CBMS Conference on Fast Direct Solvers

Dartmouth College

June 23 – June 27, 2014

Lecture 10: Scattering matrices

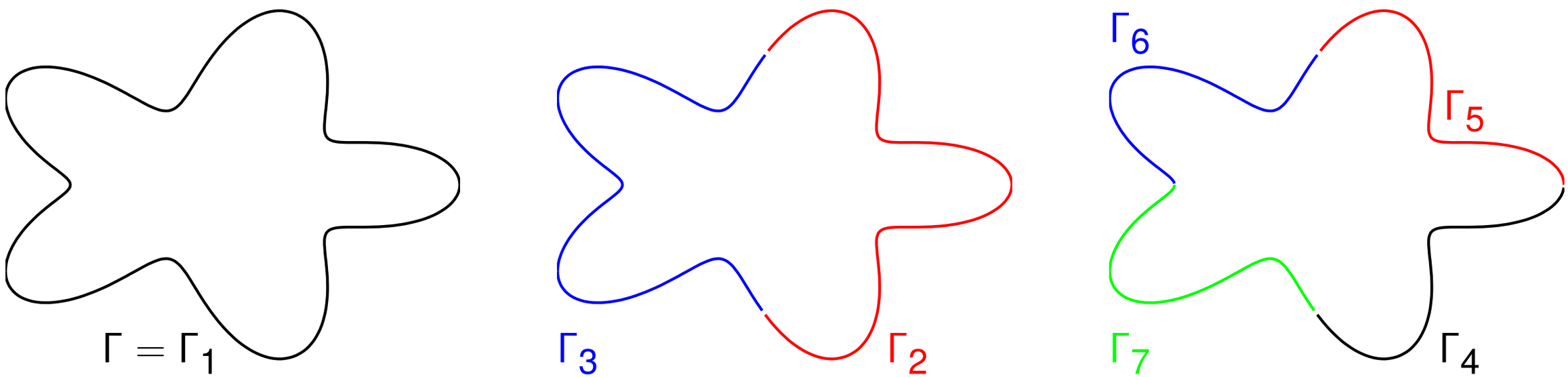
Gunnar Martinsson

The University of Colorado at Boulder

Research support by:

Recall from the last lecture (#9) that the direct solvers for integral equations are based on a hierarchical tree structure on the domain.

To illustrate, consider a BIE defined on a contour $\Gamma \subset \mathbb{R}^2$.



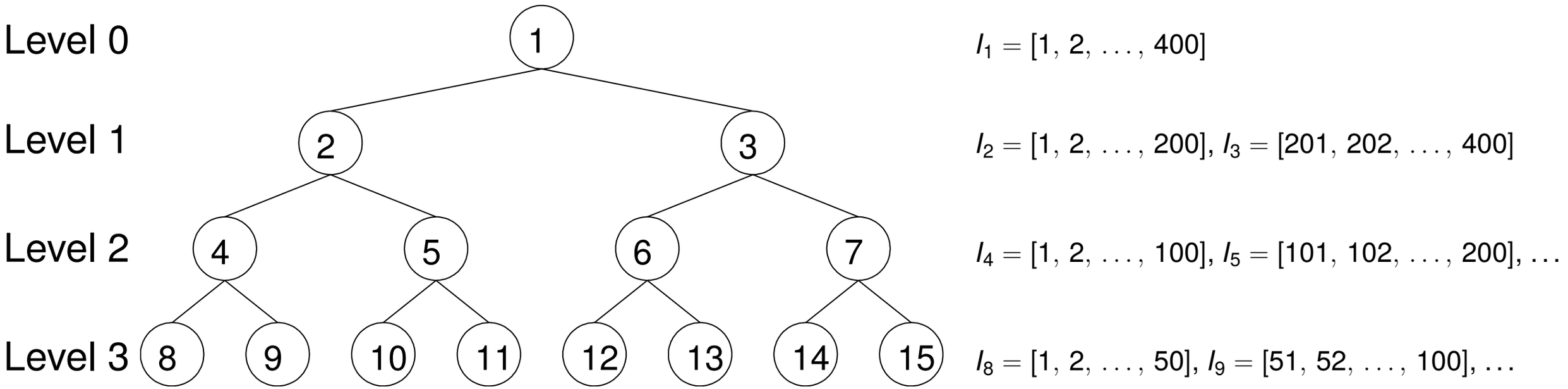
Let $\Gamma = \Gamma_1$ denote the root of a tree.

Partition Γ_1 into two pieces $\Gamma_1 = \Gamma_2 \cup \Gamma_3$.

Further partition $\Gamma_2 = \Gamma_4 \cup \Gamma_5$ and $\Gamma_3 = \Gamma_6 \cup \Gamma_7$.

The tree partitioning of the physical domain corresponds to a partitioning of the index vector $I = [1, 2, 3, \ldots, N]$.

For instance, if $N = 400$, and we use a tree with 4 levels, and split the index vector by halves each time, we get:

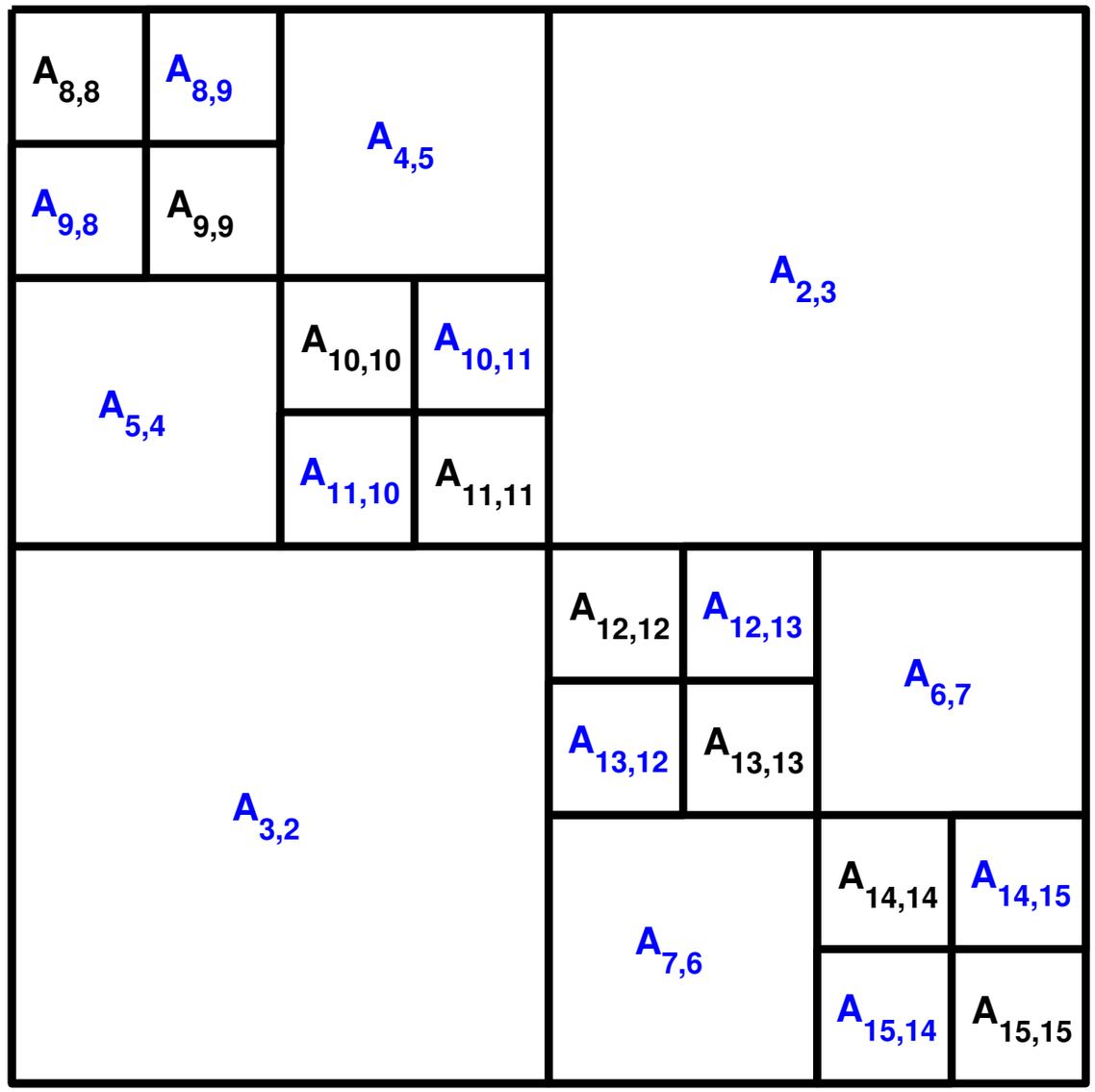


Note: *This simplistic illustration would be accurate for a simple curve.*

For complicated curves, for surfaces/volumes, etc, the index vectors are not contiguous.

The key is to subdivide based on locations $\{\boldsymbol{x}_i\}_{i=1}^{N}$ *in physical space.*

