Protein pK_a calculations using a fast direct boundary element solver

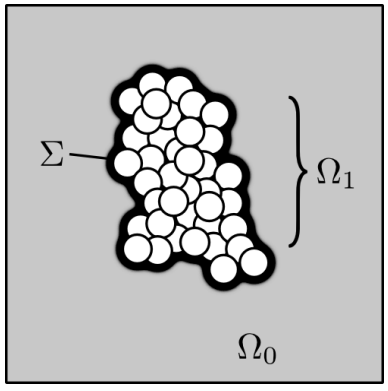
Kenneth L. Ho and Leslie Greengard

Courant Institute, New York University

SIAM LS 2012

Outline

Macromolecular electrostatics



Molecule: discrete collection of charged atoms

Ω_0: solvent

Ω_1: (solvent-excluded) molecular volume

Σ: molecular surface

PDE for the electrostatic potential:

$$-\left(\Delta - \kappa^2\right)\varphi = 0 \qquad \text{in } \Omega_0 \quad \text{(linearized Poisson-Boltzmann)}$$

$$-\Delta\varphi = \frac{1}{\varepsilon_1}\sum_i q_i \delta\left(\mathbf{r} - \mathbf{r}_i\right) \qquad \text{in } \Omega_1 \quad \text{(Poisson)}$$

$$[\varphi] = \left[\varepsilon \frac{\partial \varphi}{\partial \nu}\right] = 0 \qquad \text{on } \Sigma \quad \text{(continuity)}$$

- Continuum solvent, atomic detail (singular sources)
- Linear, second-order, elliptic

Boundary integral formulation

Green's function: $$G_k(\mathbf{r},\mathbf{s}) = \frac{e^{-k|\mathbf{r}-\mathbf{s}|}}{4\pi|\mathbf{r}-\mathbf{s}|}$$

Single-layer potential: $$S_k[\sigma](\mathbf{r}) = \int_\Sigma G_k(\mathbf{r},\mathbf{s})\,\sigma(\mathbf{s})\,dA_\mathbf{s} \qquad \text{in } \Omega_{0,1}$$

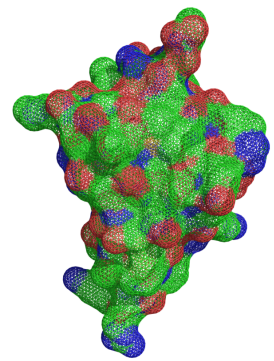
Double-layer potential: $$D_k[\mu](\mathbf{r}) = \int_\Sigma \frac{\partial G_k}{\partial \nu_\mathbf{s}}(\mathbf{r},\mathbf{s})\,\mu(\mathbf{s})\,dA_\mathbf{s} \qquad \text{in } \Omega_{0,1}$$

Solution representation:

$$\varphi \equiv \begin{cases} S_\kappa\sigma + D_\kappa\mu & \text{in } \Omega_0, \\ S_0\sigma + \alpha D_0\mu + \varphi_s & \text{in } \Omega_1, \end{cases} \qquad \alpha \equiv \frac{\varepsilon_0}{\varepsilon_1}, \quad \varphi_s(\mathbf{r}) \equiv \frac{1}{\varepsilon_1}\sum_i q_i G_0(\mathbf{r},\mathbf{r}_i)$$

Boundary integral equation on Σ:

$$\frac{1}{2}(1+\alpha)\mu + (S_\kappa - S_0)\sigma + (D_\kappa - \alpha D_0)\mu = \varphi_s,$$
$$-\frac{1}{2}(1+\alpha)\sigma + (\alpha S'_\kappa - S'_0)\sigma + \alpha(D'_\kappa - D'_0)\mu = \frac{\partial\varphi_s}{\partial\nu}$$



Rewrite in block form: $$(I+\lambda K)\begin{bmatrix}\mu\\ \sigma\end{bmatrix} = \lambda\begin{bmatrix}\varphi_s\\ -\varphi'_s\end{bmatrix} \xrightarrow{\text{discretize}} A(\Sigma)\,x = b(q)$$

Numerical considerations

Why integral equations?

- Pros: high accuracy, handles singular functions, dimensional reduction
- Cons: dense matrices, **computational cost**

(Compare with finite differences or finite elements.)

Numerical considerations

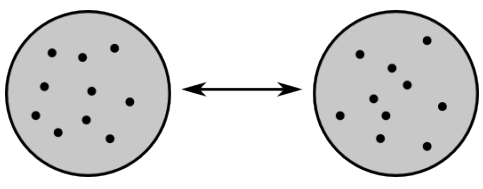
Why integral equations?

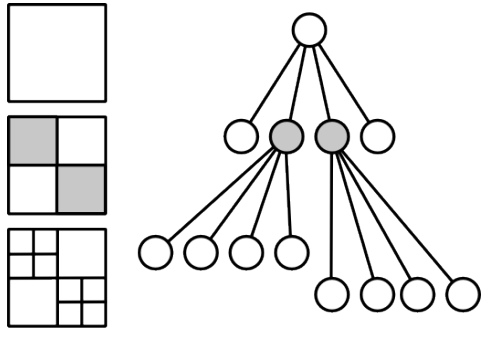
- Pros: high accuracy, handles singular functions, dimensional reduction
- Cons: dense matrices, **computational cost**

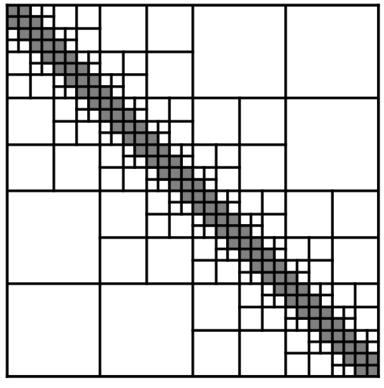
(Compare with finite differences or finite elements.)

But integral equation matrices are often structured.

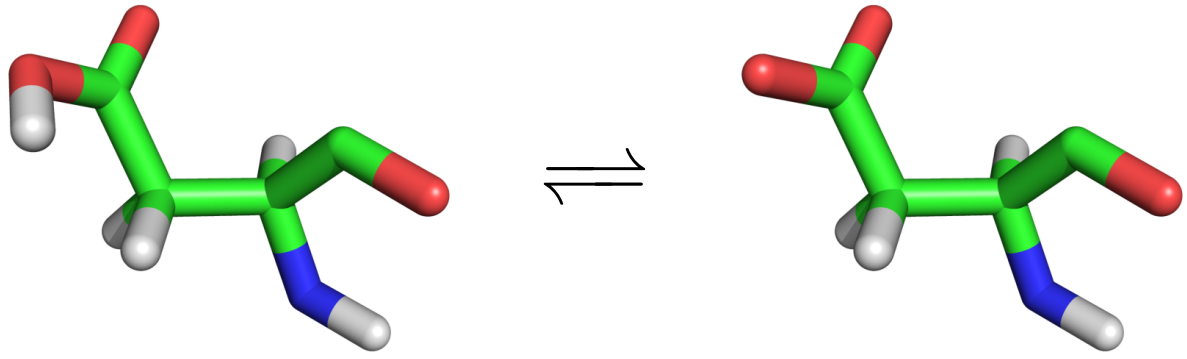
- Hierarchical low-rank approximation of far-field interactions
- Matrix-vector multiplication in $\mathcal{O}(N \log N)$ operations
  - Treecode, FMM, panel clustering, pFFT, FFTSVD
- Fast iterative solvers when combined with GMRES, BiCG, CGR, etc.







