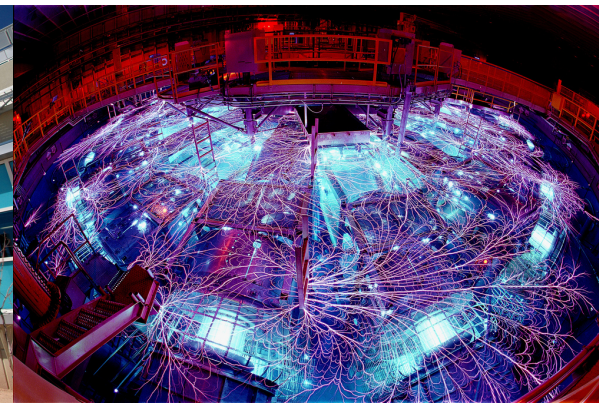
Exceptional service in the national interest

Sandia National Laboratories

PyNucleus – A FEM code for nonlocal problems

https://github.com/sandialabs/PyNucleus

Christian Glusa

caglusa@sandia.gov

Center for Computing Research, Sandia National Laboratories

December 2, 2021

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. SAND-XXXXXXXX

boratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LL
Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE

Elliptic nonlocal operators

Let $\delta \in (0, \infty]$ be the *horizon*, $\Omega \subset \mathbb{R}^n$ a bounded open domain, define the *interaction domain*

$$\Omega_I := \{\mathbf{y} \in \mathbb{R}^n \setminus \Omega : |\mathbf{x} - \mathbf{y}| \leq \delta, \quad \text{for } \mathbf{x} \in \Omega\}.$$

We want to numerically solve equations involving the nonlocal operator

$$\mathcal{L}u(\mathbf{x}) = \text{p.v.} \int_{\Omega \cup \Omega_I} (u(\mathbf{y}) - u(\mathbf{x}))\gamma(\mathbf{x}, \mathbf{y}) d\mathbf{y}, \qquad \mathbf{x} \in \Omega,$$

with

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

- Examples:
  - Integral fractional Laplacian: $\phi \sim$ const, $\beta = n + 2s$, $s \in (0, 1)$, $\delta = \infty$
  - Tempered fractional Laplacian: $\phi(\mathbf{x}, \mathbf{y}) \sim \exp(-\lambda|\mathbf{x} - \mathbf{y}|)$
  - Truncated fractional Laplacian: δ finite
  - Variable order fractional Laplacians with varying coefficient: $\beta(\mathbf{x}, \mathbf{y}) = n + 2s(\mathbf{x}, \mathbf{y})$, $\phi(\mathbf{x}, \mathbf{y}) > 0$
  - Integrable kernels: constant kernel ($\beta = 0$), "peridynamic" kernel ($\beta = 1$)
- Assumptions (for now):
  - γ is symmetric.
  - Interaction domain is defined wrt 2-norm.

Features of PyNucleus

Sandia National Laboratories

- Written in Python, lots of optimized kernels compiled to C via Cython.
- Compatible with NumPy/SciPy
- Organized as a collection of sub-packages.
- Simplical meshes in 1D, 2D, (3D); uniform refinement with boundary snapping options
- Mesh (re)partitioning using (PAR)METIS
- Finite Element discretizations: discontinuous P_0, continuous P_1, P_2, P_3
- Assembly of local differential operators
- Lots of solvers (direct, Krylov, simple preconditioners),
  and in particular geometric multigrid
- MPI distributed computations via mpi4py
- Assembly of the nonlocal operators in weak form:

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

  into
  - CSR sparse matrix ($\delta \sim h$),
  - Dense matrix ($\delta \gg h$): at least $\mathcal{O}(N^2)$ complexity and memory
  - $\mathcal{H}^2$ hierarchical matrix ($\delta \gg h$; only tested for fractional kernels)
    Quasi-optimal assembly & matrix-vector product: $\mathcal{O}(N \log^{2n} N)$
- For fractional kernels: quadrature orders are tuned for optimal convergence.

Code example

Sandia National Laboratories

```
from PyNucleus import (kernelFactory, nonlocalMeshFactory, dofmapFactory,
                       functionFactory, HOMOGENEOUS_DIRICHLET, solverFactory)

# Infinite horizon fractional kernel
kernel = kernelFactory('fractional', dim=2, s=0.75, horizon=inf)

# Mesh for unit disc, no interaction domain for homogeneous Dirichlet
mesh, _ = nonlocalMeshFactory('disc', kernel=kernel,
                              boundaryCondition=HOMOGENEOUS_DIRICHLET,
                              hTarget=0.15)

dm = dofmapFactory('P1', mesh)                          # P1 finite elements
f = functionFactory('constant', 1.)                     # constant forcing
b = dm.assembleRHS(rhs)                                 # ∫_Ω f φ_i
A = dm.assembleNonlocal(kernel, matrixFormat='h2')      # a(φ_i, φ_j), hierarchical
u = dm.zeros()                                          # solution vector

# solve with diagonally preconditioned CG
solver = solverFactory('cg-jacobi', A=A, setup=True)
solver(b, u)
u.plot()
```

- The documentation contains two examples of how to setup and solve local and nonlocal problems with a lot more explanations.
- The repository contains several drivers that demonstrate some of the code capabilities.

