}_{\Omega_I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}_\Omega \\ \boldsymbol{g}_{\Omega_I} \end{pmatrix} \quad \Rightarrow \quad A_{\Omega,\Omega}\boldsymbol{u}_\Omega = \boldsymbol{L}_\Omega - A_{\Omega,\Omega_I}\boldsymbol{g}_{\Omega_I}$$

Here

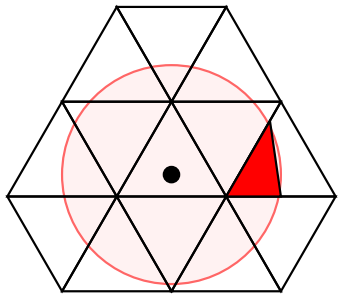
$$A_{\Omega,\Omega} = \{a(\phi_i, \phi_j)\}_{i,j\in\Omega} \qquad A_{\Omega,\Omega_I} = \{a(\phi_i, \phi_j)\}_{i\in\Omega, j\in\Omega_I},$$
$$u_h(\boldsymbol{x}) = \sum_i \boldsymbol{u}_{\Omega,i}\phi_i(\boldsymbol{x}) + \sum_i \boldsymbol{u}_{\Omega_I,i}\phi_i(\boldsymbol{x})$$

Finite element approximation

- Sub-assembly loop:

$$a(\phi_i, \phi_j) = \frac{1}{2} \sum_{K, \tilde{K} \in \mathcal{T}_h^2} \sum \iint_{K \times \tilde{K}} \left(\phi_i\left(\boldsymbol{x}\right) - \phi_i\left(\boldsymbol{y}\right)\right)\left(\phi_j\left(\boldsymbol{x}\right) - \phi_j\left(\boldsymbol{y}\right)\right) \gamma(\boldsymbol{x}, \boldsymbol{y})\, d\boldsymbol{y}\, d\boldsymbol{x}$$

- No closed form for local stiffness matrix
  $\rightarrow$ need to use numerical quadrature to evaluate double integrals
- Avoid integration across discontinuities for finite horizon δ or jumps in kernels: approximate with sub-simplices, $\mathcal{O}(h_K^2)$ error[1]



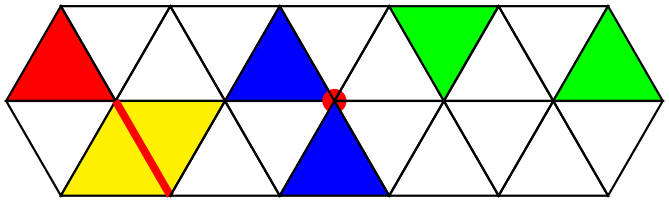
[1] Marta D'Elia, Max Gunzburger, and Christian Vollmann. "A cookbook for approximating Euclidean balls and for quadrature rules in finite element methods for nonlocal problems". In: *Mathematical Models and Methods in Applied Sciences* 31.08 (2021), pp. 1505–1567.

Quadrature

- In subassembly procedure, use quadrature to evaluate element pair contributions:

$$a^{K\times\tilde{K}}(\phi_i,\phi_j) = \frac{1}{2}\int_K d\boldsymbol{x}\int_{\tilde{K}} d\boldsymbol{y}\,(\phi_i(\boldsymbol{x})-\phi_i(\boldsymbol{y}))\,(\phi_j(\boldsymbol{x})-\phi_j(\boldsymbol{y}))\,\gamma(\boldsymbol{x},\boldsymbol{y})$$

- Treatment for element pairs $K\cap\tilde{K}\neq\emptyset$, containing the singularity at $\boldsymbol{x}=\boldsymbol{y}$:



  - split $K\times\tilde{K}$ into sub-simplices,
  - Duffy transform onto a hypercube, with Jacobian canceling the singularity.
  - Choose quadrature order so that quadrature error $\leq$ discretization error[2]:
    - $|\log h_K|$ if the elements coincide (*red*),
    - $|\log h_K|^2$ if the elements share only an edge (*yellow*),
    - $|\log h_K|^3$ if the elements share only a vertex (*blue*),
    - $|\log h_K|^4$ if the elements are "near neighbours" (*green*), and
    - C if the elements are well separated.