Protein pK_a calculations



$$\mathrm{p}K_\mathrm{a} \equiv -\log_{10}\frac{[\mathrm{A}]\,[\mathrm{H}]}{[\mathrm{AH}]} = \log_{10}\frac{[\mathrm{AH}]}{[\mathrm{A}]} + \mathrm{pH}$$

Ionization behavior is important for many biomolecular phenomena

- Binding affinities
- Enzymatic activities
- Structural properties

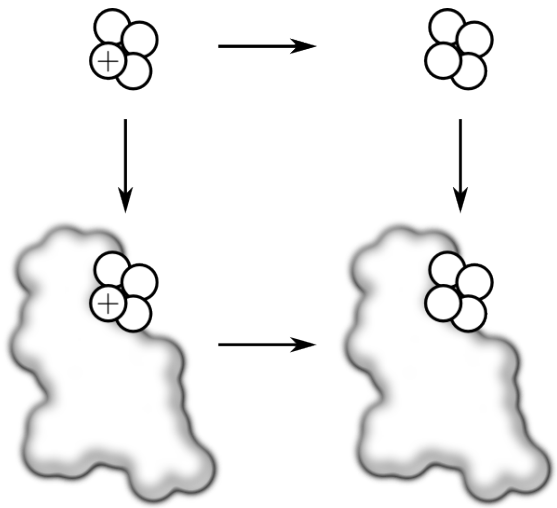
Theoretical interest: Bashford and Karplus, Juffer et al., Alexov et al.

A single titrating site

$$\mathrm{p}K_\mathrm{a} = \frac{\beta}{\ln 10}\Delta G^\mathrm{p}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}}$$

$$\begin{aligned}\Delta G^\mathrm{p}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}} &= \Delta G^\mathrm{s}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}} + \Delta G^{\mathrm{s}\to\mathrm{p}}_\mathrm{A} - \Delta G^{\mathrm{s}\to\mathrm{p}}_\mathrm{AH} \\ &= \underbrace{\Delta G^\mathrm{s}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}}}_{\text{experiment}} + \underbrace{\Delta G^\mathrm{s}_{\mathrm{A}\to\mathrm{AH}} - \Delta G^\mathrm{p}_{\mathrm{A}\to\mathrm{AH}}}_{\text{electrostatic only}}\end{aligned}$$

$$\begin{array}{ccc} \mathrm{A_sH} & \xrightarrow{\Delta G^\mathrm{s}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}}} & \mathrm{A_s} + \mathrm{H} \\ \Big\downarrow{\scriptstyle \Delta G^{\mathrm{s}\to\mathrm{p}}_\mathrm{AH}} & & \Big\downarrow{\scriptstyle \Delta G^{\mathrm{s}\to\mathrm{p}}_\mathrm{A}} \\ \mathrm{A_pH} & \xrightarrow{\Delta G^\mathrm{p}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}}} & \mathrm{A_p} + \mathrm{H} \end{array}$$



$$\mathrm{p}K_\mathrm{a} = \underbrace{\mathrm{p}K^\mathrm{model}_\mathrm{a}}_{\text{experiment}} - \frac{\beta}{\ln 10}\underbrace{\Delta\Delta G^{\mathrm{s}\to\mathrm{p}}_{\mathrm{A}\to\mathrm{AH}}}_{\text{electrostatic}}$$

Multiple titrating sites

Let $\theta_i \in \{0, 1\}$ denote the protonation state of each site $i = 1, \ldots, M$.

$$\mathrm{p}K_i^{\mathrm{intr}} \equiv \mathrm{p}K_i^{\mathrm{model}} - \frac{\beta}{\ln 10} \Delta\Delta G^{\mathrm{s}\to\mathrm{p}}_{\mathrm{A}\to\mathrm{A}(e_i)}$$

$$\Delta G_{\mathrm{A}\to\mathrm{A}(e_i)}\,(\mathrm{pH}) = -RT \ln 10 \left(\mathrm{p}K_i^{\mathrm{intr}} - \mathrm{pH}\right)$$

$$\Delta G_{\mathrm{A}\to\mathrm{A}(\theta)}\,(\mathrm{pH}) = -RT \ln 10 \sum_i \theta_i \left(\mathrm{p}K_i^{\mathrm{intr}} - \mathrm{pH}\right) + \frac{1}{2} \sum_i \theta_i \sum_{j\neq i} \theta_j \Delta G_{ij}$$

Sample mean site protonation using Markov chain Monte Carlo:

$$\langle \theta_i \rangle\,(\mathrm{pH}) = \frac{1}{Z} \sum_\theta \theta_i e^{-\beta \Delta G_{\mathrm{A}\to\mathrm{A}(\theta)}(\mathrm{pH})}, \quad \mathrm{p}K_i = \arg\underset{\mathrm{pH}}{} \langle \theta_i \rangle\,(\mathrm{pH}) = \frac{1}{2}$$

Bottleneck: interaction energies in protein

- Calculate φ_j for each j: solve $A(\Sigma)x = b(q_j)$
- Compute $\Delta G_{ij} = q_i^{\mathsf{T}} \varphi_j$ for each i
- Requires M solves with the same matrix

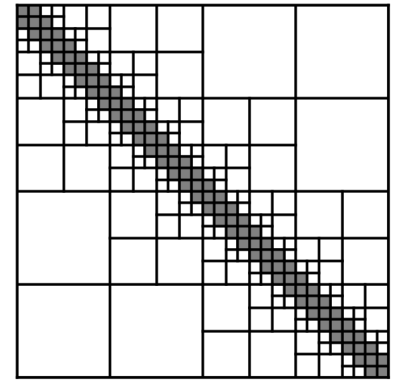
Solving systems with multiple right-hand sides

Standard iterative solvers for $Ax = b$:

- Sequence of operations depends on b
- Can be inefficient for multiple right-hand sides
- c.f. blocking, projection, deflation, subspace recycling

An alternative: direct solvers

- Compute A^{-1} (factor A)
- Reuse factors for each solve
- Robust, always works
- Accelerate using similar low-rank ideas



Various approaches in recent years:

- $\mathscr{H}$-matrices (Hackbusch, Börm, Grasedyck, Bebendorf et al.)
- HSS matrices (Chandrasekaran, Gu, Xia, Li et al.)
- Skeletonization (Martinsson, Rokhlin, Greengard, Gillman et al.)
  - BIEs in 2D
  - One-level BIEs in 3D

A fast direct solver for integral equations

Here, we present a multilevel skeletonization-based fast direct solver in general dimension. For BIEs:

	2D	3D
precomp	$\mathcal{O}(N)$	$\mathcal{O}(N^{3/2})$
solve	$\mathcal{O}(N)$	$\mathcal{O}(N \log N)$

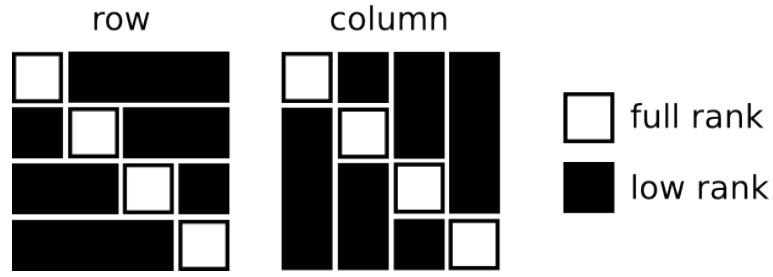
Main ideas/take-home messages :

- Kernel-independent: Laplace, Stokes, Yukawa, low-frequency Helmholtz, etc.
- Robust to geometry (e.g., boundary vs. volume, dimensionality)
- User-specified precision: trade accuracy for speed
- Naturally exposes the data-sparsity of integral equation matrices
- Very fast solve times, beating the FMM by factors of 100–1000
- Simple framework: easy to analyze, implement, and optimize
- Somewhat similar in flavor to nested dissection
- Can also apply to PDE formulations (Xia, Gillman et al.)