Applications

Sandia National Laboratories

We have used PyNucleus to solve:

- Poisson problems (also using mesh adaptivity),
- diffusion problems, reaction-diffusion systems (e.g. Brusselator),
- Helmholtz problems,
- Cahn-Hilliard, Allen-Cahn equations,
- optimal control problems (source control, learning of kernel parameters),
- nonlocal interface problems.

When something does not work as expected or is confusing:

PyNucleus is a research code under active development.
Some features are quite mature, others are more experimental.
Open an issue on Github, or send me an email: `caglusa@sandia.gov`

Thanks for listening!

Code:
https://github.com/sandialabs/PyNucleus

Documentation and examples:
https://sandialabs.github.io/PyNucleus

Funding:

PyNucleus' development is funded through the MATNIP project (PI: Marta D'Elia) of the LDRD program at Sandia National Laboratories.
The MATNIP project develops for the first time a rigorous nonlocal interface theory based on physical principles that is consistent with the classical theory of partial differential equations when the nonlocality vanishes and is mathematically well-posed. This will improve the predictive capability of nonlocal models and increase their usability at Sandia and, more in general, in the computational-science and engineering community. Furthermore, this theory will provide the groundwork for the development of nonlocal solvers, reducing the burden of prohibitively expensive computations.

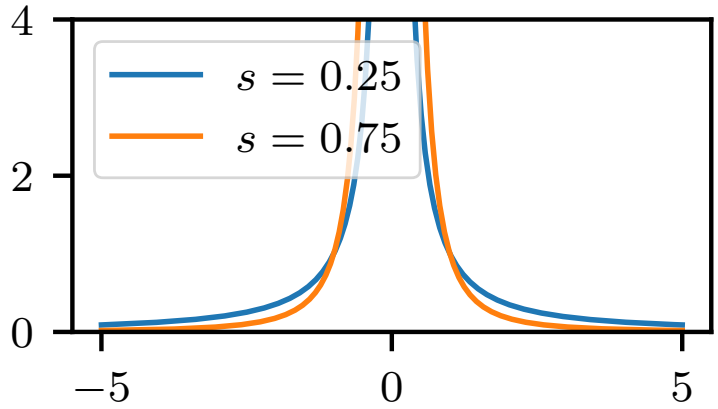


Figure: Kernel functions in $d = 1$ dimensions.

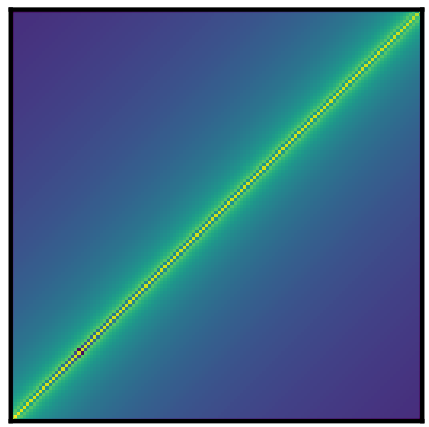


Figure: Magnitude of matrix entries.

Proposed approach

Panel clustering / Fast Multipole Method / $\mathcal{H}$-Matrix approximation

Cluster method: admissible clusters

Build tree of clusters of DoFs.

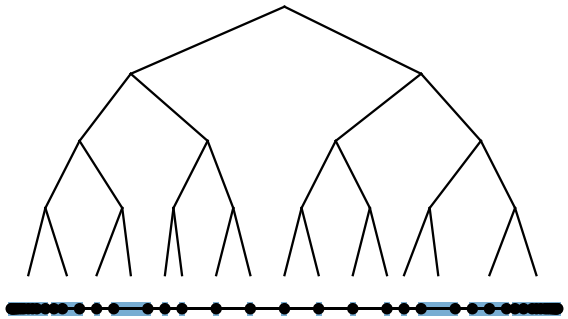


Figure: Cluster tree in $d = 1$ dimensions.

Approximate *admissible* cluster pairs $P \times Q$, i.e. when they are sufficiently separated compared to their size.

