

Bempp - What's next?

Timo Betcke

t.betcke@ucl.ac.uk

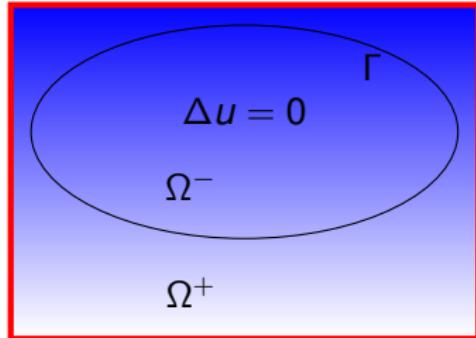
University College London

Joint with: Sri Kailasa,
Maria Ignacia Fierro Pic-
cardo, Matthew Scroggs,
Lorena Barba, and many
more.



Supported by EPSRC Grants EP/W007460/1, EP/W026260/1

A primer on boundary integral equations



Green's fct. for Laplace equation

$$g(x, y) = \frac{1}{4\pi|x - y|}$$

Representation formula

$$u(x) = \int_{\Gamma} g(x, y) \gamma_1 u(y) ds(y) - \int_{\Gamma} \frac{\partial g(x, y)}{\partial n(y)} \gamma_0 u(y) ds(y), \quad x \in \Omega^-$$