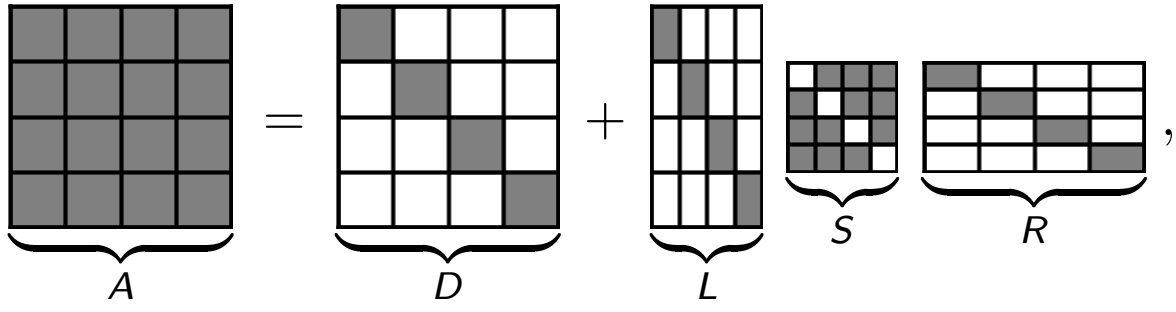
Block separable matrices

A block matrix A is block separable if

$$\underbrace{\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}}_{A_{ij}} = \underbrace{\begin{bmatrix} \times \\ \times \end{bmatrix}}_{L_i} \underbrace{\begin{bmatrix} \times \end{bmatrix}}_{S_{ij}} \underbrace{\begin{bmatrix} \times & \times \end{bmatrix}}_{R_j}, \quad i \neq j.$$



Then



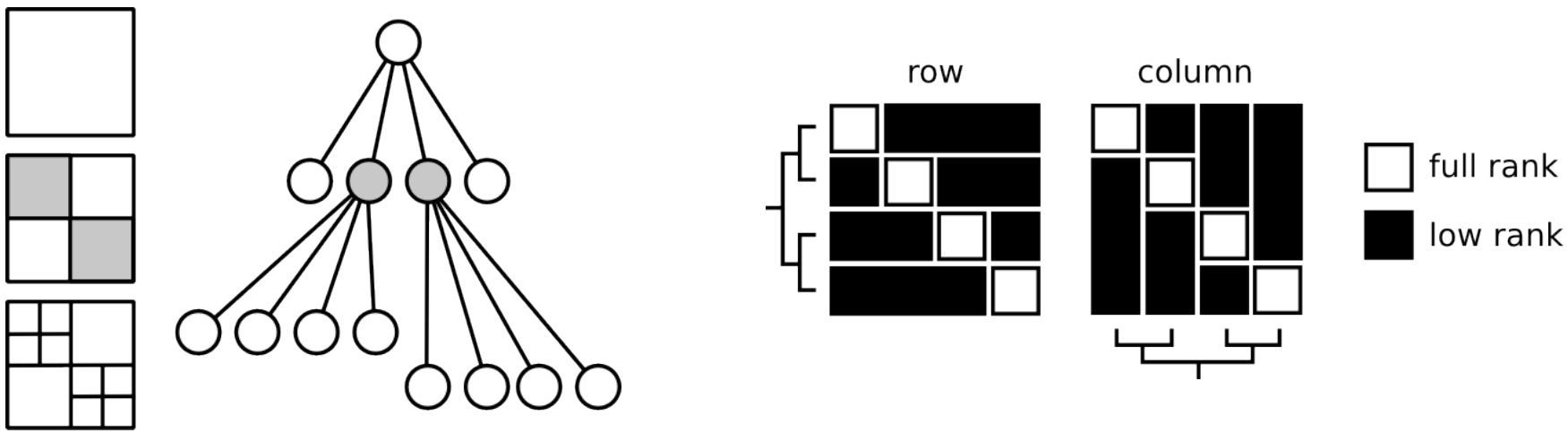
so $Ax = b$ is equivalent to the structured sparse system

$$\begin{bmatrix} D & L & \\ R & & -I \\ & -I & S \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}$$

with $z \equiv Rx$ and $y \equiv Sz$. Factor using UMFPACK, SuperLU, WSMP, etc.

Hierarchically block separable matrices

Integral equation matrices are, in fact, hierarchically block separable, i.e., they are block separable at every level of an octree-type ordering.



In this setting, much more powerful algorithms can be developed.

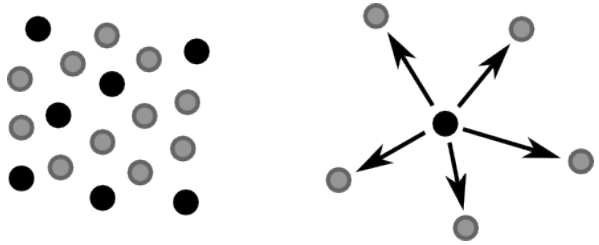
Interpolative decomposition

An interpolative decomposition of a rank-k matrix is a factorization

$$\underbrace{A}_{m\times n} = \underbrace{B}_{m\times k}\,\underbrace{P}_{k\times n},$$

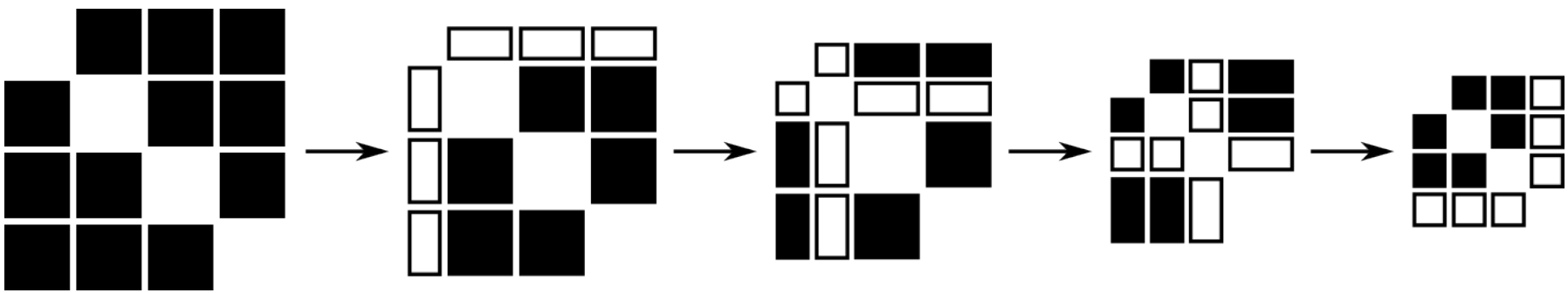
where B is a column-submatrix of A (with $\|P\|$ small).