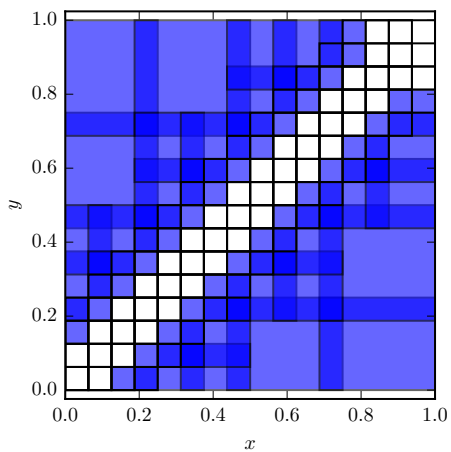


Figure: Elements of admissible cluster pairs in blue.Overlaps in dark blue.

Cluster method: far-field approximation

(P, Q) admissible cluster pair; ϕ, ψ functions with supp $\phi \subset P$, supp $\psi \subset Q$.

$$a(\phi, \psi) = -\int_\Omega \int_\Omega \gamma(\mathbf{x}, \mathbf{y}) \phi(\mathbf{x}) \psi(\mathbf{y}) .$$

Let ξ_α^P be Chebyshev nodes in P and L_α^P the associated Lagrange polynomials. Then

$$k(\mathbf{x}, \mathbf{y}) \approx \sum_{\alpha,\beta=1}^{m^d} \gamma\left(\xi_\alpha^P, \xi_\beta^Q\right) L_\alpha^P(\mathbf{x}) L_\beta^Q(\mathbf{y}), \quad \mathbf{x} \in P, \ \mathbf{y} \in Q.$$

and

$$a(\phi, \psi) \approx -\sum_{\alpha,\beta=1}^{m^d} \gamma\left(\xi_\alpha^P, \xi_\beta^Q\right) \int_P \phi(\mathbf{x}) L_\alpha^P(\mathbf{x}) \, d\mathbf{x} \int_Q \psi(\mathbf{y}) L_\beta^Q(\mathbf{y}) \, d\mathbf{y}.$$

- Decouples ϕ and ψ, "sparsifies" off-diagonal matrix blocks.
- Works for *all* pairs (ϕ, ψ) with support in $P \times Q$.

Cluster method: shift coefficients

For $\mathbf{x}$ in a sub-cluster P of Q, i.e. $P \subset Q$,

$$L_\alpha^Q(\mathbf{x}) = \sum_{\beta=1}^{m^d} L_\alpha^P\left(\xi_\beta^Q\right) L_\beta^P(\mathbf{x}).$$

(That's just polynomial interaction.)
Hence, suffices to compute

$$\int_P \phi(\mathbf{x})\, L_\alpha^P(\mathbf{x})\ d\mathbf{x}$$

only for $P = \operatorname{supp}\phi$ and all basis functions ϕ.
Need to compute

- Far-field coefficients $\int_P \phi(\mathbf{x})\, L_\alpha^P(\mathbf{x})\ d\mathbf{x}$,
- shift coefficients $L_\alpha^P\left(\xi_\beta^Q\right)$,
- kernel approximations $\gamma\left(\xi_\alpha^P, \xi_\beta^Q\right)$.
  (Our special choice of k played no major role!)
- near-field, i.e. all entries that are not in admissible cluster pairs.

FE assembly and matrix-vector product in $\mathcal{O}\left(N \log^{2d} N\right)$ operations.

(Rules for the choice of polynomial order m of the Chebyshev interpolation and the necessary quadrature order in [**SauterSchwab2010_BoundaryElementMethods**, **AinsworthGlusa2018**, **AinsworthGlusa2017**].)

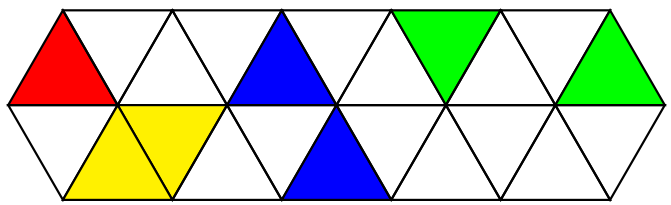
Gauss rules for matrix assembly

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

$$a^{K\times\tilde{K}}(\phi_i,\phi_j)=\frac{1}{2}\iint_{K\times\tilde{K}}\left(\phi_i\left(\mathbf{x}\right)-\phi_i\left(\mathbf{y}\right)\right)\left(\phi_j\left(\mathbf{x}\right)-\phi_j\left(\mathbf{y}\right)\right)\gamma(\mathbf{x},\mathbf{y}).$$

Since $k(\mathbf{x},\mathbf{y})\sim|\mathbf{x}-\mathbf{y}|^{-d-2s}$, special treatment for element pairs $K\cap\tilde{K}\neq\emptyset$:

- 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 does not exceed discretization error of the FE approximation:

- $|\log h_K|$ 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.