- `PyNucleus` tries to handle the selection of quadrature rules automatically.

[2] Mark Ainsworth and Christian Glusa. "Aspects of an adaptive finite element method for the fractional Laplacian: A priori and a posteriori error estimates, efficient implementation and multigrid solver". In: *Computer Methods in Applied Mechanics and Engineering* (2017).

FE convergence

Rate of convergence depends on

- Polynomial order of the FE space
- Mesh resolution of salient features
- Regularity of the solution
  - kernel function
  - regularity of the data (forcing, boundary data)
  - domain regularity
- Compared with classical PDEs:
  - Rates are generally lower than for classical PDEs
  - Major difference: nice domain + nice data $\not\Rightarrow$ high regularity

    (But one can obviously construct examples where higher rates of convergence are obtained.)

Choice of mesh

Fractional problems often display steep gradients near $\partial\Omega$.

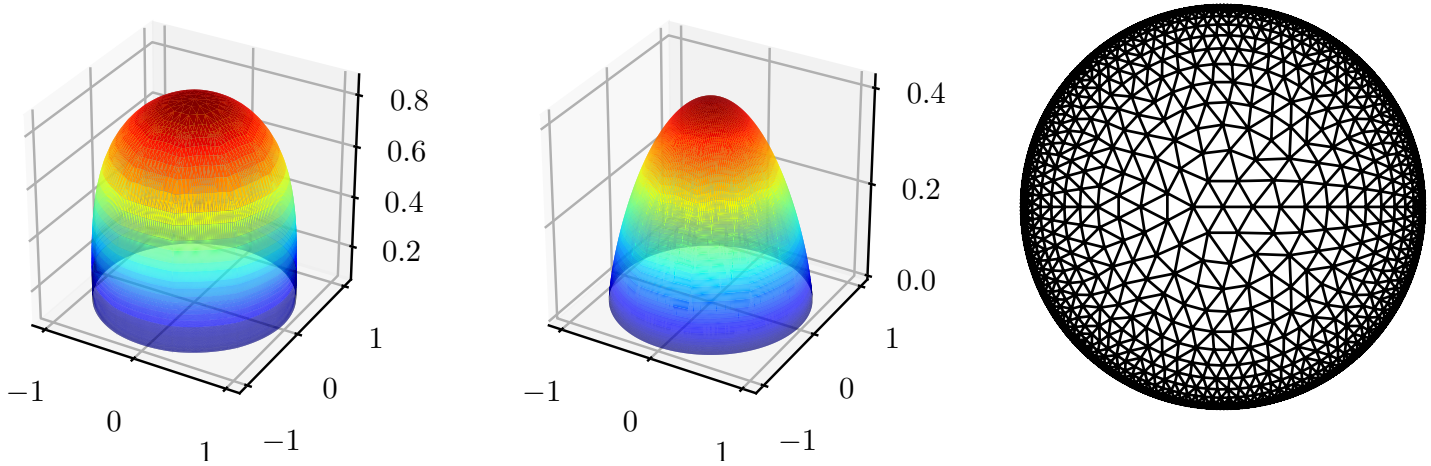
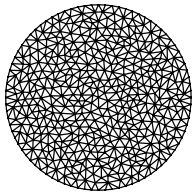
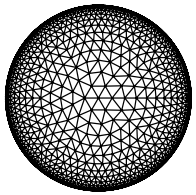


Figure: Solutions $u = C(1 - |\boldsymbol{x}|^2)^s \sim \text{dist}\,(\boldsymbol{x}, \partial\Omega)^s$ corresponding to the constant right-hand side $f = 1$ for $s = 0.25$ and for $s = 0.75$, $\delta = \infty$.

