Electrostatic simulations with Bempp and Exafmm - A black-box coupling approach

Timo Betcke
t.betcke@ucl.ac.uk
University College London

Joint with:
Tingyu Wang, Christopher Cooper, and Lorena Barba

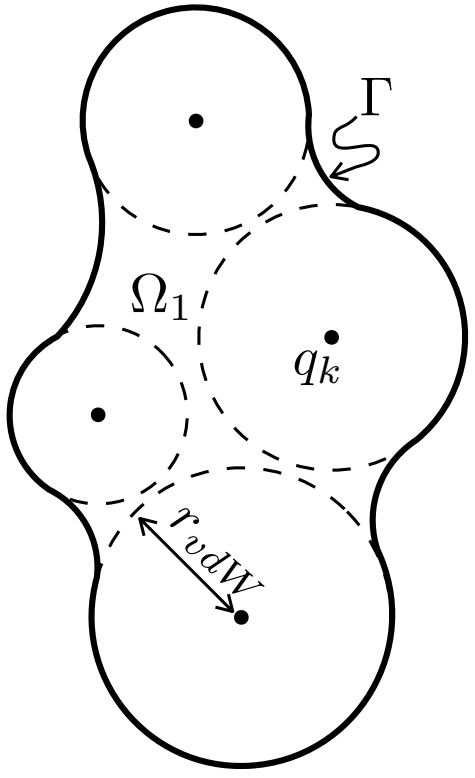
Excalibur-SLE: Exascale HPC for System Level Engineering
EPSRC Grant: EP/V001531/1

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 Γ:

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

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

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

Boundary Integral Formulation

Exterior field formulation[1]

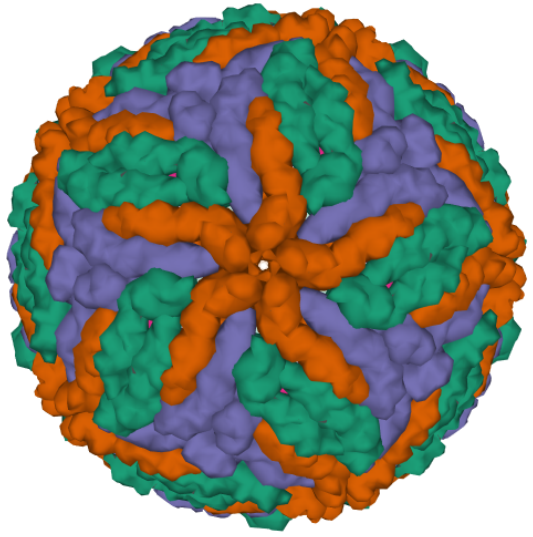
$$\frac{\phi_{2,\Gamma}}{2}\left(\frac{\epsilon_1}{\epsilon_2}+1\right)-\left(K_Y^\Gamma-\frac{\epsilon_1}{\epsilon_2}K_L^\Gamma\right)(\phi_{2,\Gamma})$$
$$+\left(V_Y^\Gamma-V_L^\Gamma\right)\left(\frac{\partial}{\partial\mathbf{n}}\phi_{2,\Gamma}\right)=\sum_{k=0}^{N_q}\frac{q_k}{4\pi\epsilon_2|\mathbf{r}_\Gamma-\mathbf{r}_k|}$$
$$-\frac{\epsilon_1}{\epsilon_2}\left(W_Y^\Gamma-W_L^\Gamma\right)(\phi_{2,\Gamma})+\frac{1}{2}\frac{\phi_{2,\Gamma}}{\partial\mathbf{n}}\left(1+\frac{\epsilon_1}{\epsilon_2}\right)$$
$$+\left(\frac{\epsilon_1}{\epsilon_2}K_Y'^\Gamma-K_L'^\Gamma\right)\left(\frac{\partial}{\partial\mathbf{n}}\phi_{2,\Gamma}\right)=\sum_{k=0}^{N_q}\frac{\partial}{\partial\mathbf{n_r}}\left(\frac{q_k}{4\pi\epsilon_2|\mathbf{r}_\Gamma-\mathbf{r}_k|}\right)$$

Also possible to derive interior field formulation (due to Juffur) or simple direct coupling from first line of Calderòn projector. Exterior formulation most favourable in terms of conditioning.