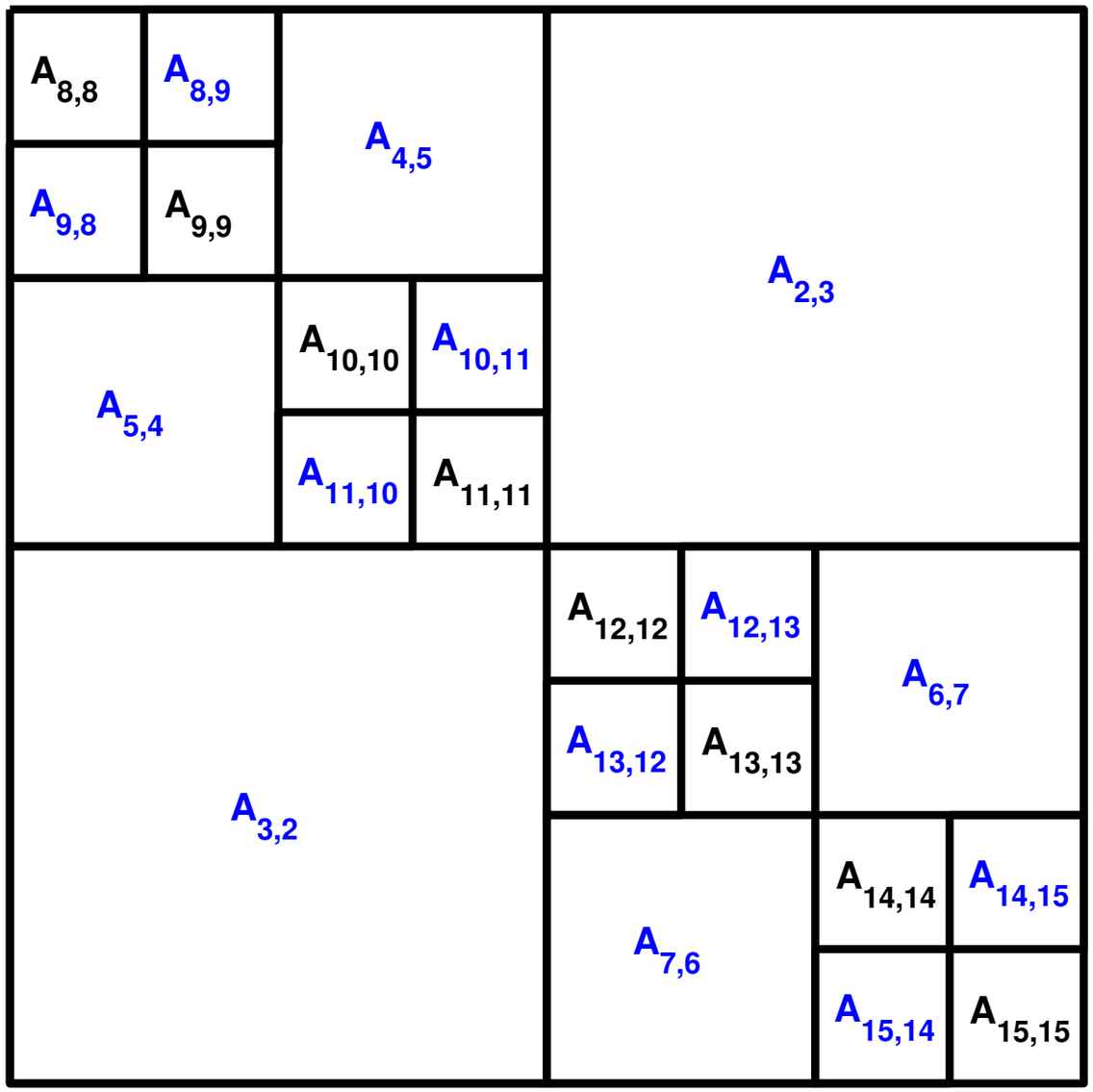
We earlier (see Lecture #4) defined the a rank-structured matrix format we informally called "$\mathcal{S}$-matrices" based on tessellations like the following:



We require that:

- the diagonal blocks are of size at most $2k \times 2k$
- the off-diagonal blocks (in blue in the figure) have rank at most k.

We earlier (see Lecture #4) defined the a rank-structured matrix format we informally called "$\mathcal{S}$-matrices" based on tessellations like the following:



We require that:

- the diagonal blocks are of size at most $2k \times 2k$
- the off-diagonal blocks (in blue in the figure) have rank at most k.

To formalize, we require that for every sibling pair $\{\sigma, \nu\}$, we can factor the associated off-diagonal blocks as:

$$\underset{n_\sigma \times n_\nu}{\mathbf{A}(I_\sigma, I_\nu)} = \underset{n_\sigma \times k}{\mathbf{U}_\sigma^{\text{long}}} \; \underset{k \times k}{\tilde{\mathbf{A}}_{\sigma,\nu}} \; \underset{k \times n_\nu}{(\mathbf{V}_\nu^{\text{long}})^*} \quad \text{and} \quad \underset{n_\sigma \times n_\nu}{\mathbf{A}(I_\nu, I_\sigma)} = \underset{n_\sigma \times k}{\mathbf{U}_\nu^{\text{long}}} \; \underset{k \times k}{\tilde{\mathbf{A}}_{\nu,\sigma}} \; \underset{k \times n_\nu}{(\mathbf{V}_\sigma^{\text{long}})^*}$$

Observe the sizes of the "long" matrices!

A drawback of the S-matrix format is that it is very costly to store (and operate on) the "long" basis vectors.

An HBS matrix can be viewed as a special case of an S-matrix, that satisfies a more rigorous requirement on the basis matrices — namely, that they can be expressed "hierarchically" — this is much more economical!

Definition: We say that an S-matrix is a Hierarchically Block Separable (HBS) matrix if for any parent-node τ with children σ_1 and σ_2, we can express the "long" basis matrices $\mathbf{U}_\tau^{\text{long}}$ and $\mathbf{V}_\tau^{\text{long}}$ in terms of the long basis matrices of their children:

$$\underset{n_\tau \times k}{\mathbf{U}_\tau^{\text{long}}} = \underset{n_\tau \times 2k}{\begin{bmatrix} \mathbf{U}_{\sigma_1}^{\text{long}} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_{\sigma_2}^{\text{long}} \end{bmatrix}} \underset{2k \times k}{\mathbf{U}_\tau}. \qquad \text{and} \qquad \underset{n_\tau \times k}{\mathbf{V}_\tau^{\text{long}}} = \underset{n_\tau \times 2k}{\begin{bmatrix} \mathbf{V}_{\sigma_1}^{\text{long}} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_{\sigma_2}^{\text{long}} \end{bmatrix}} \underset{2k \times k}{\mathbf{V}_\tau}.$$

For a leaf node τ, we simply define

$$\mathbf{U}_\tau^{\text{long}} = \mathbf{U}_\tau \qquad \text{and} \qquad \mathbf{V}_\tau^{\text{long}} = \mathbf{V}_\tau.$$

Example: Suppose boxes a and b are leaf nodes with parent e. Then we can express the "long" basis matrices of e as

$$\underset{n_e \times k}{\mathbf{U}_e^{\text{long}}} = \underset{n_e \times 2k}{\begin{bmatrix} \mathbf{U}_a & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_b \end{bmatrix}} \underset{2k \times k}{\mathbf{U}_e}.$$

Example: Suppose boxes a and b are leaf nodes with parent e. Then we can express the "long" basis matrices of e as

$$\underset{n_e \times k}{\mathbf{U}_e^{\text{long}}} = \underset{n_e \times 2k}{\begin{bmatrix} \mathbf{U}_a & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_b \end{bmatrix}} \underset{2k \times k}{\mathbf{U}_e}.$$

Next suppose further that a box f has children c and d, and that e and f are siblings with parent g (see blackboard!). Then the "long" basis matrix for g can be expressed

$$\underset{n_g \times k}{\mathbf{U}_g^{\text{long}}} = \underset{n_g \times 4k}{\begin{bmatrix} \mathbf{U}_a^{\text{long}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_b^{\text{long}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_c^{\text{long}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{U}_d^{\text{long}} \end{bmatrix}} \underset{4k \times 2k}{\begin{bmatrix} \mathbf{U}_e & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_f \end{bmatrix}} \underset{2k \times k}{\mathbf{U}_g}.$$

Example: Suppose boxes a and b are leaf nodes with parent e. Then we can express the "long" basis matrices of e as

$$\underset{n_e \times k}{\mathbf{U}_e^{\text{long}}} = \underset{n_e \times 2k}{\begin{bmatrix} \mathbf{U}_a & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_b \end{bmatrix}} \underset{2k \times k}{\mathbf{U}_e}.$$

Next suppose further that a box f has children c and d, and that e and f are siblings with parent g (see blackboard!). Then the "long" basis matrix for g can be expressed

$$\underset{n_g \times k}{\mathbf{U}_g^{\text{long}}} = \underset{n_g \times 4k}{\begin{bmatrix} \mathbf{U}_a^{\text{long}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_b^{\text{long}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_c^{\text{long}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{U}_d^{\text{long}} \end{bmatrix}} \underset{4k \times 2k}{\begin{bmatrix} \mathbf{U}_e & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_f \end{bmatrix}} \underset{2k \times k}{\mathbf{U}_g}.$$

Key point: For each node τ you can store the "small" basis matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ instead of the "long" basis matrices $\mathbf{U}_\tau^{\text{long}}$ and $\mathbf{V}_\tau^{\text{long}}$.

HBS MATVEC: *Given vector* $\mathbf{q}$ *and matrix* $\mathbf{A}$ *in HBS format, compute* $\mathbf{u} = \mathbf{A}\,\mathbf{q}$.

loop over all leaf boxes τ

$\hat{\mathbf{q}}_\tau = \mathbf{V}_\tau^*\,\mathbf{q}(I_\tau)$.

end loop

loop over levels, finer to coarser, $\ell = L-1,\ L-2,\ \ldots,\ 1$

loop over all parent boxes τ on level ℓ,

$$\hat{\mathbf{q}}_\tau = \mathbf{V}_\tau^* \begin{bmatrix} \hat{\mathbf{q}}_{\sigma_1} \\ \hat{\mathbf{q}}_{\sigma_2} \end{bmatrix}, \text{ where } \{\sigma_1, \sigma_2\} \text{ are the children of } \tau.$$

end loop

end loop

$\hat{\mathbf{u}}_1 = 0$

loop over all levels, coarser to finer, $\ell = 1,\ 2,\ \ldots,\ L-1$

loop over all parent boxes τ on level ℓ

$$\begin{bmatrix} \hat{\mathbf{u}}_{\sigma_1} \\ \hat{\mathbf{u}}_{\sigma_2} \end{bmatrix} = \mathbf{U}_\tau\,\hat{\mathbf{u}}_\tau + \begin{bmatrix} 0 & \mathbf{B}_{\sigma_1,\sigma_2} \\ \mathbf{B}_{\sigma_2,\sigma_1} & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{q}}_{\sigma_1} \\ \hat{\mathbf{q}}_{\sigma_2} \end{bmatrix}, \text{ where } \{\sigma_1, \sigma_2\} \text{ are the children of } \tau.$$

end loop

end loop

loop over all leaf boxes τ

$\mathbf{u}(I_\tau) = \mathbf{U}_\tau\,\hat{\mathbf{u}}_\tau + \mathbf{D}_\tau\,\mathbf{q}(I_\tau)$.

end loop

Comparing the HBS matvec to the FMM we see that:

- The interaction list with 27 elements is replaced by a single sibling.
- Interactions between touching neighbors is compressed.
- Ranks are higher —substantially so for problems that are not 1D!

Comparing the HBS matvec to the FMM we see that:

- The interaction list with 27 elements is replaced by a single sibling.
- Interactions between touching neighbors is compressed.
- Ranks are higher —substantially so for problems that are not 1D!

Non-nested bases:	***Nested bases:***
Barnes-Hut	Fast Multipole Method
$\mathcal{S}$-matrices	HBS (HSS) matrices
$\mathcal{H}$-matrices	$\mathcal{H}^2$-matrices

Comparing the HBS matvec to the FMM we see that:

- The interaction list with 27 elements is replaced by a single sibling.
- Interactions between touching neighbors is compressed.
- Ranks are higher —substantially so for problems that are not 1D!

