

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

Timo Betcke

[t.betcke@ucl.ac.uk](mailto: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

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

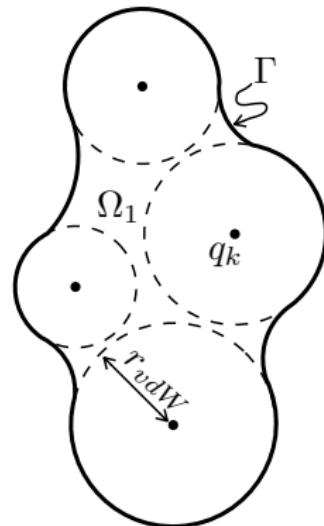
$$(\Delta - \kappa^2)\phi_2 = 0 \text{ in } \Omega_2$$

Interface conditions on  $\Gamma$ :

$$\phi_1 = \phi_2,$$

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

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.



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

Exterior field formulation<sup>1</sup>

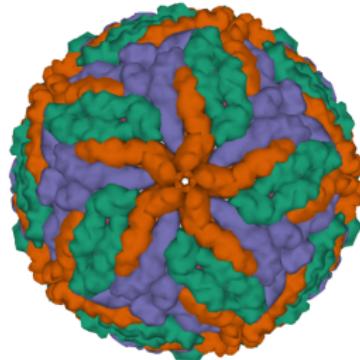
$$\begin{aligned}
 & \frac{\phi_{2,\Gamma}}{2} \left( \frac{\epsilon_1}{\epsilon_2} + 1 \right) - \left( K_Y^\Gamma - \frac{\epsilon_1}{\epsilon_2} K_L^\Gamma \right) (\phi_{2,\Gamma}) \\
 & + (V_Y^\Gamma - V_L^\Gamma) \left( \frac{\partial}{\partial \mathbf{n}} \phi_{2,\Gamma} \right) = \sum_{k=0}^{N_q}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

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.

---

<sup>1</sup>Lu et. al., Proc. Natl. Acad. Sci. USA 103 (51) (2006) 19314-19319

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



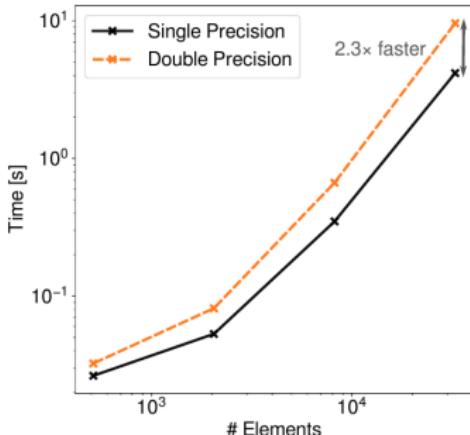
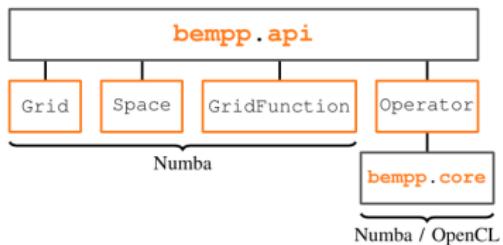
Use Galerkin BEM code Bempp-cl accelerated with fast particle interactions through Exafmm

Outline of rest of talk:

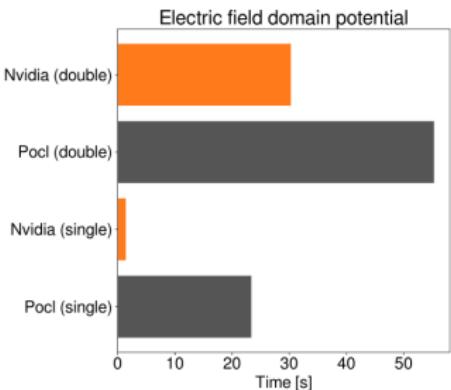
- A short introduction to Bempp-cl and Exafmm
- Black-box coupling strategies for Galerkin BEM
- Practical issues and ongoing work