[1] Lu et. al., Proc. Natl. Acad. Sci. USA 103 (51) (2006) 19314-19319

Zika virus

- Around 1.6m atoms
- Generated mesh has around 10m surface elements



Use Galerkin BEM code Bempp-cl accelerated with fast particle interactions through Exafmm-t

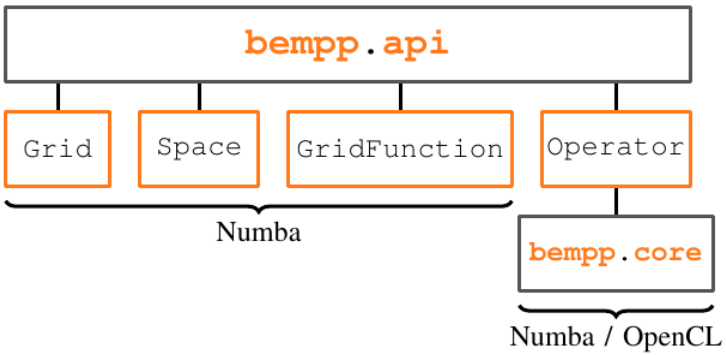
Outline of rest of talk:

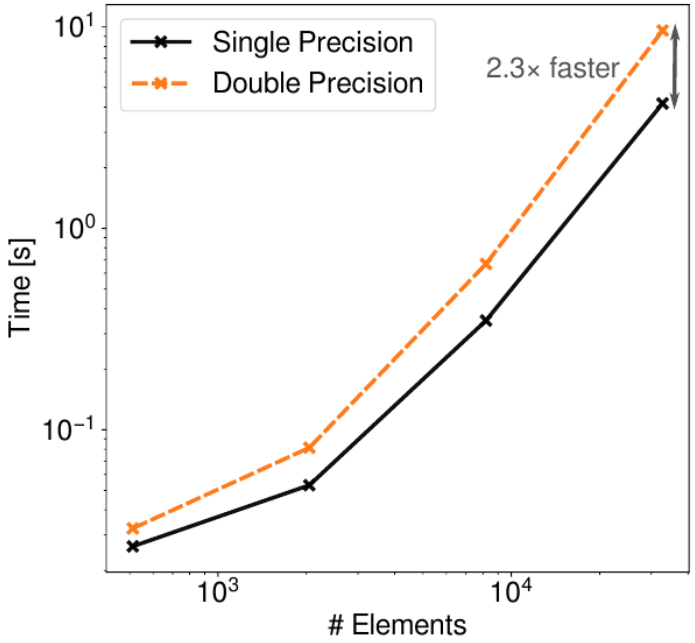
- A short introduction to Bempp-cl and Exafmm-t
- Black-box coupling strategies for Galerkin BEM
- Practical issues and ongoing work

Bempp-cl

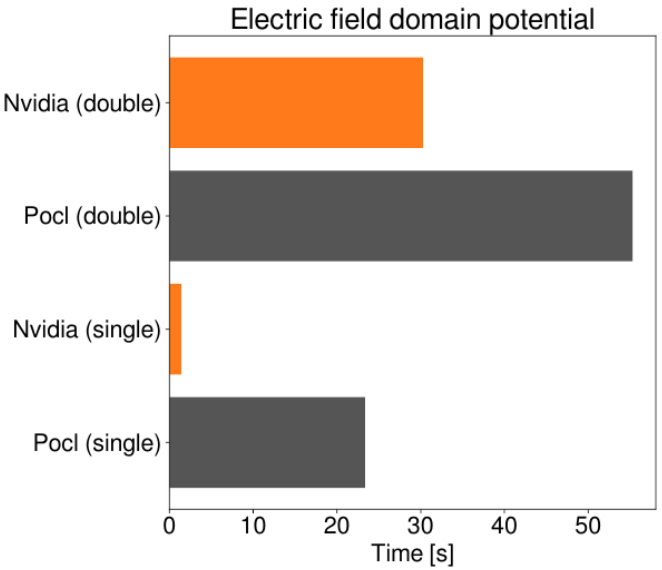
- Successor of the Bempp boundary element library.
- Fully developed using Python with fast OpenCL kernels for computational routines.
- Galerkin discretisation of operators for Laplace, Helmholtz, and Maxwell problems.
- Compute kernels can be run on CPU or offloaded to GPU.
- Core routines optimised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