Non-nested bases:	***Nested bases:***
Barnes-Hut	Fast Multipole Method
$\mathcal{S}$-matrices	HBS (HSS) matrices
$\mathcal{H}$-matrices	$\mathcal{H}^2$-matrices

We will next re-derive the HBS direct solver, using outgoing and incoming expansions.

Definition of the outgoing expansion

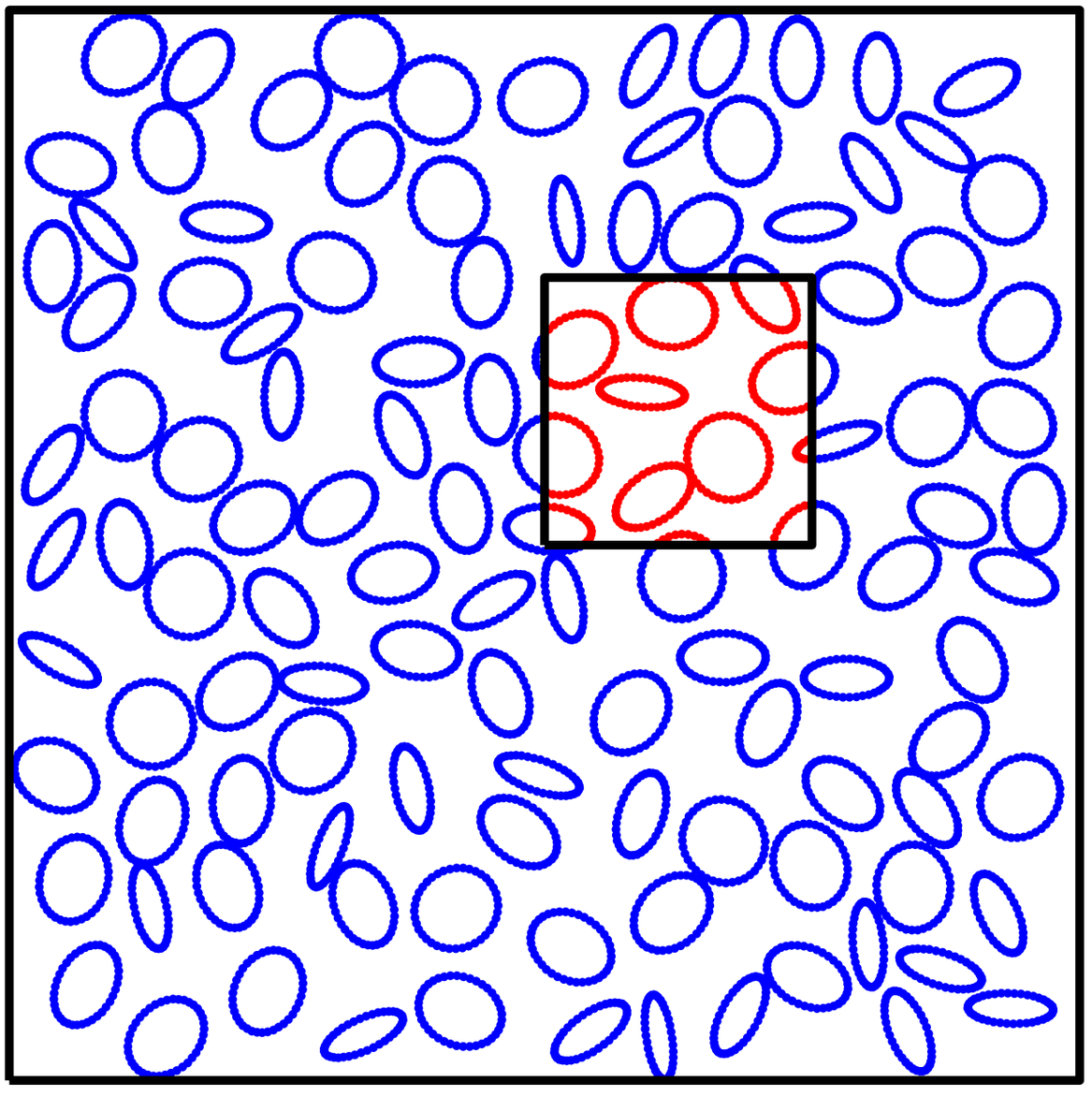
Consider an equilibrium equation $\mathbf{Aq} = \mathbf{f}$.

For a leaf node τ, define its *outgoing expansion* via $\tilde{\mathbf{q}}_\tau = \mathbf{V}_\tau^* \mathbf{q}(I_\tau)$.

The outgoing expansion encodes all information about the sources in Ω_τ that is required to evaluate the potential *anywhere* in $\Omega_\tau^c = \Omega \backslash \Omega_\tau$.

Observe that this is different from how the FMM does things. There is no "buffer zone" and no condition on boxes being well-separated.

This definition leads to *simpler inversion,* but also *higher ranks*, and necessitates *numerical compression.*



Ω_τ in red, Ω_τ^c in blue.

For a parent node τ with children $\{\alpha, \beta\}$ the outgoing expansion is $\tilde{\mathbf{q}}_\tau = \mathbf{V}_\tau^* \begin{bmatrix} \tilde{\mathbf{q}}_\alpha \\ \tilde{\mathbf{q}}_\beta \end{bmatrix}$.

Definition of the incoming expansion: Consider an equilibrium equation $\mathbf{Aq} = \mathbf{f}$.

For any node τ, define the *incoming potential* $\mathbf{g}_\tau$ via

$$\mathbf{g}_\tau = \mathbf{A}(I_\tau, I_\tau^{\mathrm{C}})\mathbf{q}(I_\tau^{\mathrm{C}}) = \mathbf{f}(I_\tau) - \mathbf{A}(I_\tau, I_\tau)\mathbf{q}(I_\tau).$$

$\mathbf{g}_\tau$ is the potential in τ, caused by charges $\mathbf{q}(I_\tau^{\mathrm{C}})$ in Ω_τ^{C}.

Then the local equilibrium equation on τ can be written

(1) $$\mathbf{A}(I_\tau, I_\tau)\mathbf{q}(I_\tau) + \mathbf{g}_\tau = \mathbf{f}(I_\tau).$$

The *incoming expansion* $\tilde{\mathbf{g}}_\tau$ is a compact representation of $\mathbf{g}_\tau$.

Mathematically, the incoming expansion $\mathbf{g}_\tau$ is given by

$$\tilde{\mathbf{g}}_\tau = \mathbf{R}_\tau \, \mathbf{g}_\tau,$$

where $\mathbf{R}_\tau$ is simply restriction (subsampling!) to the skeleton nodes in τ. Then

$$\mathbf{g}_\tau = \mathbf{U}_\tau \tilde{\mathbf{g}}_\tau.$$

Our objective is now to derive a local equilibrium equation analogous to (1),
but involving the compact representations.

Compression of leaf equilibrium equation

Let τ be a leaf node. Set $\mathbf{A}_{\sigma,\tau} = \mathbf{A}(I_\sigma, I_\tau)$, $\mathbf{q}_\tau = \mathbf{q}(I_\tau)$, etc.

The equilibrium equation $\mathbf{Aq} = \mathbf{f}$ restricted to τ reads

$$\mathbf{A}_{\tau,\tau}\mathbf{q}_\tau + \underbrace{\mathbf{A}(I_\tau, I_\tau^{\mathrm{C}})\mathbf{q}(I_\tau^{\mathrm{C}})}_{=:\mathbf{g}_\tau} = \mathbf{f}_\tau.$$

Left multiply by $\mathbf{A}_{\tau,\tau}^{-1}$

$$\begin{array}{ccccccc} \mathbf{q}_\tau & + & \mathbf{A}_{\tau,\tau}^{-1} & \mathbf{g}_\tau & = & \mathbf{A}_{\tau,\tau}^{-1}\mathbf{f}_\tau. \\ n\times 1 & & n\times n & n\times 1 & & n\times 1 \end{array}$$

Left multiply by $\mathbf{V}_\tau^*$ and use that $\mathbf{g}_\tau = \mathbf{U}_\tau\tilde{\mathbf{g}}_\tau$,

$$\underbrace{\mathbf{V}_\tau^*\mathbf{q}_\tau}_{=:\tilde{\mathbf{q}}_\tau} + \underbrace{\mathbf{V}_\tau^*\mathbf{A}_{\tau,\tau}^{-1}\mathbf{U}_\tau}_{=:\mathbf{S}_\tau}\tilde{\mathbf{g}}_\tau = \underbrace{\mathbf{V}_\tau^*\mathbf{A}_{\tau,\tau}^{-1}\mathbf{f}_\tau}_{=:\tilde{\mathbf{r}}_\tau}.$$

We obtain the leaf equilibrium equation

$$\begin{array}{ccccccc} \tilde{\mathbf{q}}_\tau & + & \mathbf{S}_\tau & \tilde{\mathbf{g}}_\tau & = & \tilde{\mathbf{r}}_\tau. \\ k\times 1 & & k\times k & k\times 1 & & k\times 1 \end{array}$$

(Recall: $k < n$ and often $k \ll n$.)

Compare the uncompressed leaf equilibrium equation

$$\underset{n\times 1}{\mathbf{q}_\tau} + \underset{n\times n}{\mathbf{A}^{-1}_{\tau,\tau}}\ \underset{n\times 1}{\mathbf{g}_\tau} = \underset{n\times 1}{\mathbf{A}^{-1}_{\tau,\tau}\mathbf{f}_\tau},$$

to the compressed leaf equilibrium equation

$$\underset{k\times 1}{\tilde{\mathbf{q}}_\tau} + \underset{k\times k}{\mathbf{S}_\tau}\ \underset{k\times 1}{\tilde{\mathbf{g}}_\tau} = \underset{k\times 1}{\tilde{\mathbf{r}}_\tau}.$$

n is the number of nodes in Ω_τ

k is the rank of interaction between Ω_τ and Ω_τ^{c}

The point is that very often we have $k \ll n$.

The scattering matrix $\mathbf{S}_\tau$ encodes all the information about Ω_τ that is needed to process interactions with the outside world.