Higher rates of convergence can be achieved by resolving this behavior via graded or adaptively refined meshes.

FEM convergence rates for fractional kernels, $\mathbf{P}_1$ elements

			$V = \widetilde{H}^s(\Omega)$		$L^2(\Omega)$	
$d = 1$		quasi-uniform	$h^{1/2}$	$\frac{1}{n^{1/2}}$	$h^{(1/2+s)\wedge 1}$	$\frac{1}{n^{(1/2+s)\wedge 1}}$
		graded / adaptive		$\frac{1}{n^{2-s}}$		$\frac{1}{n^2}$
$d \geq 2$		quasi-uniform	$h^{1/2}$	$\frac{1}{n^{1/(2d)}}$	$h^{(1/2+s)\wedge 1}$	$\frac{1}{n^{(1/2+s)/d)\wedge(1/d)}}$
		graded / adaptive		$\frac{1}{n^{1/d}}$		$\frac{1}{n^{(1+s)/d}}$

mesh size h, number of unknowns n, spatial dimension d, fractional order s

Representation of linear operators

After FEM discretization:

$$A\boldsymbol{u} = \boldsymbol{L}, \qquad A \in \mathbb{R}^{n \times n}$$

Depending on δ and h:

- Assembly and solve has complexity and memory usage
  - $\mathcal{O}\left(n(\frac{\delta}{h})^d\right)$ for $\delta < \infty$ and
  - $\mathcal{O}(n^2)$ for $\delta = \infty$.
- For small $\frac{\delta}{h}$ we can use a sparse matrix (CSR format)
- For large $\frac{\delta}{h}$, we end up with an (almost) dense matrix.

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$

CSR format:

$$\texttt{rowptr} = (\,0 \quad 2 \quad 4 \quad 5\,)$$

$$\texttt{indices} = (\,0 \quad 1 \quad 0 \quad 1 \quad 2\,)$$

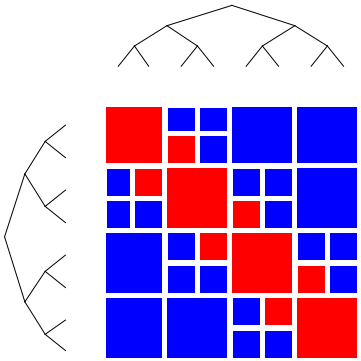
$$\texttt{values} = (\,1 \quad 2 \quad 3 \quad 4 \quad 5\,)$$

Better approach

Panel clustering / Fast Multipole Method / hierarchical matrix approximation

- Split operator into near and far interactions
- Directly assembly near interactions
- Low-rank approximation of far interactions
- Keep approximation error below discretization error to preserve FE convergence.

Hierarchical matrices: Admissible sub-blocks



Tasks:

1. Choose sub-blocks to be compressed.
2. Construct low-rank approximations.

Build tree of clusters of DoFs.

- root contains all unknowns
- subdivision based on DoF coordinates
- distributed computations: first level given by MPI distribution of unknowns

Admissibility criterion:

- Cluster pairs (P, Q) that are sufficiently separated compared to their sizes are *admissible* for compression:

$$\eta \,\mathrm{dist}(P, Q) \leq \max\{\mathrm{diam}(P), \mathrm{diam}(Q)\}, \qquad \eta > 0 \text{ fixed parameter}$$

- Matrix entries that are not admissible are assembled directly into a sparse near-field matrix A_{near}.