- 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



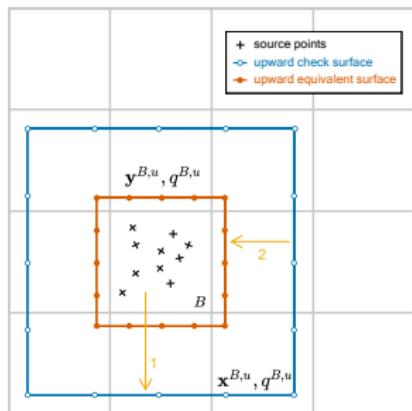
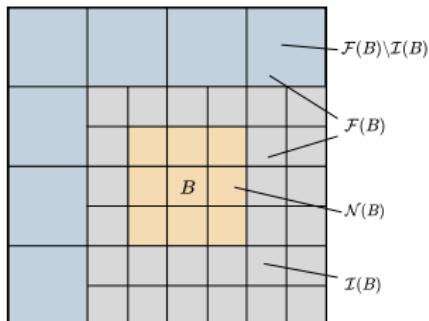
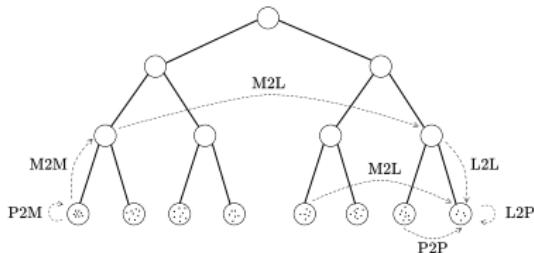
- Top: Structure of Bempp-cl
- Top-Right: AVX acceleration in Bempp-cl
- Bottom-Right: Offloading of a domain potential operator



# Exafmm-t: High-Performance KIFMM



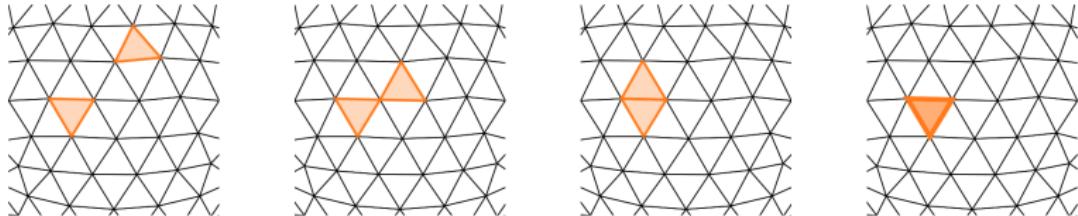
- Kernel-Independent FMM library.
- Written in C++, Provides Python Interface.
- Highly efficient at higher accuracies.



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.



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

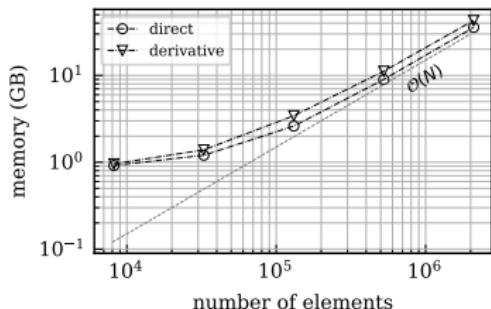
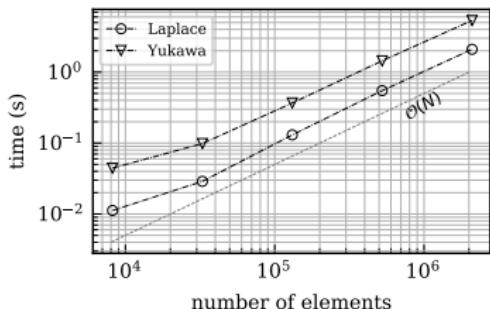
Example: Double-Layer Potential Operator