*The size of the scattering matrix $\mathbf{S}_\tau$ is determined by the rank of interaction between Γ_τ and the rest of the world, **not** by the number of nodes needed to resolve Γ_τ internally.*

Recall: The scattering matrix is defined by

$$\underset{k\times k}{\mathbf{S}_\tau} = \underset{k\times n}{\mathbf{V}_\tau^*}\ \underset{n\times n}{\left(\mathbf{A}(I_\tau, I_\tau)\right)^{-1}}\ \underset{n\times k}{\mathbf{U}_\tau}$$

Suppose that $I = I_1 \cup I_2$, and set

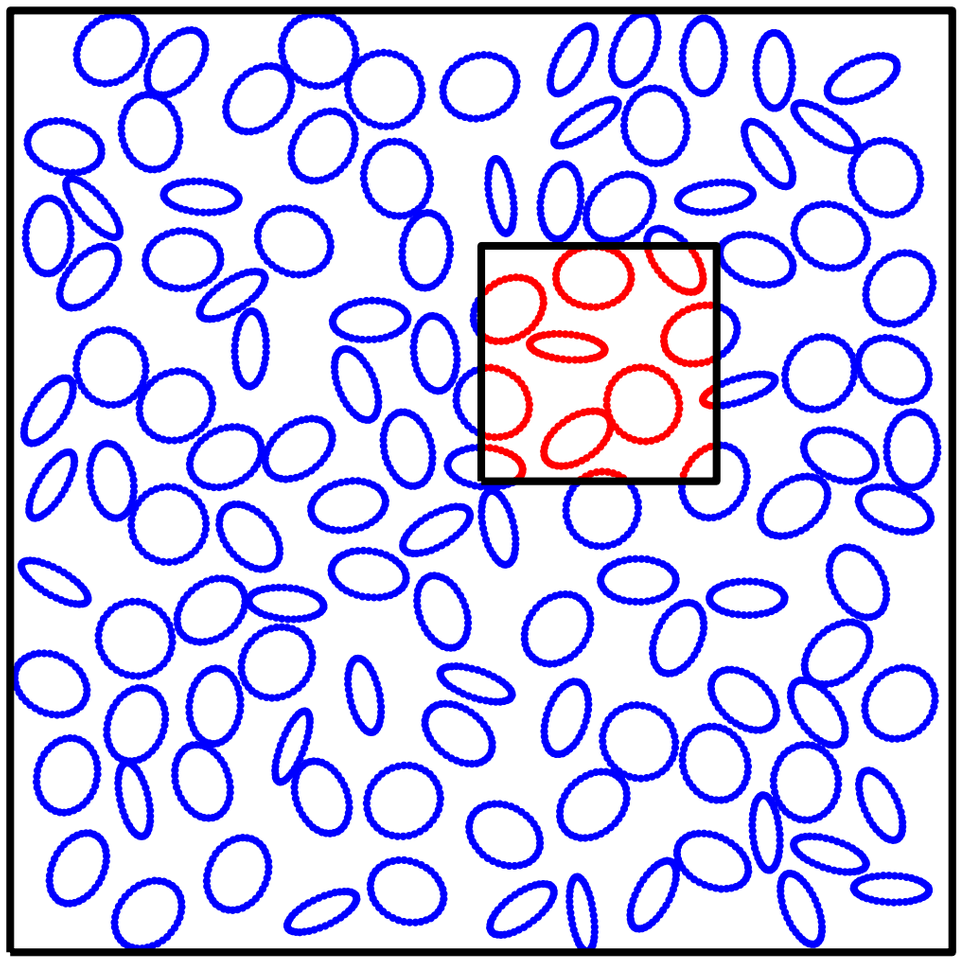
$$\mathbf{A}_{12} = \mathbf{A}(I_1, I_2)$$

$$\mathbf{A}_{21} = \mathbf{A}(I_2, I_1).$$

Then

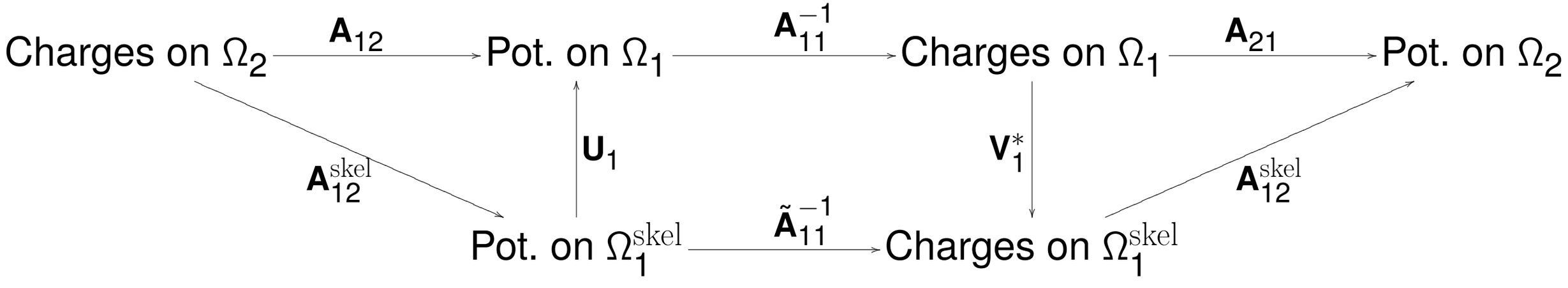
$$\mathbf{A}_{12} = \mathbf{U}_1 \mathbf{A}_{12}^{(\mathrm{skel})}$$

$$\mathbf{A}_{21} = \mathbf{A}_{21}^{(\mathrm{skel})} \mathbf{V}_1^*$$



I_1 in red, I_2 in blue.

We can visualize the "information flow" in a diagram



$\mathbf{S}_1$ contains *all the information the outside world needs to know about Ω_1*.

Recall: The scattering matrix is defined by

$$\underset{k\times k}{\mathbf{S}_\tau} = \underset{k\times n}{\mathbf{V}_\tau^*}\ \underset{n\times n}{\left(\mathbf{A}(I_\tau, I_\tau)\right)^{-1}}\ \underset{n\times k}{\mathbf{U}_\tau}$$

Suppose that $I = I_1 \cup I_2$, and set

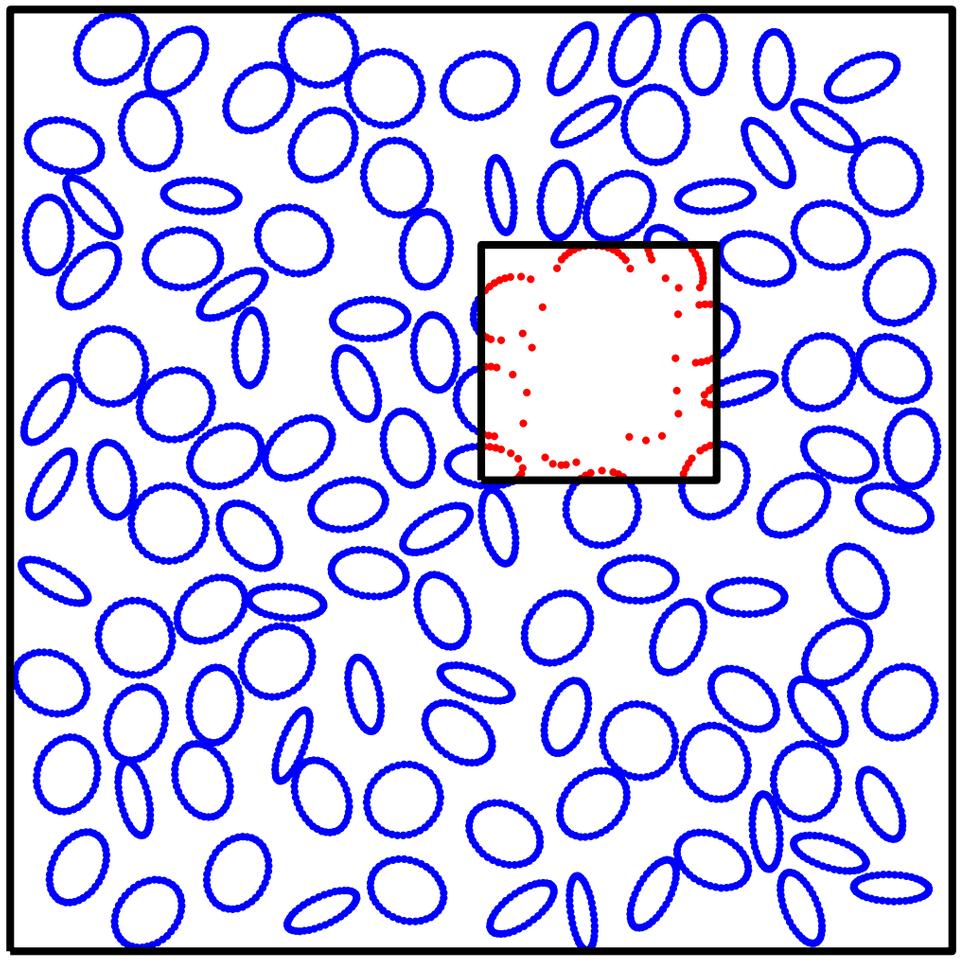
$$\mathbf{A}_{12} = \mathbf{A}(I_1, I_2)$$

$$\mathbf{A}_{21} = \mathbf{A}(I_2, I_1).$$

Then

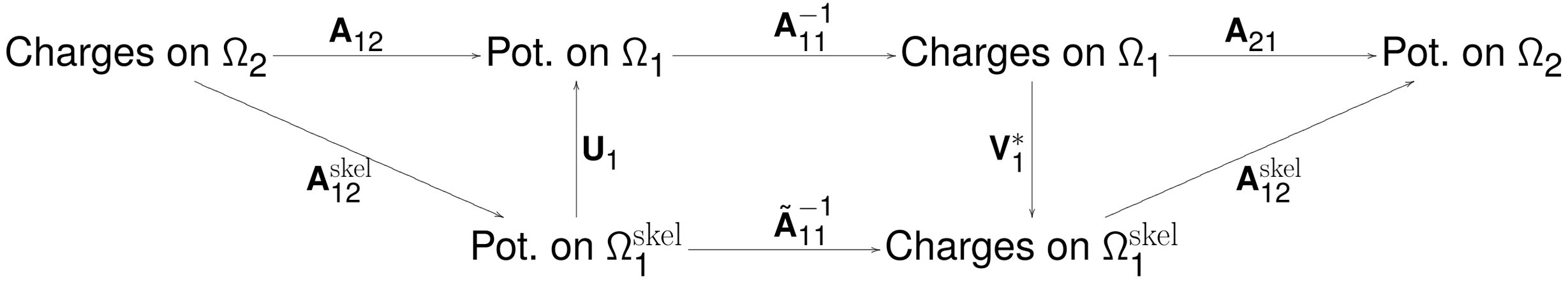
$$\mathbf{A}_{12} = \mathbf{U}_1 \mathbf{A}_{12}^{(\mathrm{skel})}$$