Hierarchical matrices: low-rank approximation

- Splitting of operator into sub-blocks based on admissibility

$$A = A_{\text{near}} + A_{\text{far}} = A_{\text{near}} + \sum_{\text{blocks}(P,Q)} A_{P,Q}$$

- $\mathcal{H}$-matrix approximation

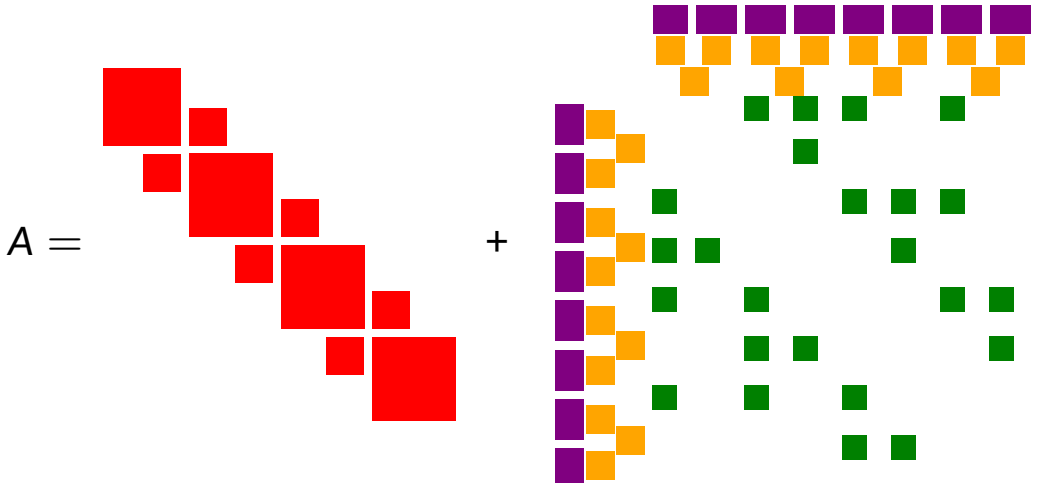
$$A_{P,Q} \approx U_P \Gamma_{P,Q} U_Q^T \qquad \text{(low-rank approximation)}$$

  I use Chebyshev interpolation, but other techniques are possible, e.g. Adaptive Cross Approximation (ACA).

- $\mathcal{H}^2$-matrices
  Using hierarchical nestedness of clusters, can express

$$U_P = \sum_{Q \text{ child of } P} U_Q T_{Q,P}$$

Matrix-vector product with an $\mathcal{H}^2$-matrix



Steps:

- Matvec with sparse near-field matrix
- Upward recursion
- Cluster-cluster interaction
- Downward recursion

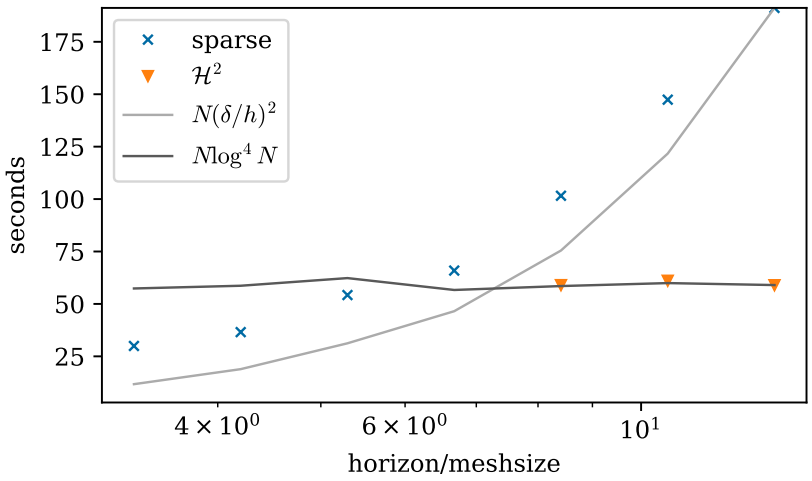
$\mathcal{H}^2$-matrix approximation

Finite element assembly and matrix-vector product and in $\mathcal{O}\left(n \log^{2d} n\right)$ operations & memory. Approximation error same order as discretization error.

Hierarchical matrices for finite horizon kernels

- Regular CSR sparse matrices are efficient (complexity, memory) for small horizon $\delta \sim h$.
- $\mathcal{H}^2$-matrices are efficient for $\delta = \infty$.
- Low-rank approximation relies on smoothness of the kernel
  $\rightarrow$ need to fully assemble entries near $\partial B_\delta(\boldsymbol{x})$.