Neumann trace Dirichlet trace

Trace + Representation Formula
~~~~~ Calderón identities

Take the Dirichlet and Neumann trace to obtain



$$V = [\gamma_0 u \quad \gamma_1 u]^T$$

$$A = \begin{bmatrix} -K & V \\ W & K' \end{bmatrix}$$

Interior Problem

$$V = \left[ \frac{1}{2} I + A \right] V$$

Exterior Problem

$$V = \left[ \frac{1}{2} I - A \right] V$$

- Projection Property

$$\left[ \frac{1}{2} I \pm A \right]^2 V = \left[ \frac{1}{2} I \pm A \right] V$$

for any possible pair of boundary data  $V$ .

- Regularisation Property

$$(2A)^2 = I$$

# How to discretise a boundary integral operator?



Consider single-layer operator  $V : H^{-1/2}(\Gamma) \rightarrow H^{1/2}(\Gamma)$

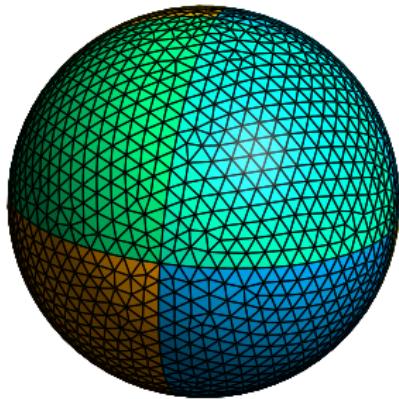
$$[V\phi](x) = \int_{\Gamma} g(x, y)\phi(y)ds(y).$$

Introduce (weak) variational form

$$\langle V\phi, \psi \rangle_{\Gamma} = \int_{\Gamma} \psi(x) \int_{\Gamma} g(x, y)\phi(y)ds(y)ds(x), \quad \psi \in H^{-1/2}(\Gamma).$$

Compute matrix elements

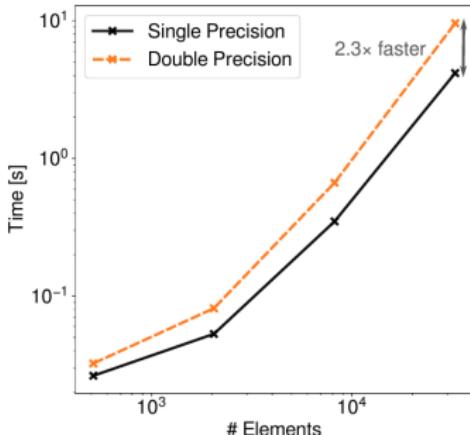
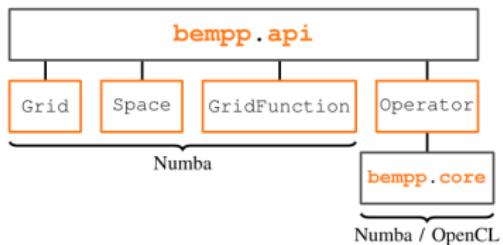
$$v_{ij} = \int_{\tau_i} \int_{\tau_j} g(x, y)ds(y)ds(x)$$



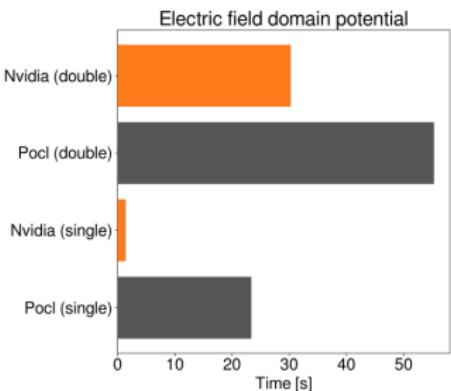
- Originally developed as C++ library, later moved to Python (Bempp-cl)
- Support for Laplace, Helmholtz, Maxwell operators
- FEM/BEM coupling with FEnics
- Old C++ Bempp has support for Hierarchical-Matrix acceleration
- New Bempp supports Fast Multipole Method

- Successor of the Bempp boundary element library.
- Fully developed using Python with fast OpenCL kernels for computational routines.
- Galerkin discretisation of operators for Laplace, Helmholtz, and Maxwell problems.
- Compute kernels can be run on CPU or offloaded to GPU.
- Core routines optimised for fast dense assembly of operators.
- Library provides coupling interfaces to Exafmm, can be easily adapted to other fast particle summation libraries.

# Characteristics of Bempp-cl



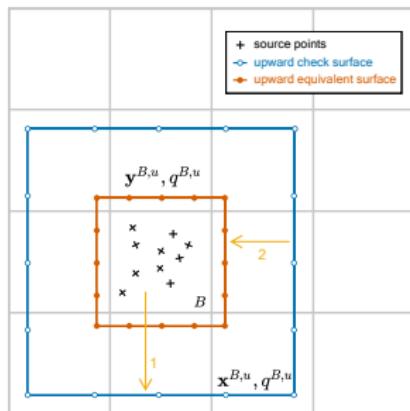
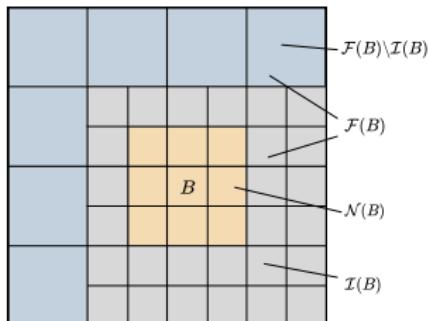
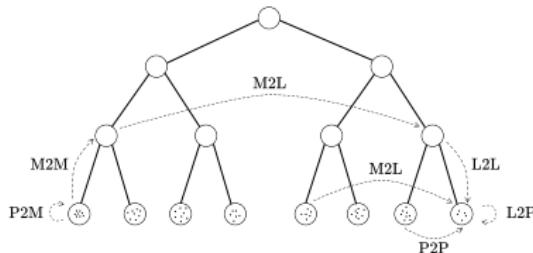
- Top: Structure of Bempp-cl
- Top-Right: AVX acceleration in Bempp-cl
- Bottom-Right: Offloading of a domain potential operator



# FMM Acceleration with Exafmm-t



- Kernel-Independent FMM library.
- Written in C++, Provides Python Interface.