$$\mathbf{A}_{21} = \mathbf{A}_{21}^{(\mathrm{skel})} \mathbf{V}_1^*$$



$\tilde{I}_1$ *in red,* I_2 *in blue.*

We can visualize the "information flow" in a diagram

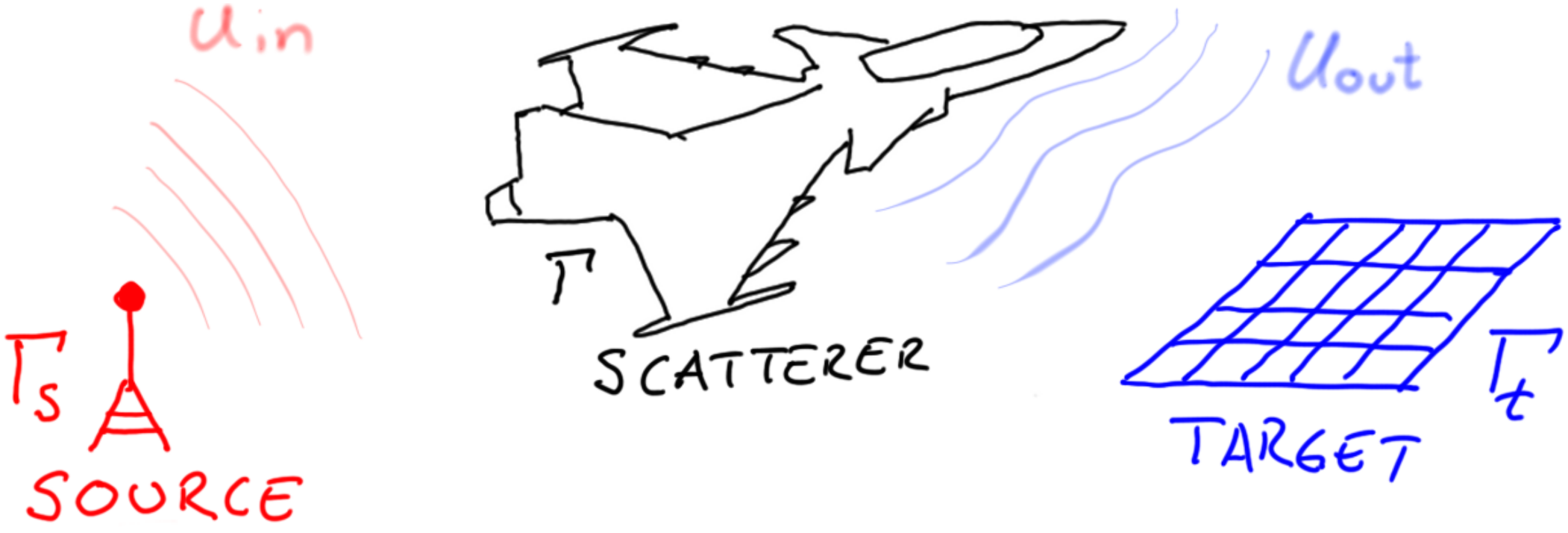


$\mathbf{S}_1$ contains *all the information the outside world needs to know about* Ω_1.

The classical concept of scattering matrices:

Consider a classical scattering problem:

- An "incident wave" u_{in} generated by charges on the radiation source Γ_{s} is given.
- The incident wave hits a "scatterer" Γ, and a "scattered wave" u_{out} is generated.
- The measured output is the scattered wave restricted to some "target" Γ_{t}.



The classical concept of scattering matrices:

The map we seek to evaluate can be expressed mathematically as

$$\underset{\text{outgoing field on } \Gamma_t}{u_{\text{out}}} = \underset{}{T_{\text{out}}\ T_\Gamma^{-1}\ T_{\text{in}}} \underset{\text{given charges on } \Gamma_s}{q}, \tag{2}$$

where $T_{\text{in}} : L^2(\Gamma_{\text{in}}) \to L^2(\Gamma)$ is the operator

$$[T_{\text{in}} q](\boldsymbol{x}) = \int_{\Gamma_{\text{in}}} S_\kappa(\boldsymbol{x}, \boldsymbol{y}) q(\boldsymbol{y}) ds(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Gamma,$$

where $T_\Gamma : L^2(\Gamma) \to L^2(\Gamma)$ is the "combined field" operator

$$[T_\Gamma g](\boldsymbol{x}) = \frac{1}{2} g(\boldsymbol{x}) + \int_\Gamma \big(D_\kappa(\boldsymbol{x}, \boldsymbol{y}) + i\kappa\, S_\kappa(\boldsymbol{x}, \boldsymbol{y})\big)\, q(\boldsymbol{y})\, ds(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Gamma, \tag{3}$$

and where $T_{\text{out}} : L^2(\Gamma) \to L^2(\Gamma_{\text{out}})$ is the operator

$$[T_{\text{out}} q](\boldsymbol{x}) = \int_\Gamma \big(D_\kappa(\boldsymbol{x}, \boldsymbol{y}) + i\kappa\, S_\kappa(\boldsymbol{x}, \boldsymbol{y})\big)\, q(\boldsymbol{y})\, ds(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Gamma_{\text{out}}.$$

The kernel functions are the standard single and double layer kernels, in 2D:

$$S_\kappa(\boldsymbol{x}, \boldsymbol{y}) = H_0(\kappa|\boldsymbol{x} - \boldsymbol{y}|), \qquad \text{and} \qquad D_\kappa(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{n}(\boldsymbol{y}) \cdot \nabla_{\boldsymbol{y}} H_0(\kappa|\boldsymbol{x} - \boldsymbol{y}|).$$

Observe that since T_Γ is an invertible second kind integral operator, its singular values do not cluster around zero.

The classical concept of scattering matrices:

The map we seek to evaluate can be expressed mathematically as

$$u_{\text{out}} = T_{\text{out}} T_{\Gamma}^{-1} T_{\text{in}} q. \tag{4}$$

The operator in the middle, T_Γ, is a second kind Fredholm operator. Moreover, T_{in} and T_{out} are not only compact, but have singular values that decay exponentially fast. This means that for any finite precision ε (e.g. $\varepsilon = 10^{-10}$), there exists a finite k (think $k = 100$ or $k = 1000$ or $k = 10000$) such that

$$||T_{\text{out}} - T_{\text{out}} P_{\text{out}}^{(k)}|| \leq \varepsilon \qquad \text{and} \qquad ||T_{\text{in}} - P_{\text{in}}^{(k)} T_{\text{in}}|| \leq \varepsilon,$$

where $P_{\text{in}}^{(k)}$ and $P_{\text{out}}^{(k)}$ denote the orthogonal projections onto the first k singular vectors of T_{in} and T_{out}, respectively. Consequently,

$$T_{\text{out}} T_{\Gamma}^{-1} T_{\text{in}} \approx T_{\text{out}} P_{\text{out}}^{(k)} T_{\Gamma}^{-1} P_{\text{in}}^{(k)} T_{\text{in}}.$$

The "scattering matrix" for Γ is now $S_\Gamma = P_{\text{out}}^{(k)} T_{\Gamma}^{-1} P_{\text{in}}^{(k)}$.

Classically, the operators $P_{\text{out}}^{(k)}$ and $P_{\text{in}}^{(k)}$ were determined via expansions in cylindrical (2D) or spherical (3D) harmonics.

Scattering matrices and Schur complements

Another way of viewing things is that Ω_1 interacts with Ω_2 *only through* $\mathbf{A}_{12}$ *and* $\mathbf{A}_{21}$.

In other words, we are only interested in the Schur complement

$$\mathbf{A}_{21}\,\mathbf{A}_{11}^{-1}\,\mathbf{A}_{12}.$$

Now insert the factorizations

$$\mathbf{A}_{12} = \mathbf{U}_1 \mathbf{A}_{12}^{(\mathrm{skel})} \qquad \text{and} \qquad \mathbf{A}_{21} = \mathbf{A}_{21}^{(\mathrm{skel})} \mathbf{V}_1^*$$

to obtain

$$\mathbf{A}_{21}\,\mathbf{A}_{11}^{-1}\,\mathbf{A}_{12} = \mathbf{A}_{21}^{(\mathrm{skel})} \underbrace{\mathbf{V}_1^*\,\mathbf{A}_{11}^{-1}\,\mathbf{U}_1}_{=:\mathbf{S}_1} \mathbf{A}_{12}^{(\mathrm{skel})}.$$

For reference, compare to the "physics" notion of a scattering matrix. We seek to evaluate the map

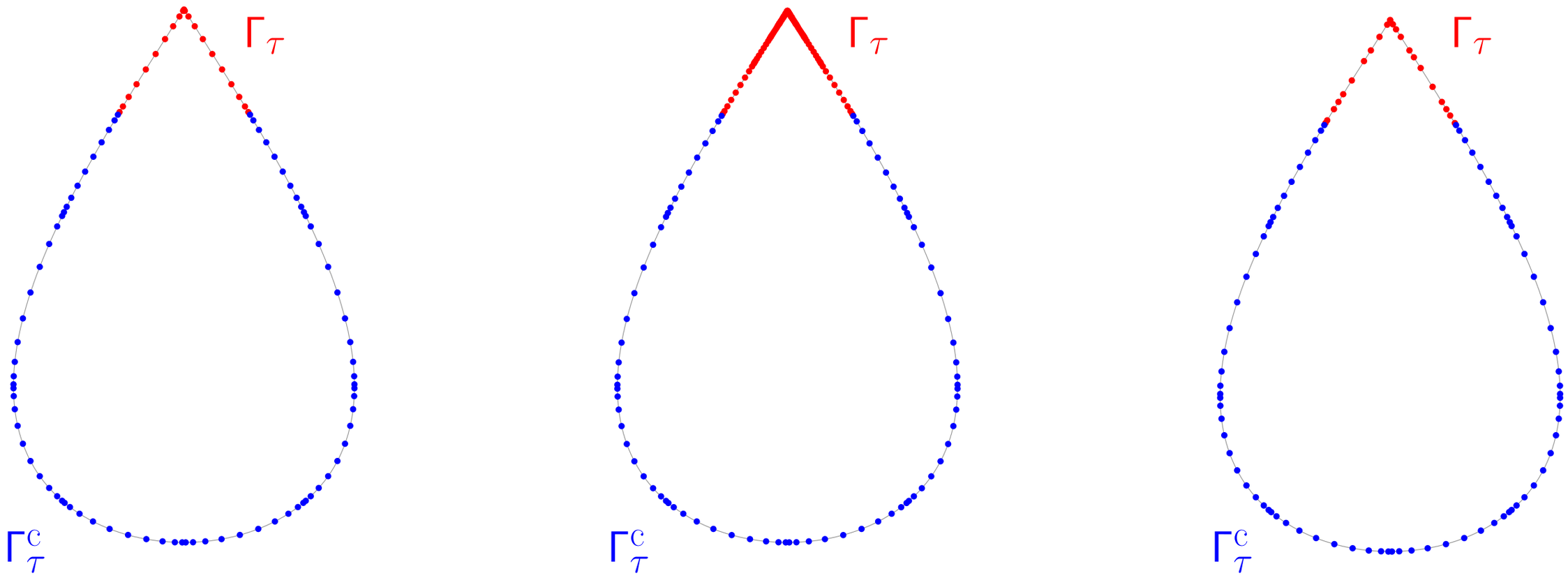
$$T_{\mathrm{out}}\,T_\Gamma^{-1}\,T_{\mathrm{in}}.$$

Using that T_{out} and T_{in} can be represented (typically via expansions in harmoics) we get

$$T_{\mathrm{out}}\,T_\Gamma^{-1}\,T_{\mathrm{in}} = T_{\mathrm{out}} \underbrace{P_{\mathrm{out}}\,T_\Gamma^{-1}\,P_{\mathrm{in}}}_{=S_\Gamma} T_{\mathrm{in}}.$$

Scattering matrices are extremely useful for handling corners and edges:

Consider a domain Γ discretized with Nyström based on a panel-based method using 10-point Gaussian quadrature. Suppose further that Γ has a corner, causing singularities in the layer potentials.