At what ratio δ/h do $\mathcal{H}^2$-matrices become more efficient than sparse matrices?



- $\Omega \subset \mathbb{R}^2$, $\gamma(\boldsymbol{x}, \boldsymbol{y}) = c_{d,\delta} \mathcal{X}_{|\boldsymbol{x}-\boldsymbol{y}| \leq \delta}$
- Break-even:
  - 1D: $\delta/h \sim 100 - 200$
  - 2D: $\delta/h \sim 5 - 10$
- Break-even depends on:
  - Cost of quadrature
  - Lots of implementation details

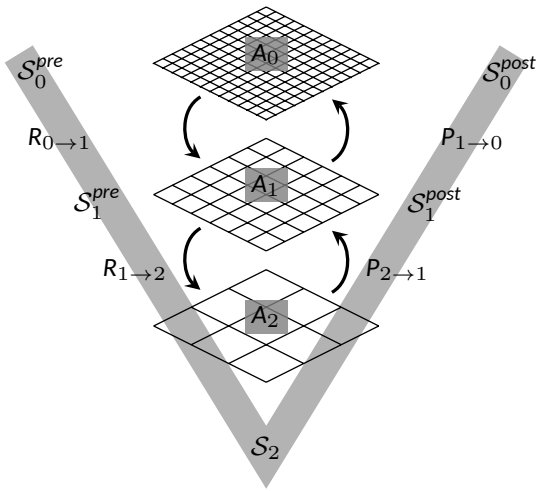
Conditioning and scalable solvers

- Direct solvers for hierarchical matrices have quasi-optimal $\mathcal{O}(n \log^\alpha n)$ scaling
  BUT: can be tricky to implement, especially in distributed memory.
- $\mathcal{O}(n \log^\alpha n)$ matrix-vector product in all cases $\rightarrow$ use iterative solvers
- Number of iterations required depends on condition number $\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|_2 \left\|\boldsymbol{A}^{-1}\right\|_2$
  - Steady-state:
    - Integrable kernel, δ finite[8]: $\kappa(\boldsymbol{A}) \sim \delta^{-2}$
    - Fractional kernel, $\delta = \infty$: $\kappa(\boldsymbol{A}) \sim h^{-2s} \sim n^{2s/d}$
    - Fractional kernel, $\delta \leq \delta_0$: $\kappa(\boldsymbol{A}) \sim \delta^{2s-2} h^{-2s} \sim \delta^{2s-2} n^{2s/d}$
  - Time-dependent:
    - $\kappa(\boldsymbol{M} + \Delta t \boldsymbol{A}) \sim 1 + \Delta t\, \kappa(\boldsymbol{A})$
    - Depending on time-stepper and CFL condition, this is well-conditioned for small s, large δ.
- Ill-conditioned cases $\kappa(\boldsymbol{A}) \gg 1$ need a *preconditioner*:

$$\boldsymbol{PAu} = \boldsymbol{PL}$$

  such that $\kappa(\boldsymbol{PA}) \ll \kappa(\boldsymbol{A})$ and $\text{cost}(\boldsymbol{P}) \sim \text{cost}(\boldsymbol{A})$.
- A solver (+preconditioner) is *scalable* if the number of iterations required is independent of the number of unknowns, $\kappa \sim 1$.
- Scalable solver options:
  - Krylov method + multigrid,
  - Krylov method + domain decomposition,
  - Unpreconditioned Krylov method (when $\kappa(\boldsymbol{A}) \sim 1$).

Geometric multigrid in a nutshell



User specifies:

- Operators A_ℓ, assembled on hierarchy of nested meshes
- Transfer operators: prolongations $P_{\ell+1\to\ell}$, restrictions $R_{\ell\to\ell+1} = P^T_{\ell+1\to\ell}$,
- Smoothers $\mathcal{S}_\ell^{\text{pre/post}}$ (Jacobi, Gauss-Seidel, Chebyshev, ...)
- Coarse solver $\mathcal{S}_L$ (typically direct solver)

How does multigrid work?