- Highly efficient at higher accuracies.



Let  $A$  be the matrix representation of a Galerkin discretised boundary integral operator. Let  $x$  be any vector. Reformulate  $Ax$  as

$$Ax = P_{test}^T(G - C)P_{dom}x + Sx$$

- $P_{test}$  and  $P_{dom}$  highly sparse matrices that map function space coefficients to particle weights at quadrature points.
- $G$  is black-box operator evaluating particle sums over weights across all quadrature points across all elements.
- $C$  is sparse matrix that contains the particle interactions between neighboring triangles.
- $S$  is sparse matrix containing all singular Galerkin integrals in adjacent elements.

# Applications

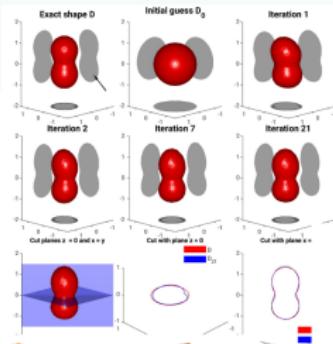
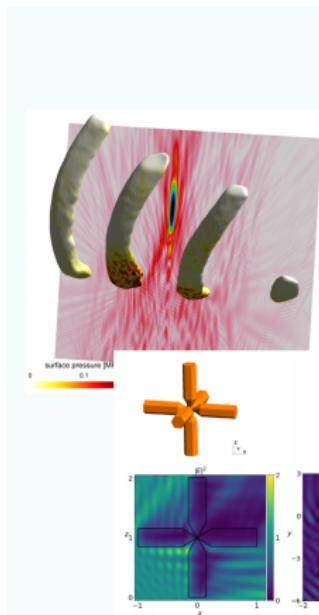


FIGURE 17. Example 3.3: Mesh  $T_\ell$  and distribution of the error estimator  $\eta_\ell^2$  for  $\ell = 10$  (left) with 360 elements and  $\ell = 15$  (right) with 1524 elements.

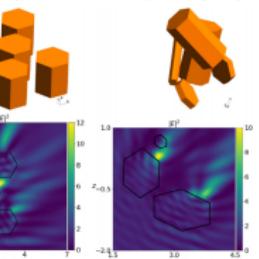
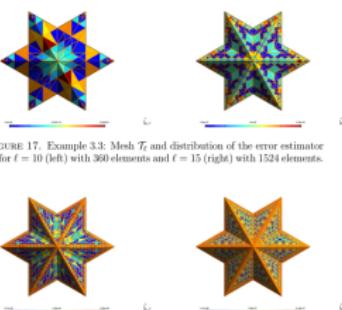


FIGURE 18. Example 3.3: Mesh  $T_\ell$  and distribution of the error estimator  $\eta_\ell^2$  for  $\ell = 20$  (left) with 5754 elements and  $\ell = 25$  (right) with 20304 elements.



# Poisson-Boltzmann for virus simulations

Solve Poisson-Boltzmann equation

$$\Delta\phi_1 = \frac{1}{\epsilon_1} \sum_k q_k \delta(\mathbf{r}, \mathbf{r}_k) \text{ in } \Omega_1$$

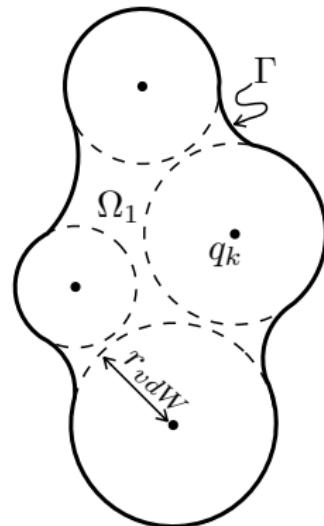
$$(\Delta - \kappa^2)\phi_2 = 0 \text{ in } \Omega_2$$

Interface conditions on  $\Gamma$ :

$$\phi_1 = \phi_2,$$

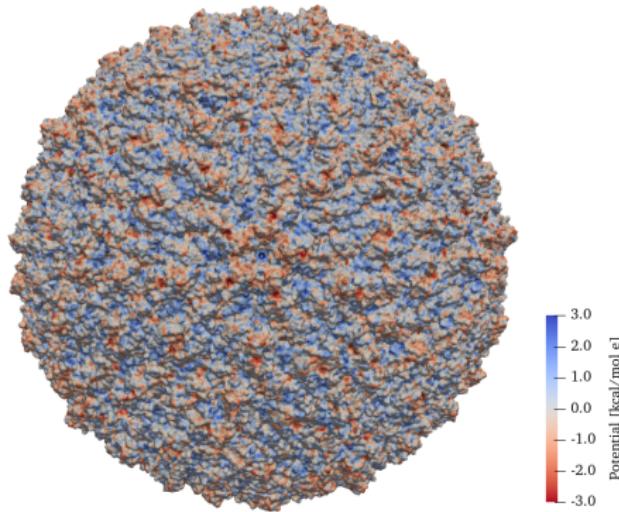
$$\epsilon_1 \frac{\partial \phi_1}{\partial \mathbf{n}} = \epsilon_2 \frac{\partial \phi_2}{\partial \mathbf{n}}$$

Let  $\phi_1 = \phi_{\text{reac}} + \phi_{\text{coul}}$  ( $\phi_{\text{coul}}$  is potential generated from the point charges only). Then want to compute the solvation energy.



$$\Delta G_{\text{solv}}^{\text{polar}} = \frac{1}{2} \sum_{k=1}^{N_q} q_k \phi_{\text{reac}}(\mathbf{r}_k)$$

# Results for Zika Virus



10m surface elements. 40 Core Compute node. Total time: 139.5 minutes, GMRES time: 80 minutes, 18 GMRES iterations, Max RAM use: 43GB,  $\Delta G_{solv} = -116254.9 \text{ kcal/mol}$ . Diagonal preconditioner through inverse of mass matrix with mass lumping.