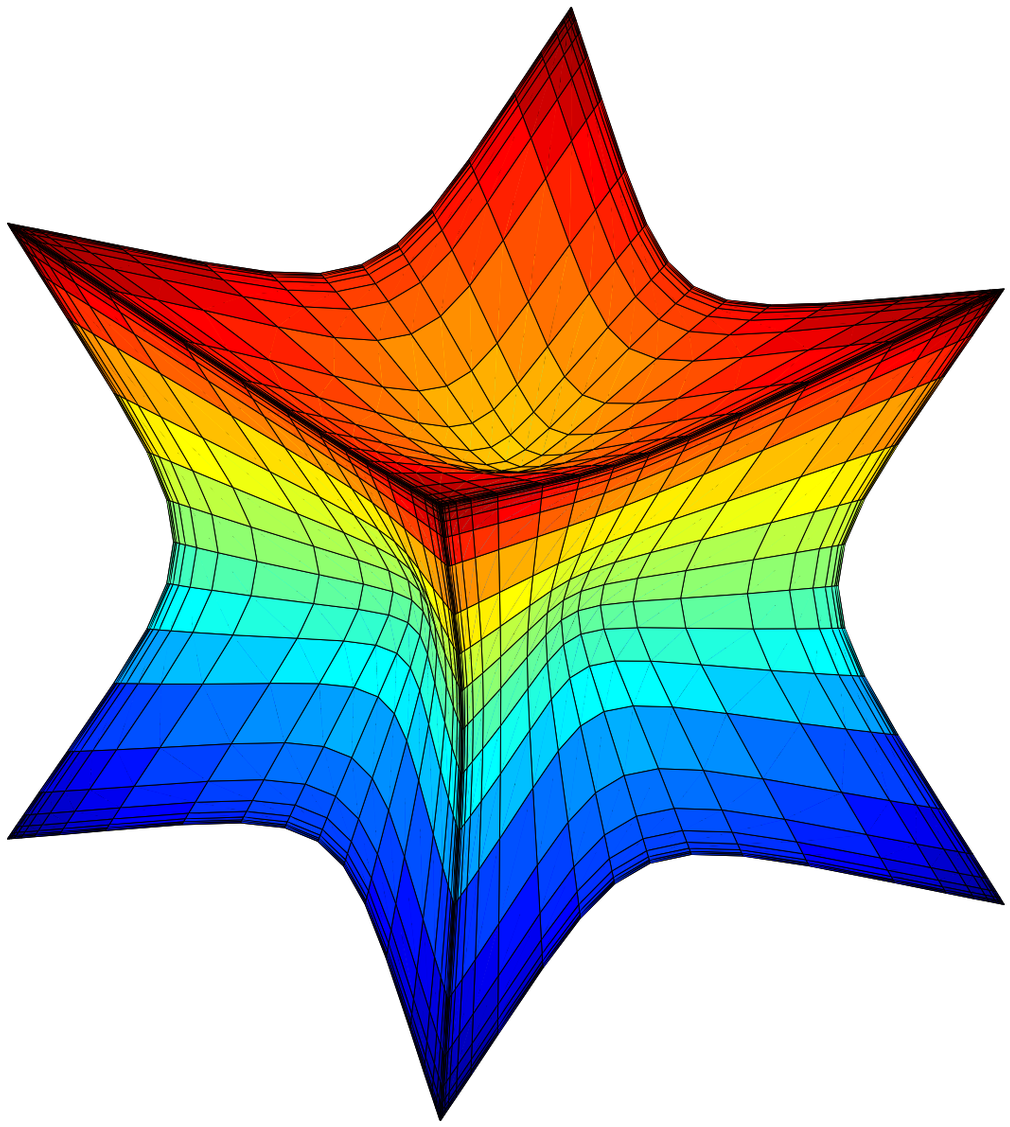


(a) (b) (c)

(a) Ignoring the singularity at the corner gives a small system but poor accuracy.

(b) Refining the grid near the corner gives great accuracy, but *N* gets very large.

(c) After local "compression" in which we compute a compressed scattering matrix for Γ_τ, we get the same accuracy as the discretization in (b), using as many nodes as in (a)!



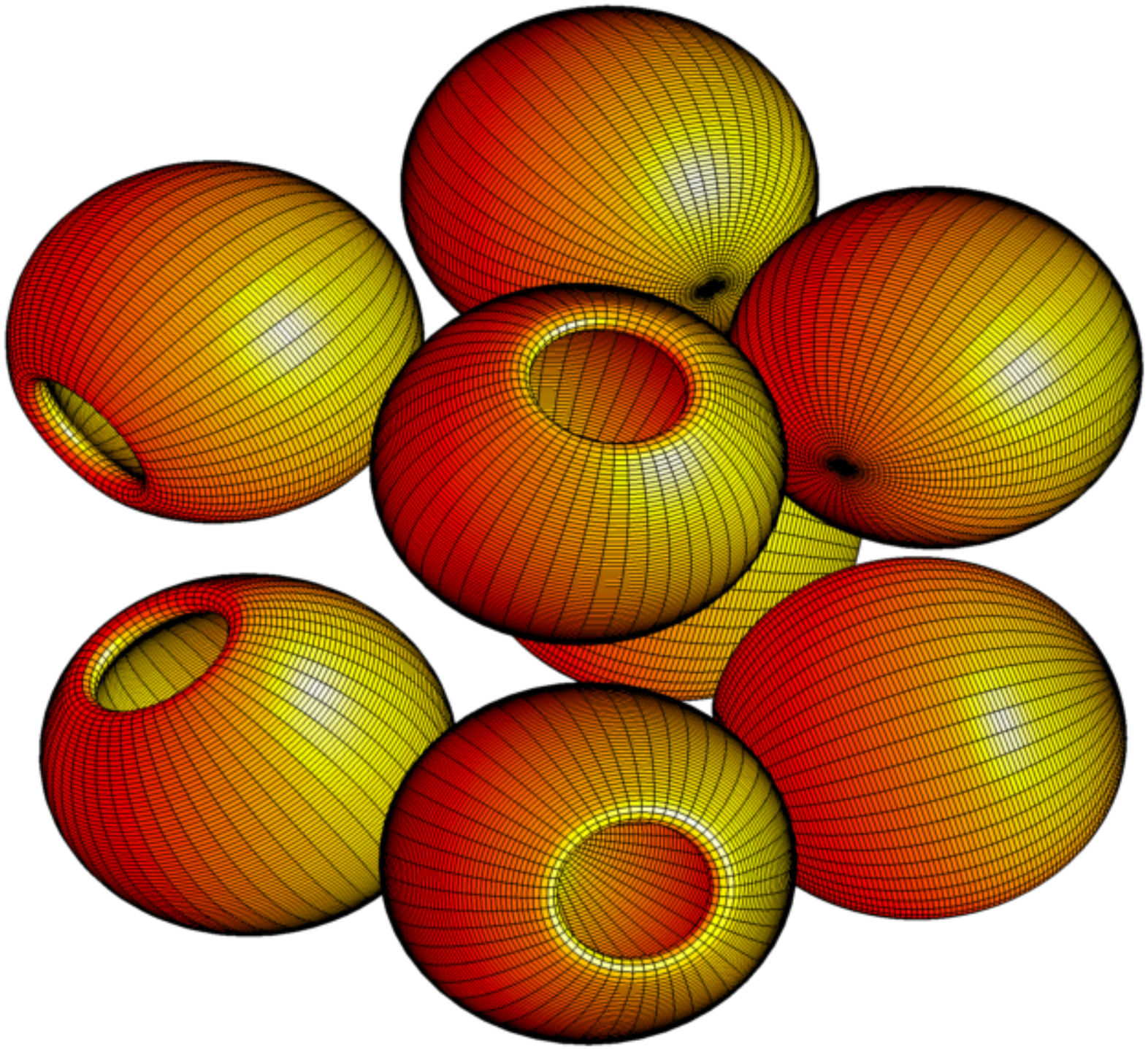
A surface Γ with corners and edges.

The grid has been refined to attain high accuracy.

Computing scattering matrices for the corners is conceptually easy (but laborious).