- The ID compresses the column space; to compress the row space, apply the ID to A^{T}. We call the retained rows and columns skeletons.
- Adaptive algorithms can compute the ID to any specified precision $\epsilon > 0$.
- Related factorizations: SVD, RRQR, pseudoskeleton (CUR), ACA



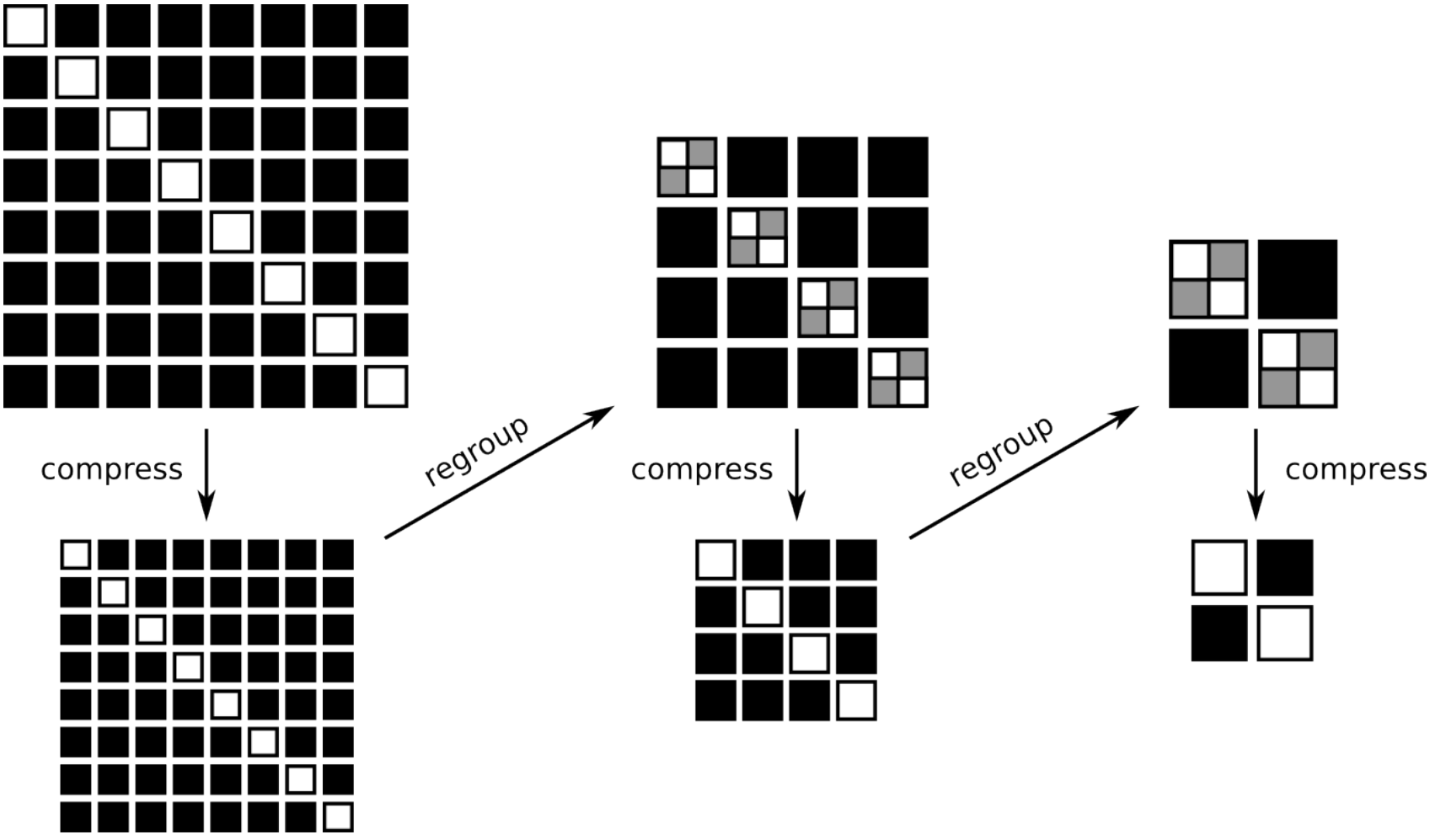
One-level matrix compression

- Compress the row space of each off-diagonal block row.
  Let the L_i be the corresponding row interpolation matrices.
- Compress the column space of each off-diagonal block column.
  Let the R_j be the corresponding column interpolation matrices.
- Approximate the off-diagonal blocks by $A_{ij} \approx L_i S_{ij} R_j$ for $i \neq j$.
- S is a skeleton submatrix of A