Characteristics of Bempp-cl



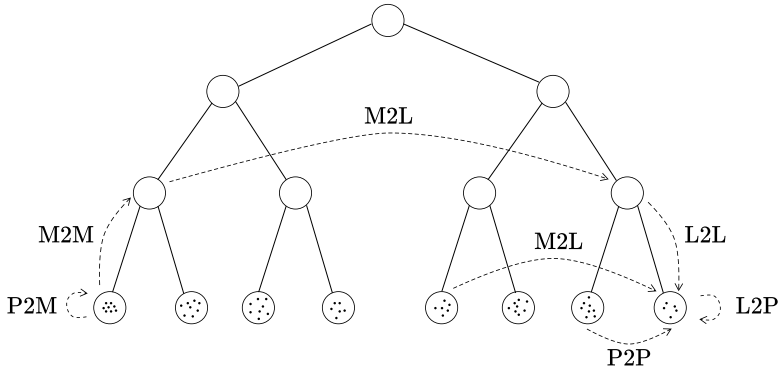


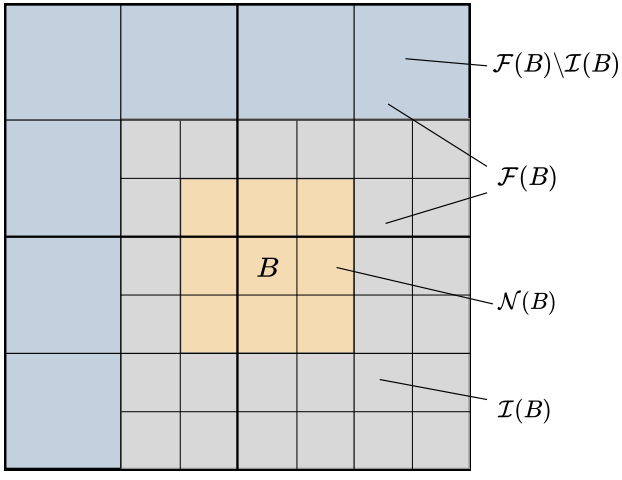
- Top: Structure of Bempp-cl
- Top-Right: AVX acceleration in Bempp-cl
- Bottom-Right: Offloading of a domain potential operator

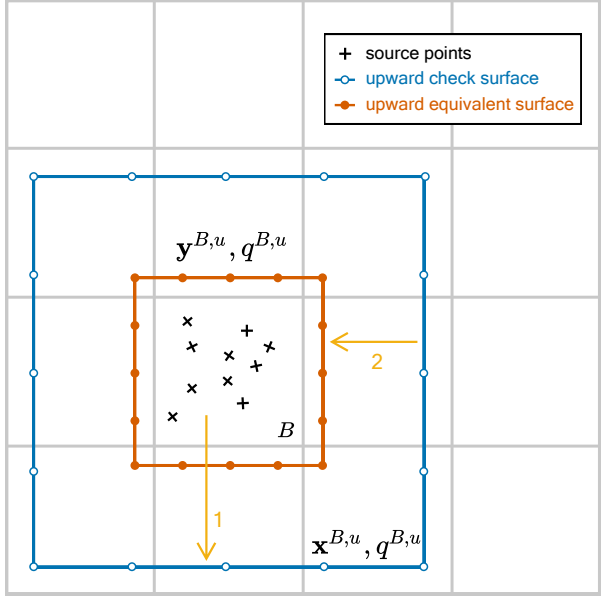


Exafmm-t: High-Performance KIFMM

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







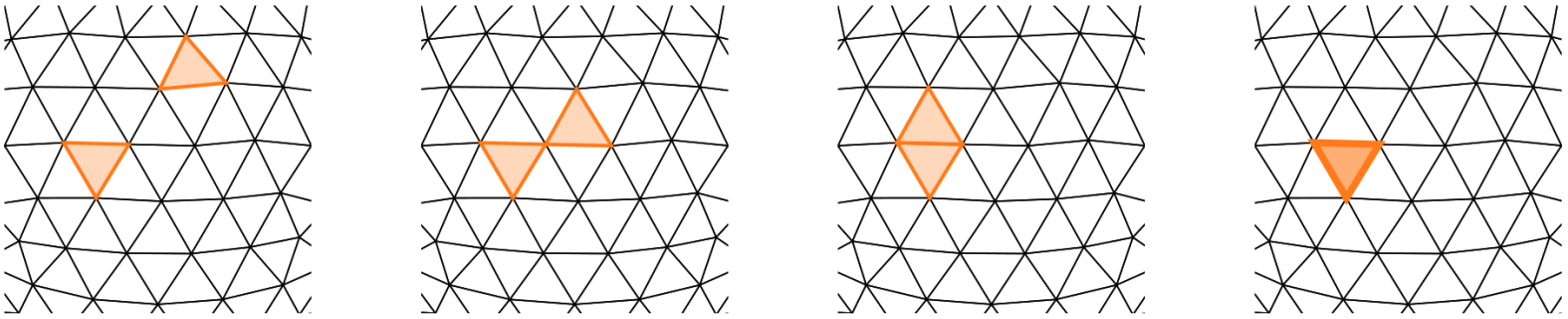
Coupling Bempp-cl and Exafmm-t

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.