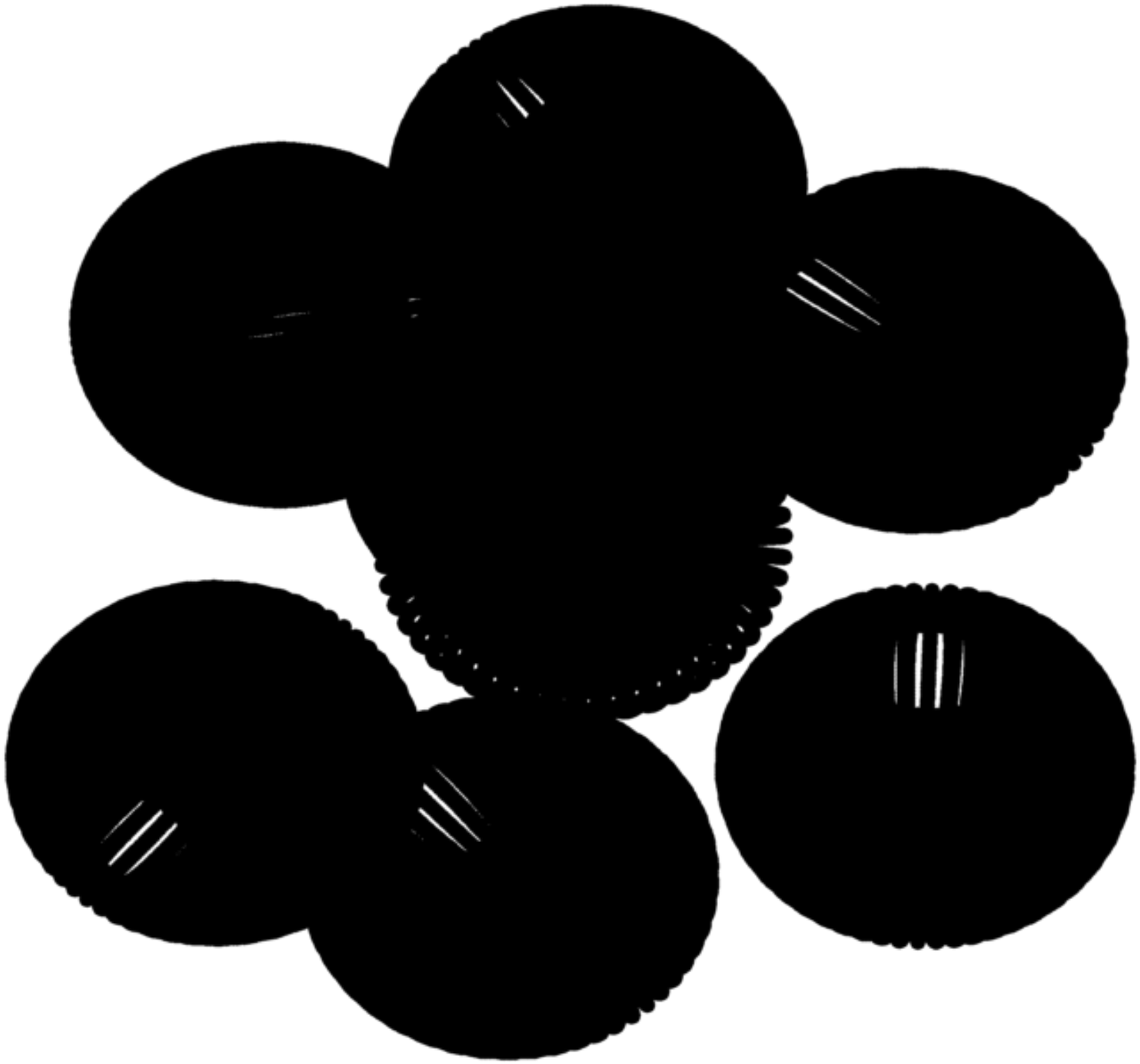
Compressing the edges takes effort!

Example of efficient encoding of internal information: Multibody scattering



Consider scattering from some complicated multibody domain.

Example of efficient encoding of internal information: Multibody scattering



There are lots of discretization nodes involved. Very computationally intense!

Example of efficient encoding of internal information: Multibody scattering

After local compression of each scatter, the problem is much more tractable.

Recall compression of a leaf node τ:

The *incoming potential* is the vector

$$\mathbf{g}_\tau = \mathbf{A}(I_\tau, I_\tau^c)\,\mathbf{q}(I_\tau^c) = \mathbf{f}(I_\tau) - \mathbf{A}(I_\tau, I_\tau)\,\mathbf{q}(I_\tau).$$

The *incoming expansion* is a compact representation $\tilde{\mathbf{g}}_\tau$ of $\mathbf{g}_\tau$,

$$\mathbf{g}_\tau = \mathbf{U}_\tau\,\tilde{\mathbf{g}}_\tau.$$

The uncompressed equilibrium equation is

$$\begin{array}{ccccc} \mathbf{q}(I_\tau) & + & \mathbf{A}(I_\tau, I_\tau)^{-1} & \mathbf{g}_\tau & = \mathbf{A}(I_\tau, I_\tau)^{-1}\,\mathbf{f}(I_\tau). \\ n \times 1 & & n \times n & n \times 1 & n \times 1 \end{array}$$

The compressed equilibrium equation is

$$\begin{array}{ccccc} \tilde{\mathbf{q}}_\tau & + & \mathbf{S}_\tau & \tilde{\mathbf{g}}_\tau & = \tilde{\mathbf{r}}_\tau, \\ k \times 1 & & k \times k & k \times 1 & k \times 1 \end{array}$$

where $k < n$, and where

$$\tilde{\mathbf{q}}_\tau = \mathbf{V}_\tau^* \mathbf{q}(I_\tau), \quad \tilde{\mathbf{r}}_\tau = \mathbf{V}_\tau^* \mathbf{A}(I_\tau, I_\tau)^{-1} \mathbf{f}(I_\tau), \quad \mathbf{S}_\tau = \mathbf{V}_\tau^* \mathbf{A}(I_\tau, I_\tau)^{-1} \mathbf{U}_\tau.$$

The direct solver can be viewed as the hierarchical construction of scattering matrices.

Let τ be a parent node. The scattering matrix for τ is then defined via

$$\begin{array}{ccccc} \mathbf{S}_\tau & = & \left(\mathbf{V}_\tau^{\text{long}}\right)^* & \mathbf{A}(I_\tau, I_\tau)^{-1} & \left(\mathbf{U}^{\text{long}}\right)_\tau. \\ k \times k & & k \times N_\tau & N_\tau \times N_\tau & N_\tau \times k \end{array}$$

Expensive to compute using definition, since N_τ (the number of nodes *originally* inside box τ) can be very long.

Theorem: Let τ be a node with children α and β. Then

$$\begin{array}{ccccccc} \mathbf{S}_\tau & = & \mathbf{V}_\tau^* & \begin{bmatrix} \mathbf{I} & \mathbf{S}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \mathbf{S}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{I} \end{bmatrix}^{-1} & \begin{bmatrix} \mathbf{S}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_\beta \end{bmatrix} & \mathbf{U}_\tau. \\ k \times k & & k \times 2k & 2k \times 2k & 2k \times 2k & 2k \times k. \end{array} \tag{5}$$

In other words, if $\mathbf{S}_\alpha$ and $\mathbf{S}_\beta$ are known, then $\mathbf{S}_\tau$ can be computed cheaply.

Proof: Using the formula for the factorization of sibling interaction matrices we get

$$\begin{aligned}
\mathbf{A}(I_\tau, I_\tau) &= \begin{bmatrix} \mathbf{A}_{\alpha,\alpha} & \mathbf{A}_{\alpha,\beta} \\ \mathbf{A}_{\beta,\alpha} & \mathbf{A}_{\beta,\beta} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\alpha,\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \hat{\mathbf{U}}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \hat{\mathbf{V}}_\beta^* \\ \hat{\mathbf{U}}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} \hat{\mathbf{V}}_\alpha^* & \mathbf{0} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_{\alpha,\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \hat{\mathbf{U}}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \hat{\mathbf{U}}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{V}}_\alpha^* & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{V}}_\beta^* \end{bmatrix} \\
&= \left(\mathbf{I} + \begin{bmatrix} \mathbf{0} & \hat{\mathbf{U}}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \hat{\mathbf{U}}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{V}}_\alpha^* \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{V}}_\beta^* \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix} \right) \begin{bmatrix} \mathbf{A}_{\alpha,\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta} \end{bmatrix} \\
&= (\mathbf{I} + \mathbf{E}\mathbf{F}^*) \begin{bmatrix} \mathbf{A}_{\alpha,\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta} \end{bmatrix}, \qquad (6)
\end{aligned}$$

where we defined $\mathbf{E} = \begin{bmatrix} \mathbf{0} & \hat{\mathbf{U}}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \hat{\mathbf{U}}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{0} \end{bmatrix}$ and $\mathbf{F}^* = \begin{bmatrix} \hat{\mathbf{V}}_\alpha^* \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{V}}_\beta^* \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix}$.

Formula (6) shows that $\mathbf{A}(I_\tau, I_\tau)$ can be written as a product between a low-rank perturbation to the identity, and a block-diagonal matrix. The Woodbury formula for inversion of a low-rank perturbation of the identity states that $(\mathbf{I} + \mathbf{E}\mathbf{F}^*)^{-1} = \mathbf{I} - \mathbf{E}(\mathbf{I} + \mathbf{F}^*\mathbf{E})^{-1}\mathbf{F}^*$. To exploit this formula, we first define

$$\mathbf{Z}_\tau \overset{\text{def}}{=} (\mathbf{I} + \mathbf{F}^*\mathbf{E})^{-1} = \left(\mathbf{I} + \begin{bmatrix} \mathbf{0} & \hat{\mathbf{V}}_\alpha^* \mathbf{A}_{\alpha,\alpha}^{-1} \hat{\mathbf{U}}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \hat{\mathbf{V}}_\beta^* \mathbf{A}_{\beta,\beta}^{-1} \hat{\mathbf{U}}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & 0 \end{bmatrix} \right)^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{S}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \mathbf{S}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{I} \end{bmatrix}^{-1}. \qquad (7)$$

Then

$$\mathbf{A}(I_\tau, I_\tau)^{-1} = \begin{bmatrix} \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix} \left(\mathbf{I} - \begin{bmatrix} \mathbf{0} & \hat{\mathbf{U}}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \hat{\mathbf{U}}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{0} \end{bmatrix} \mathbf{Z}_\tau \begin{bmatrix} \hat{\mathbf{V}}_\alpha^* \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{V}}_\beta^* \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix} \right). \qquad (8)$$

Inserting the condition on nestedness of the basis matrices, we get

$$\begin{aligned}
\mathbf{S}_\tau &= \mathbf{V}_\tau^* \begin{bmatrix} \hat{\mathbf{V}}_\alpha^* & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{V}}_\beta^* \end{bmatrix} \begin{bmatrix} \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix} \left(\mathbf{I} - \begin{bmatrix} \mathbf{0} & \hat{\mathbf{U}}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \hat{\mathbf{U}}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{0} \end{bmatrix} \mathbf{Z}_\tau \begin{bmatrix} \hat{\mathbf{V}}_\alpha^* \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{V}}_\beta^* \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix} \right) \begin{bmatrix} \hat{\mathbf{U}}_\alpha & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{U}}_\beta \end{bmatrix} \mathbf{U}_\tau \\
&= \mathbf{V}_\tau^* \left(\mathbf{I} - \begin{bmatrix} \mathbf{0} & \mathbf{S}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \mathbf{S}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{0} \end{bmatrix} \mathbf{Z}_\tau \right) \begin{bmatrix} \mathbf{S}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_\beta \end{bmatrix} \mathbf{U}_\tau
\end{aligned}$$

Compression of parent equil. equation for a node τ with children $\{\alpha, \beta\}$ (so $I_\tau = I_\alpha \cup I_\beta$)