Skeletonization

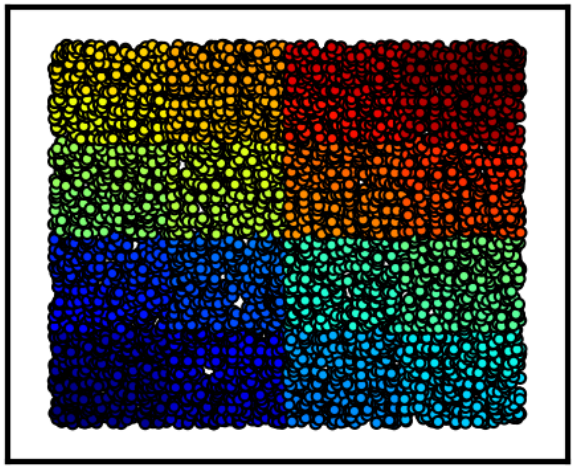
Multilevel matrix compression



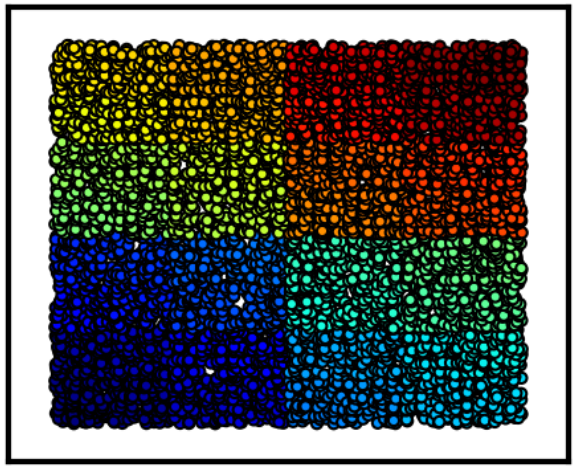
Recursive skeletonization

Data sparsification

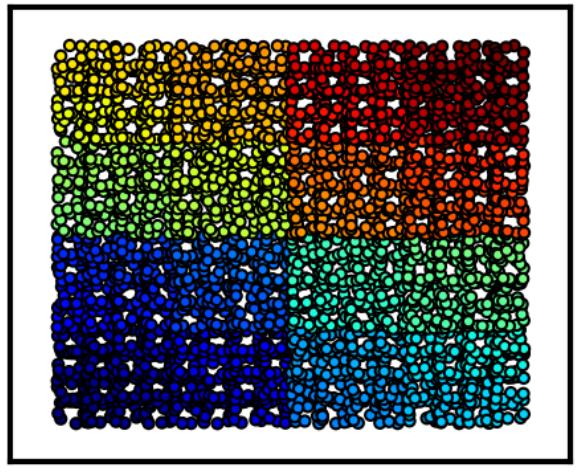
$N_0 = 8192$



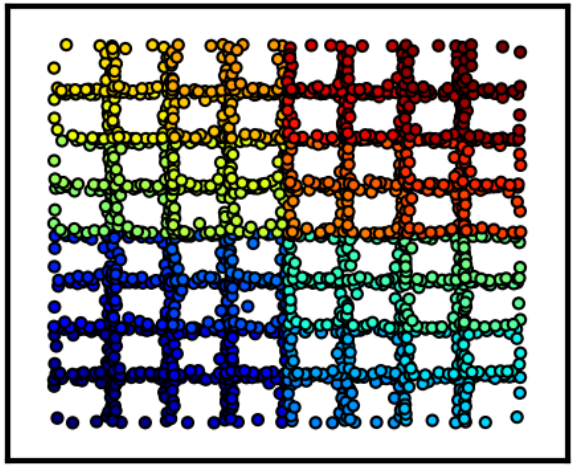
$N_1 = 7134$



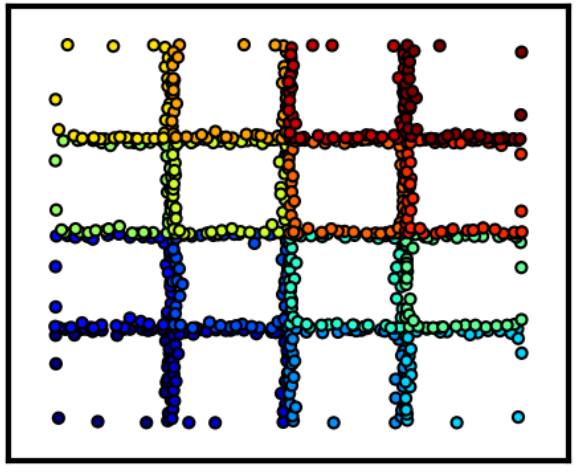
$N_2 = 4138$



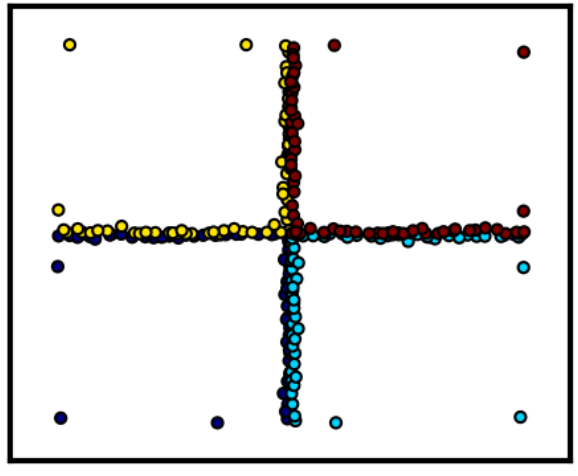
$N_3 = 1849$



$N_4 = 776$



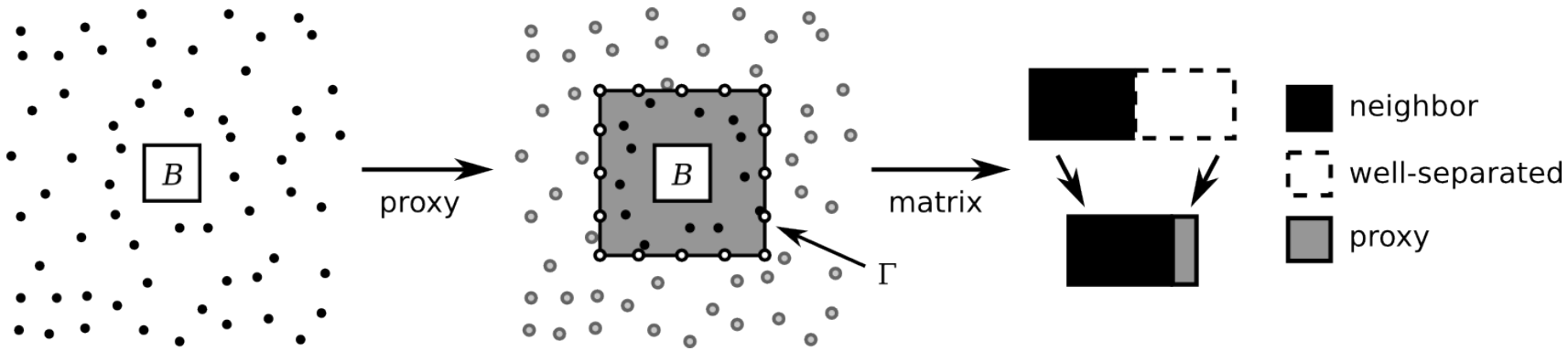
$N_5 = 265$



$$G(\mathbf{r},\mathbf{s}) = -\frac{1}{2\pi}\log|\mathbf{r}-\mathbf{s}|\ , \quad \epsilon = 10^{-3}$$

Accelerated compression for PDEs

- General compression algorithm is global and so at least $\mathcal{O}(N^2)$
- For potential fields, use Green's theorem to accelerate
- Represent well-separated interactions via a local proxy surface
- Can be generalized to non-PDE kernels using sparse grids