The correction matrix



Problem: Any FMM code also evaluates the near-field. Near-field can contain adjacent triangles (singular quadrature rules), and non-adjacent triangles (standard triangle Gauss rule).

The sources and target points in the FMM are arising from the non-singular quadrature rule.

Solution: After FMM evaluation subtract out the regular quadrature rule contribution from adjacent triangles and add in the correct singular quadrature rule contribution.

Number of FMM passes per Operator

Operator	V	K	K'	W_Y, W_L
FMM Passes	1	3	1	6, 3

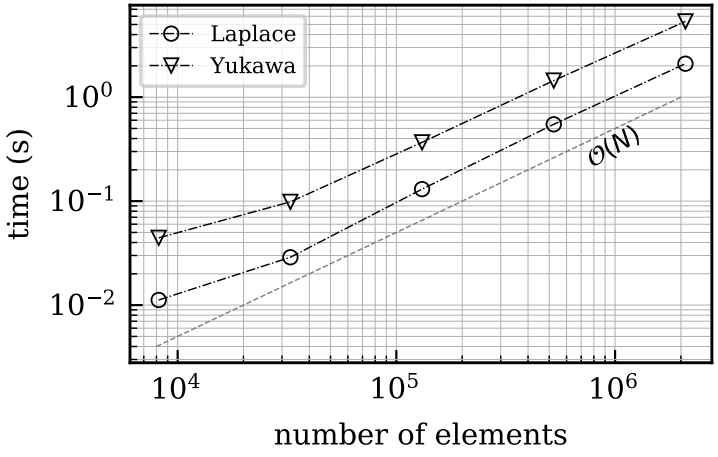
Example: Double-Layer Potential Operator

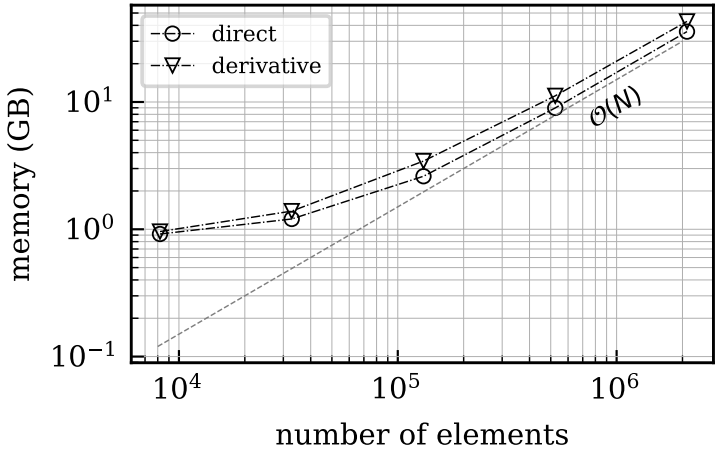
$$[K\phi](\mathbf{x}) = \int_\Gamma \partial_{\mathbf{n}(\mathbf{y})} g(\mathbf{x},\mathbf{y})\phi(\mathbf{y}) ds(\mathbf{y})$$
$$= -\sum_{j=1}^{3} [\nabla_{\mathbf{x}} g(\mathbf{x},\mathbf{y})]_j \, \mathbf{n}_j(\mathbf{y})\phi(\mathbf{y}) ds(\mathbf{y})$$

Require three FMM passes for the three different densities $\tilde{\phi}_j(\mathbf{y}) := \mathbf{n}_j(\mathbf{y})\phi(\mathbf{y})$, $j = 1, \ldots, 3$.