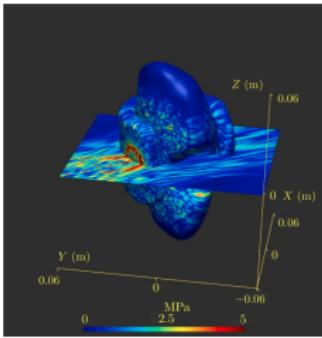
Joint work with T. Wang, C. Cooper, L. Barba

# High-Intensity Focused Ultrasound Treatment Planning



Single-trace transmission formulation

$$\left( A_{k_0}^j + \widetilde{A}_{k_j}^j \right) \gamma^{j,+} p_{tot} + \\ \sum_{i \neq j} A_{k_0}^{ji} \gamma^{i,+} p_{tot} = \gamma^{j,+} p_{inc}$$



Precondition with

$$C_j := \begin{bmatrix} \widetilde{DtN}^{j,+} \\ \widetilde{NtD}^{j,+} \end{bmatrix}$$

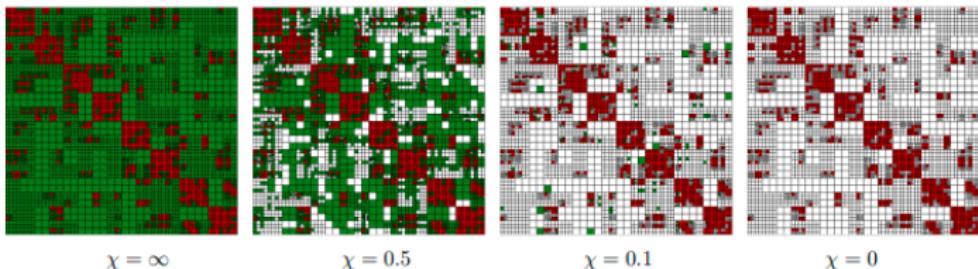
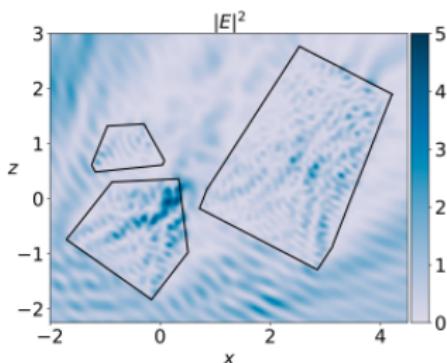
$$\widetilde{DtN} = ik \sqrt{1 + \frac{\Delta_\Gamma}{k_\epsilon^2}}$$

- Standard single-trace formulation ill-conditioned
- Not suitable for large iterative solvers

# Electromagnetic scattering through ice crystals

UCL

- Right: Electromagnetic scattering through ice crystals
- Bottom: Preconditioning by removing far-field contributions



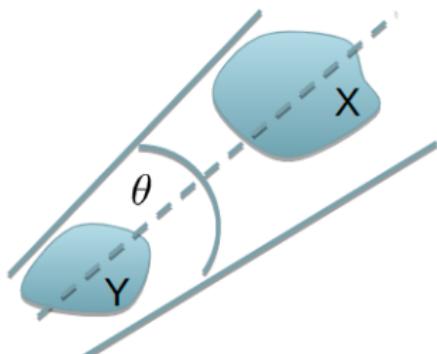
# Why is high-frequency complicated?

Fast boundary element methods rely on far-field compression.

Let  $x \in X$  (targets), and  $y \in Y$  (sources) For  $g(x, y) = \frac{e^{ik|x-y|}}{4\pi|x-y|}$  have

$$\|g(x, y) - \sum_{j=1}^{t_\epsilon} g_j(x)h_j(y)\| < \epsilon, \quad t_\epsilon \sim k^2$$

Number of terms in expansion grows quadratically with  $k$



- Require  $\theta \sim k^{-1}$  to guarantee compressibility
- Fast algorithms based on this compression proposed by Michielssen, Ying, Engquist, Bebendorf, Boerm, ...

# Designing scalable BEM for next gen HPC



Goal: Create a fast solver framework for fast parallel forward evaluation and approximate inversion of integral operators  
(<https://github.com/rusty-fast-solvers/roadmap>)

All component libraries developed in Rust with Rayon (multithreading) and MPI (across nodes) for parallelization.

Development Status:

- **rusty-green-kernel** AVX accelerated direct evaluation of Laplace, Helmholtz Green's functions. **Usable**.
- **rusty-tree** Serial and parallel Octree implementations. Serial tree implemented. Parallel tree early stages. **Partially implemented**.
- **rusty-compression** Fast randomized compression routines for approximate low-rank matrices. **Mostly finished**.
- **rusty-translation** Implementation of M2M, M2L, L2L, P2M, L2P operators based on numerical field compressions (KIFMM, etc.)  