Compressed matrix representation

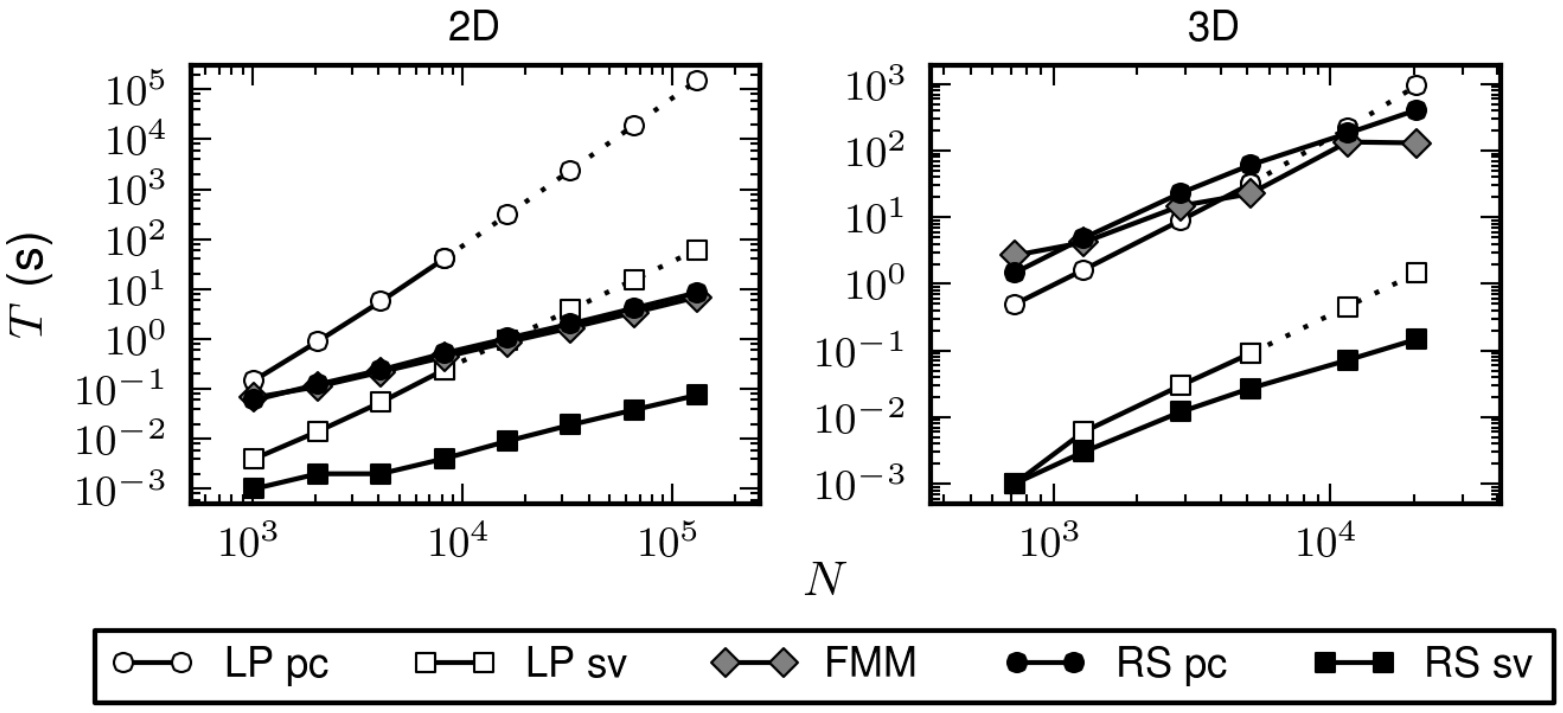
- Telescoping formula:

$$A \approx D^{(1)} + L^{(1)} \left[D^{(2)} + L^{(2)} \left(\cdots D^{(\lambda)} + L^{(\lambda)} S R^{(\lambda)} \cdots \right) R^{(2)} \right] R^{(1)}$$

- Efficient storage, fast matrix-vector multiplication (generalized FMM)
- Structured sparse inversion:

$$\begin{bmatrix}
D^{(1)} & L^{(1)} & & & & & \\
R^{(1)} & & -I & & & & \\
 & -I & D^{(2)} & L^{(2)} & & & \\
 & & R^{(2)} & \ddots & \ddots & & \\
 & & & \ddots & D^{(\lambda)} & L^{(\lambda)} & \\
 & & & & R^{(\lambda)} & & -I \\
 & & & & & -I & S
\end{bmatrix}
\begin{bmatrix} x \\ y^{(1)} \\ z^{(1)} \\ \vdots \\ \vdots \\ y^{(\lambda)} \\ z^{(\lambda)} \end{bmatrix}
=
\begin{bmatrix} b \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

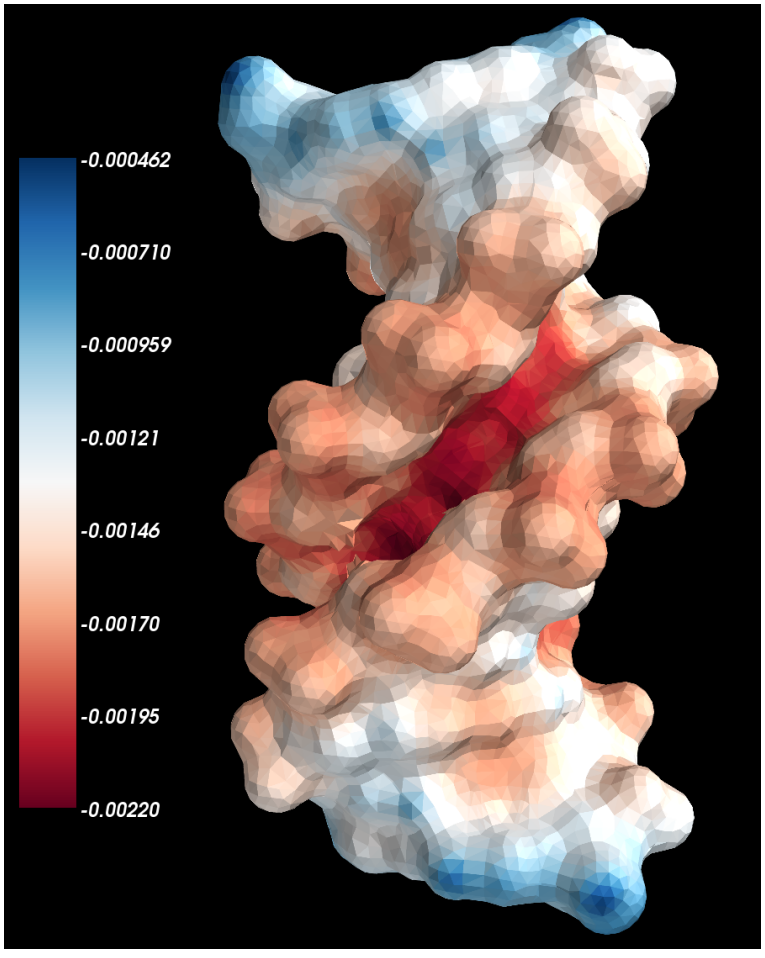
Laplace BIE solver



- Less memory-efficient than FMM/GMRES
- Each solve is extremely fast (in elements/sec)

ϵ	10^{-3}	10^{-6}	10^{-9}
2D	3.3×10^6	2.0×10^6	1.7×10^6
3D	6.0×10^5	1.4×10^5	6.2×10^4