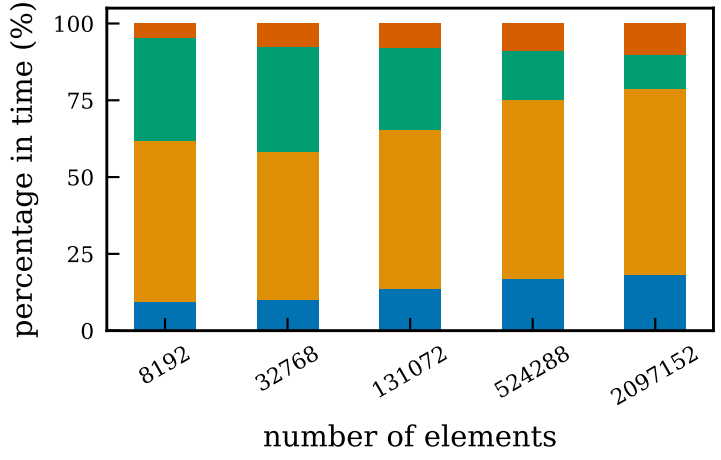
The transmission operator for the exterior Poisson-Boltzmann formulation requires 19 FMM passes.

Poisson-Boltzmann for a sphere

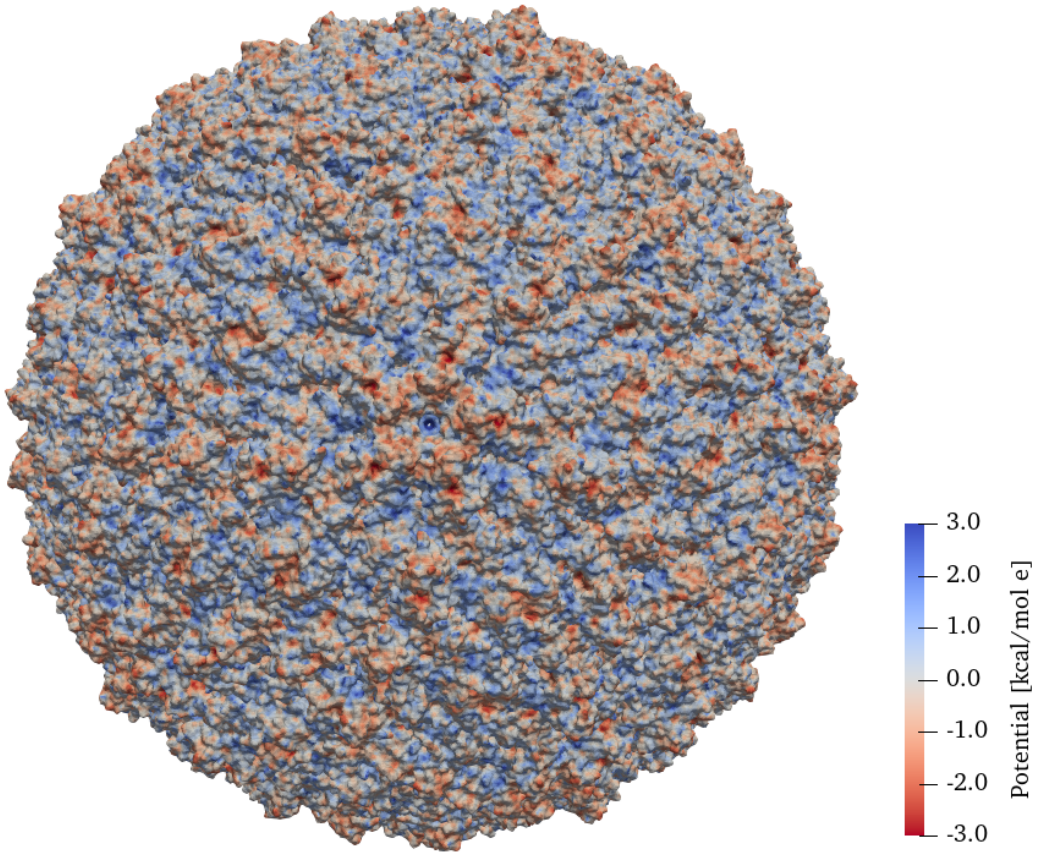




- FMM Expansion Order 5
- 6 regular quadrature points per element
- **Right** Blue: Laplace, Yellow: Yukawa, Green: Singular Correction, Red: Other



Results for Zika Virus



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

The next step: Scalable fast solver framework

Goal: Create a fast solver framework for fast parallel forward evaluation and approximate inversion of integral operators.

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. Usuable.
- 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

Summary

- Generic, practically usuable and efficient black-box coupling of Galerkin BEM and Exafmm.
- All computational results shown today steered through interactive Jupyter notebooks.
- Python glue language between the codes. High productivity through Jupyter interface.
- For links to paper, reproducible data and codes see `https://github.com/barbagroup/bempp_exafmm_paper`

Work supported through Exalibur-SLE: Exascale HPC for System Level Engineering `https://excalibur-sle.github.io/`