**In planning**
- **rusty-fmm** Serial and parallel FMM loops **In planning**
- **rusty-inverse** Evaluation of approximate inverses **Not yet started**

Efficiently evaluate sums of the form

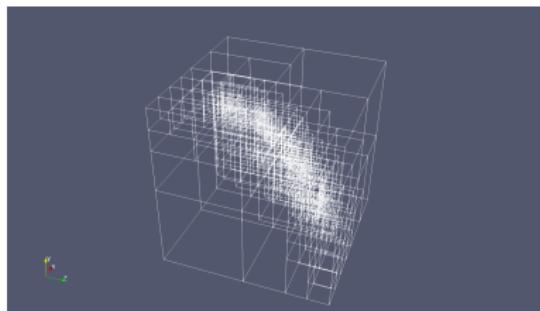
$$f(x_i) = \sum_j c_j g(x_i, y_j), \quad g(x, y) = \frac{e^{ik|x-y|}}{4\pi|x-y|}, k \in \mathbb{C}$$

[rusty-green-kernel](#) offers

- SIMD accelerated evaluation in single precision and double precision
- Optional multithreaded evaluation
- Needed for
  - Fast evaluation of nearfield interactions
  - Generation of translation operators

- Require adaptive hierarchical domain partitioning for organising nearfield and farfield data.
- Adaptive load-balanced setup of Octrees on distributed platforms (Sundar/Sampar/Biros 2008)
- Have developed working MPI parallelized library

- Initial experimental runs with **1 billion** points on ARCHER2
- Right: Small octree for quarter sphere



Joint work with Sri Kailasa, publication in preparation

# Field translation operators



Let  $y_0 \in Y$  be in the set of sources,  $x_0 \in X$  in the set of targets.  $B_{y_0}$  Octree box around source.  $B_{x_0}$  octree box around target.

Field in the exterior of  $B_{y_0}$

$$f(x) \approx \sum_{j=1}^N c_j^{(s)} g(x, y_j), \quad x \in \mathbb{R}^3 \setminus \overline{B_{y_0}}$$

Field in the interior of  $B_{x_0}$

$$f(x) \approx \sum_{j=1}^N c_j^{(t)} g(x, y_j), \quad x \in B_{x_0}$$

Field translation operator  $A$

$$c^{(t)} = A c^{(s)}$$

- Analytical multipole expansions, etc. (Greengard, Rokhlin, ...)
- Algebraic techniques
  - Chebychev series representations: Black-Box FMM (Darve, ...)
  - Fundamental solution approximations: KIFMM (Ying, ...)

**rusty-compression:** Low-rank approximate translation operators

$$A \approx UV^T, \quad U, V^H \in \mathbb{C}^{N \times k}$$

**rusty-field:** Evaluate various types of translation operators

- Large-scale acoustic/electromagnetic integral equation solvers significant undertaking
- Good results with Bempp for single node problems, but natural limits for larger applications
- We are building a new scalable and highly parallel code base from scratch that addresses the needs of very large scattering problems
- Further problems: FEM/BEM Coupling, Domain decomposition methods, High-order surface representations,...