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

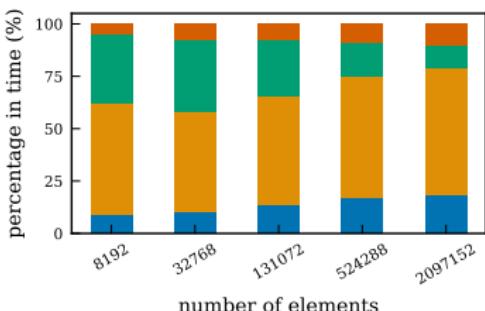
Require three FMM passes for the three different densities  
 $\tilde{\phi}_j(\mathbf{y}) := \mathbf{n}_j(\mathbf{y}) \phi(\mathbf{y}), j = 1, \dots, 3.$

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

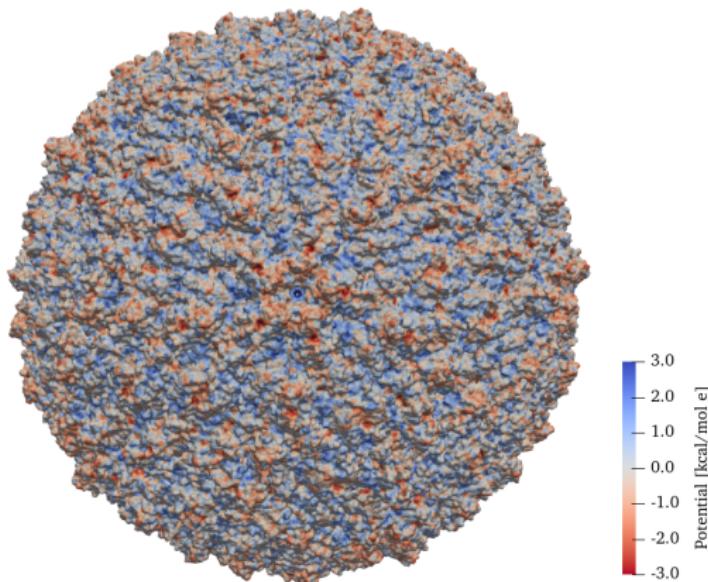
# Poisson-Boltzmann for a sphere



- 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,  
 $\Delta G_{solv} = -116254.9 \text{ kcal/mol}$ . Diagonal preconditioner through inverse of mass matrix with mass lumping.

## PyExafmm - KIFMM in Python

Developed by [Srinath Kailasa](#) at UCL

- Provide a hackable and performant Adaptive FMM in Python.
- Test bed for further developments.
- Far-field computation accelerated through Numba.
- Randomized compression of M2L operators.

<https://github.com/exafmm/pyexafmm>

# A Rust based scalable fast solver framework



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**

- Generic, practically usable 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](https://github.com/barbagroup/bempp_exafmm_paper)

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

- T. Wang, C. Cooper, T. Betcke, L. Barba, *High-productivity, high-performance workflow for virus-scale electrostatic simulations with Bempp-Exafmm*, arXiv preprint arXiv:2103.01048 (2021),  
[https://github.com/barbagroup/bempp\\_exafmm\\_paper](https://github.com/barbagroup/bempp_exafmm_paper)
- T. Betcke, M. W. Scroggs, *Bempp-cl: A fast Python based just-in-time compiling boundary element library*, Journal of Open Source Software, 6(59), 2879 (2021), <https://doi.org/10.21105/joss.02879>
- T. Betcke, M. W. Scroggs, *Designing a high-performance boundary element library with OpenCL and Numba*, Computing in Science & Engineering, to appear (2021)  
<https://doi.org/10.1109/MCSE.2021.3085420>
- T. Wang, R. Yokota, L. Barba, *ExaFMM: a high-performance fast multipole method library with C++ and Python interfaces*. Journal of Open Source Software, 6(61), 3145 (2021),  
<https://doi.org/10.21105/joss.03145>