Poisson electrostatics



$$-\Delta\varphi = 0 \qquad \text{in } \Omega_0$$

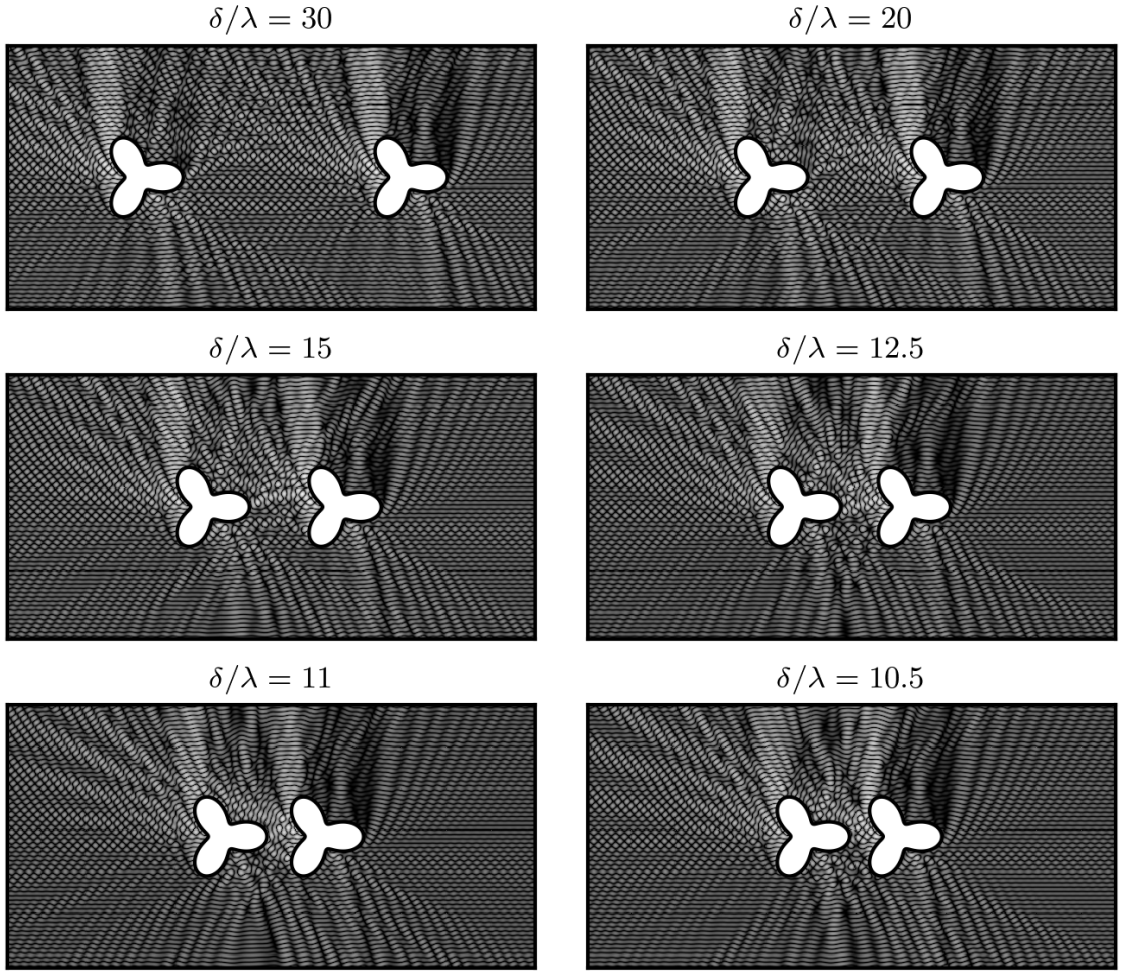
$$-\Delta\varphi = \frac{1}{\varepsilon_1}\sum_i q_i \delta\left(\mathbf{r} - \mathbf{r}_i\right) \qquad \text{in } \Omega_1$$

$$[\varphi] = \left[\varepsilon \frac{\partial\varphi}{\partial\nu}\right] = 0 \qquad \text{on } \Sigma$$

N	7612	19752
FMM/GMRES	12.6 s	26.9 s
RS precomp	151 s	592 s
RS solve	0.03 s	0.08 s

Break-even point: 10–25 solves

Multiple scattering



- Each object: 10λ

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

- FMM/GMRES with block preconditioner via RS

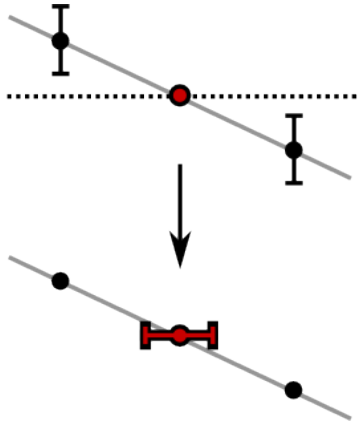
$$\begin{bmatrix} A_{11}^{-1} & \\ & A_{22}^{-1} \end{bmatrix}$$

- Unprecon: 700 iterations
- Precon: 10 iterations
- $50\times$ speedup

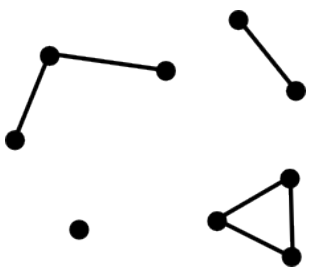
Rigid-body "docking"

pK_a algorithm

- Protein preparation
- Matrix precomputation
  - Compress/factor
- Energy calculation
- Monte Carlo sampling
  - Reduced site approximation
  - Multi-site cluster moves
- Estimate pK_i
  - Error bars



Apply delta method.

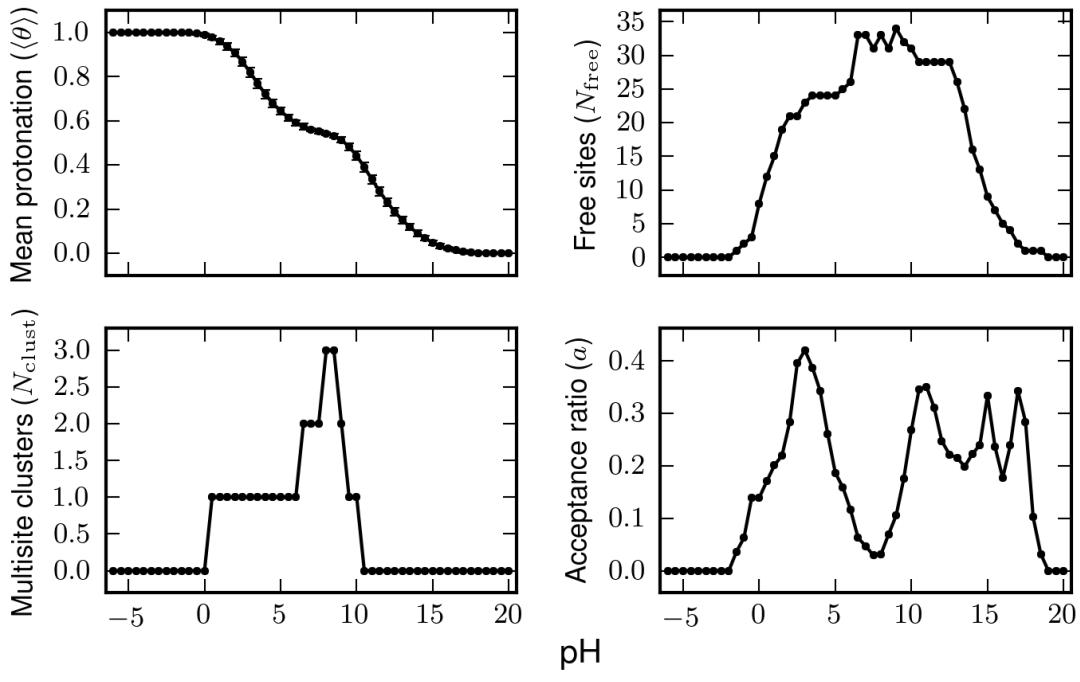


- Link sites by interaction energy
- Clusters: connected components
- Modify one cluster at random
- Pick move distance from geometric distribution