- On each level:
  - smoother reduces high frequency error
  - low frequency error is transferred to next coarser levels
- High/low frequency splitting depends on mesh
  $\to$ each error frequency gets strongly reduced on one of the levels.

Geometric multigrid for nonlocal equations

- Hierarchy of meshes from uniform or adaptive refinement
- Restriction / prolongation given by nesting of FE spaces
- Assembly into hierarchical or CSR matrix format on every level
- Smoothers:
  - $\mathcal{H}$-matrices: Jacobi, Chebyshev
  - CSR matrices: no limitations
- Coarse solve: convert to dense or CSR matrix

Topics we did not cover:

- time discretization (classical and fractional)
- adaptive mesh refinement
- local-nonlocal coupling
- inverse problems
- domain decomposition solvers
- algebraic multigrid

Getting started with PyNucleus

- Source repository: github.com/sandialabs/PyNucleus
- Documentation: sandialabs.github.io/PyNucleus

Installing & running PyNucleus:

Option 1 Directly from the browser:
No installation, not sure about stability of the cloud service
mybinder.org/v2/gh/sandialabs/PyNucleus/binder

Option 2 Locally in a container
Needs to download container image ($\sim$ 2GB!)

- Install `docker` and `docker-compose`, or `podman` and `podman-compose`.
- Download `compose.yaml` from PyNucleus repository to an **EMPTY** folder.
- `podman-compose run pynucleus`
- More detailed instructions sandialabs.github.io/PyNucleus/installation.html#pre-built-container-image

Option 3 Spack
Builds all dependencies, first installation can be slow
`spack install py-pynucleus`

Option 4 Manual build against pre-installed dependencies
Could be complicated and is system dependent

Recommendation for the hands-on session

Use option 1. If the cloud service goes down try option 2.

Operator interpolation[3,4]

Parameter learning problem requires operators for different values of s and δ.

- Piecewise Chebyshev interpolation in s:

Lemma

Let $s \in [s_{\min}, s_{\max}] \subset (0,1)$, $\delta \in (0,\infty)$, and let $\eta > 0$. Assume that $u(s) \in H_\Omega^{s+1/2-}(\mathbb{R}^n)$, $v \in H_\Omega^s(\mathbb{R}^n)$. There exists a partition of $[s_{\min}, s_{\max}]$ into sub-intervals $\mathcal{S}_k$ and interpolation orders M_k such that the piecewise Chebyshev interpolant $\tilde{a}(\cdot,\cdot;s,\delta)$ satisfies:

$$|a(u(s),v;s,\delta) - \tilde{a}(u(s),v;s,\delta)| \leq \eta \, \|u(s)\|_{H_\Omega^{\bar{s}_2(s)}(\mathbb{R}^n)} \, \|v\|_{H_\Omega^s(\mathbb{R}^n)} \, ,$$

and the total number of interpolation nodes satisfies

$$\sum_{k=1}^{K} (M_k + 1) \leq C \, |\log \eta| \, .$$

The constant C depends on δ and $s_{\max}$.

- Combined with hierarchical matrix approach: $\mathcal{O}(n \log^{2d+1} n)$ complexity & memory.
- Also allows to evaluate derivatives wrt s.
- Assembly for different values of δ is achieved by splitting the kernel into infinite horizon, singular part, and δ-dependent regular part.

[3] Olena Burkovska and Max Gunzburger. "Affine approximation of parametrized kernels and model order reduction for nonlocal and fractional Laplace models". In: *SIAM Journal on Numerical Analysis* 58.3 (2020), pp. 1469–1494.

[4] Olena Burkovska, Christian Glusa, and Marta D'Elia. "An optimization-based approach to parameter learning for fractional type nonlocal models". In: *Computers & Mathematics with Applications* (2021).