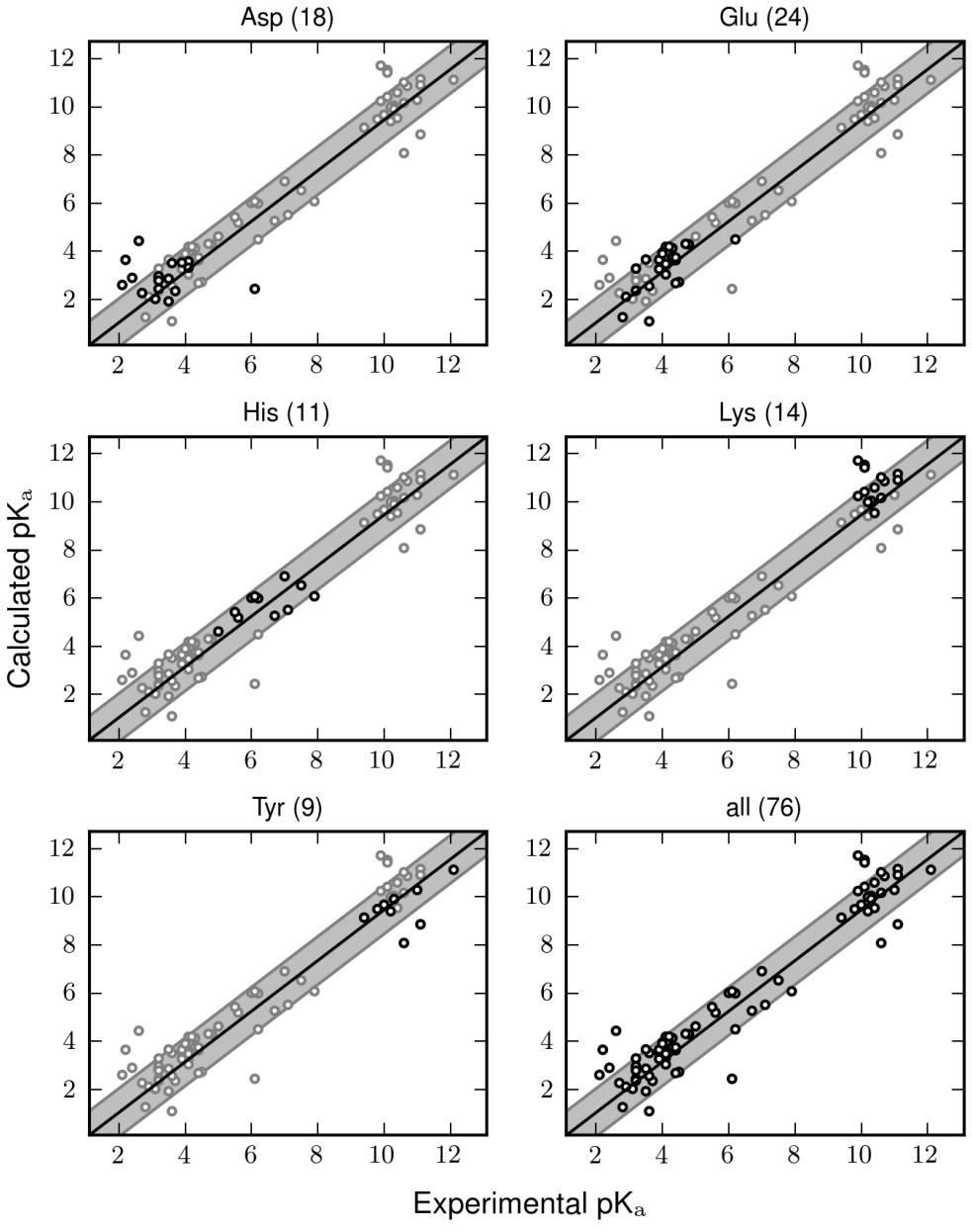
pK_a results: computational

name	PDB ID	residues	atoms	sites
BPTI	4PTI	58	891	18
OMTKY3	2OVO	56	813	15
HEWL	2LZT	129	1965	30
RNase A	3RN3	124	1865	34
RNase H	2RN2	155	2474	53

- DoFs: 10,000–30,000
- Precomp time: 1–2 hr
- Energy calc time: 10 s
- Much less memory than classical direct methods
- Much faster solves than iterative methods
- Precomp still expensive



pK_a results: biological



RMSD	protein dielectric 4	8	20
BPTI	1.47	0.96	0.82
OMTKY3	1.77	1.07	1.09
HEWL	2.52	1.49	0.79
RNase A	3.22	2.25	0.85
RNase H	4.53	2.53	1.36

type	err ≤ 1	RMSD
Arg	12 / 18	1.23
Glu	17 / 24	1.00
His	8 / 11	0.92
Lys	11 / 14	0.79
Tyr	7 / 9	1.24
all	55 / 76	1.05

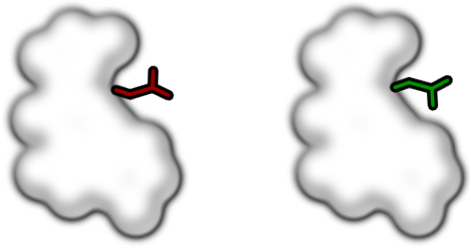
Summary

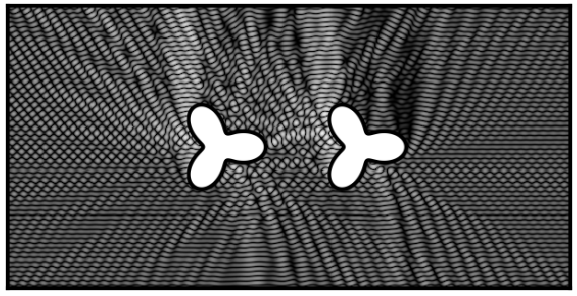
Main results:

- Can efficiently treat large numbers of titrating sites
- Similar accuracy as other Poisson-Boltzmann methods

Future improvements:

- Faster $\mathcal{O}(N \log N)$ direct solvers (forthcoming)
- Model conformational flexibility (Gunner et al.)
  - Low-rank matrix updates





Generalizations:

- Structure prediction: fixed backbone, rotamer optimization
- Docking: like multiple scattering
- Molecular dynamics (solvent boundary potential)
- Nonlocal electrostatics (Hildebrandt, Bardhan et al.)

References

pK_a calculations:

- Alexov E, Mehler EL, Baker N, Baptista AM, Huang Y, Milletti F, Nielsen JE, Farrell D, Carstensen T, Olsson MHM, Shen JK, Warwicker J, Williams S, Word JM (2011) Progress in the prediction of pK_a values in proteins. Proteins 79: 3260–3275.
- Bashford D, Karplus M (1990) pK_a's of ionizable groups in proteins: atomic detail from a continuum electrostatic model. Biochemistry 29: 10219–10225.
- Juffer AH, Argos P, Vogel HJ (1997) Calculating acid-dissociation constants of proteins using the boundary element method. J Phys Chem B 101: 7664–7673.

Fast solvers:

- Greengard L, Gueyffier D, Martinsson P-G, Rokhlin V (2009) Fast direct solvers for integral equations in complex three-dimensional domains. Acta Numer 18: 243–275.
- Ho KL, Greengard L (2012) A fast direct solver for structured linear systems by recursive skeletonization. SIAM J Sci Comput, to appear.
- Zhang B, Lu B, Cheng X, Huang J, Pitsianis N, Sun X, McCammon JA (2012) Mathematical and numerical aspects of the adaptive fast multipole Poisson-Boltzmann solver. Commun Comput Phys, in press.

Ho KL (2012) Fast direct methods for molecular electrostatics. PhD thesis, New York Univ.