The equilibrium equation for node τ can be written

$$\begin{bmatrix} \mathbf{A}_{\alpha,\alpha} & \mathbf{A}_{\alpha,\beta} \\ \mathbf{A}_{\beta,\alpha} & \mathbf{A}_{\beta,\beta} \end{bmatrix} \mathbf{q}_\tau + \underbrace{\mathbf{A}(I_\tau, I_\tau^{\mathrm{c}})\mathbf{q}(I_\tau^{\mathrm{c}})}_{=\mathbf{g}_\tau} = \mathbf{f}_\tau.$$

Left multiply by $\begin{bmatrix} \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix}$ to obtain

$$\begin{bmatrix} \mathbf{I} & \mathbf{A}_{\alpha,\alpha}^{-1}\mathbf{A}_{\alpha,\beta} \\ \mathbf{A}_{\beta,\beta}^{-1}\mathbf{A}_{\beta,\alpha} & \mathbf{I} \end{bmatrix} \mathbf{q}_\tau + \begin{bmatrix} \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix} \mathbf{g}_\tau = \begin{bmatrix} \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix} \mathbf{f}_\tau,$$

and then

$$\mathbf{q}_\tau + \begin{bmatrix} \mathbf{I} & \mathbf{A}_{\alpha,\alpha}^{-1}\mathbf{A}_{\alpha,\beta} \\ \mathbf{A}_{\beta,\beta}^{-1}\mathbf{A}_{\beta,\alpha} & \mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix} \mathbf{g}_\tau = \begin{bmatrix} \mathbf{I} & \mathbf{A}_{\alpha,\alpha}^{-1}\mathbf{A}_{\alpha,\beta} \\ \mathbf{A}_{\beta,\beta}^{-1}\mathbf{A}_{\beta,\alpha} & \mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{A}_{\alpha,\alpha}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\beta,\beta}^{-1} \end{bmatrix} \mathbf{f}_\tau,$$

One can show that the compressed version of this equation takes the form

$$\tilde{\mathbf{q}}_\tau + \underbrace{\mathbf{V}_\tau^* \begin{bmatrix} \mathbf{I} & \mathbf{S}_\alpha\tilde{\mathbf{A}}_{\alpha,\beta} \\ \mathbf{S}_\beta\tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{S}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_\beta \end{bmatrix} \mathbf{U}_\tau}_{=\mathbf{S}_\tau} \tilde{\mathbf{g}}_\tau = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{S}_\alpha\tilde{\mathbf{A}}_{\alpha,\beta} \\ \mathbf{S}_\beta\tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\mathbf{r}}_\alpha \\ \tilde{\mathbf{r}}_\beta \end{bmatrix}}_{=\tilde{\mathbf{r}}_\tau}$$

which simplifies to

$$\tilde{\mathbf{q}}_\tau + \mathbf{S}_\tau\tilde{\mathbf{g}}_\tau = \tilde{\mathbf{r}}_\tau.$$

Top level solve (root processing)

Recall that for any node τ, the (compressed) local equilibrium equation reads

$$\tilde{\mathbf{q}}_\tau + \mathbf{S}_\tau \tilde{\mathbf{g}}_\tau = \tilde{\mathbf{r}}_\tau.$$

All $\tilde{\mathbf{r}}_\tau$ can be computed from $\mathbf{f}$ via a simple upwards pass.

Nice: Once you reach the root node ($\tau = 1$), there is no incoming field, $\tilde{\mathbf{g}}_1 = \mathbf{0}$.
So for the root node, we simply get

$$\begin{bmatrix} \tilde{\mathbf{q}}_2 \\ \tilde{\mathbf{q}}_3 \end{bmatrix} = \tilde{\mathbf{q}}_1 = \tilde{\mathbf{r}}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{S}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \mathbf{S}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\mathbf{r}}_2 \\ \tilde{\mathbf{r}}_3 \end{bmatrix}.$$

Then the incoming expansions for the children $\alpha = 2$ and $\beta = 3$ are

$$\begin{bmatrix} \tilde{\mathbf{g}}_2 \\ \tilde{\mathbf{g}}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{2,3} \\ \tilde{\mathbf{A}}_{2,3} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{q}}_2 \\ \tilde{\mathbf{q}}_3 \end{bmatrix}.$$

Algorithm I: Build stage

Upwards pass:

for $\tau = N_{\text{boxes}} : (-1) : 2$

if τ is a leaf

$\mathbf{H}_\tau = \mathbf{A}_{\tau,\tau}^{-1}$

$\mathbf{S}_\tau = \mathbf{V}_\tau^* \mathbf{H}_\tau \mathbf{U}_\tau$

else

Let α and β denote the children of τ.

$$\mathbf{Z}_\tau = \begin{bmatrix} \mathbf{I} & \mathbf{S}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \mathbf{S}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{I} \end{bmatrix}^{-1}.$$

$$\mathbf{S}_\tau = \mathbf{V}_\tau^* \mathbf{Z}_\tau \begin{bmatrix} \mathbf{S}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_\beta \end{bmatrix} \mathbf{U}_\tau$$

end if

end for

Top-level solve:

$$\mathbf{Z}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{S}_2 \tilde{\mathbf{A}}_{2,3} \\ \mathbf{S}_3 \tilde{\mathbf{A}}_{3,2} & \mathbf{I} \end{bmatrix}^{-1}$$

Algorithm II: Solve stage

Upwards pass — build all "outgoing (charge) representations" $\tilde{\mathbf{r}}_\tau$*:*

for $\tau = N_{\text{boxes}} : (-1) : 2$

  if τ is a leaf

    $\tilde{\mathbf{r}}_\tau = \mathbf{V}_\tau^* \mathbf{H}_\tau \mathbf{f}(I_\tau)$

  else

    $\tilde{\mathbf{r}}_\tau = \mathbf{V}_\tau^* \mathbf{Z}_\tau \begin{bmatrix} \tilde{\mathbf{r}}_\alpha \\ \tilde{\mathbf{r}}_\beta \end{bmatrix}$, where $\{\alpha, \beta\}$ are the children of τ.

  end if

end for

Top-level solve — build "incoming fields" $\{\tilde{\mathbf{g}}_2, \tilde{\mathbf{g}}_3\}$ *at the top:*

$$\begin{bmatrix} \tilde{\mathbf{g}}_2 \\ \tilde{\mathbf{g}}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{2,3} \\ \tilde{\mathbf{A}}_{3,2} & \mathbf{0} \end{bmatrix} \mathbf{Z}_1 \begin{bmatrix} \tilde{\mathbf{r}}_2 \\ \tilde{\mathbf{r}}_3 \end{bmatrix}.$$

Downwards pass — build all "incoming fields" $\tilde{\mathbf{g}}_\tau$*:*

for $\tau = 2 : N_{\text{boxes}}$

  if τ is a parent

$$\begin{bmatrix} \tilde{\mathbf{g}}_\alpha \\ \tilde{\mathbf{g}}_\beta \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{\alpha,\beta} \\ \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{0} \end{bmatrix} \mathbf{Z}_\tau \begin{bmatrix} \tilde{\mathbf{r}}_\alpha \\ \tilde{\mathbf{r}}_\beta \end{bmatrix} + \left(\mathbf{I} - \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{\alpha,\beta} \\ \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{0} \end{bmatrix} \mathbf{Z}_\tau \begin{bmatrix} \mathbf{S}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_\beta \end{bmatrix} \right) \mathbf{U}_\tau \tilde{\mathbf{g}}_\tau.$$

  else

    $\mathbf{q}(I_\tau) = \mathbf{H}_\tau \left(\mathbf{f}(I_\tau) - \mathbf{U}_\tau \tilde{\mathbf{g}}_\tau \right)$.

  end if

end for

Comments on the merge process:

The expensive part of the computation is the evaluation of the *merge operations*:

$$\mathbf{S}_\tau \quad = \quad \mathbf{V}_\tau^* \quad \begin{bmatrix} \mathbf{I} & \mathbf{S}_\alpha \tilde{\mathbf{A}}_{\alpha,\beta} \\ \mathbf{S}_\beta \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{I} \end{bmatrix}^{-1} \quad \begin{bmatrix} \mathbf{S}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_\beta \end{bmatrix} \quad \mathbf{U}_\tau.$$

$$k_\tau \times k_\tau \qquad k_\tau \times (k_\alpha + k_\beta) \quad (k_\alpha + k_\beta) \times (k_\alpha + k_\beta) \quad (k_\alpha + k_\beta) \times (k_\alpha + k_\beta) \quad (k_\alpha + k_\beta) \times k_\tau$$

In many contexts (BIEs on contours in 2D), k_α and k_β are small enough that the formula can be evaluated by brute force.

Observe that minimal communication is required:

Upwards pass: To process a parent, only information from its children is used.

Downwards pass: To process a child, only information from its parent and sibling is used.

There are no lengthy interaction lists.

Complexity: The direct solver has the following asymptotic complexity:

	Build stage	Solve stage
Integral equations on the line:	$O(N)$	$O(N)$
Integral equations on "simple" curves in $\mathbb{R}^2$:	$O(N)$	$O(N)$
Integral equations on "space filling" curves in $\mathbb{R}^2$:	$O(N^{3/2})$	$O(N \log N)$
Lippman-Schwinger integral equation in $\mathbb{R}^2$:	$O(N^{3/2})$	$O(N \log N)$
Integral equations on "simple" surfaces in $\mathbb{R}^3$:	$O(N^{3/2})$	$O(N \log N)$
Integral equations on "space filling" surface in $\mathbb{R}^3$:	$O(N^2)$	$O(N^{4/3})$
Lippman-Schwinger integral equation in $\mathbb{R}^3$:	$O(N^2)$	$O(N^{4/3})$

For all schemes with super-linear complexity, the cost is dominated by processing large dense matrices at top of the tree.

These matrices typically have internal structure that allows us to reduce the complexity, but things get quite messy. This is work in progress.

For problems with oscillatory kernels (associated with Helmholtz equation), we have linear complexity algorithms based on low-rank considerations only for the case of "elongated" domains. Very recent work by Eric Michielssen indicates that so called "butterfly" algorithms can be very useful in this environment.

Summary

- For BIEs in 2D, existing direct solvers work extremely well — $O(N)$ with small constants.

- For BIEs in 3D, existing direct solvers work well at low accuracies on current computers. For high accuracies, it is in my opinion likely that they will be very effective in 5-10 years when computers have more cores and more memory.

- Progress on developing $O(N)$ direct solvers for problems in 3D is currently rapid. For integral equations, current work by A. Barnett, J. Bremer, E. Corona, A. Gillman, L. Greengard, J. Helsing, K. Ho, L. Ying, D. Zorin, etc.

- Direct solvers for BIEs are extremely well suited to modern computing platforms! They are expensive in that they require a lot of flops and a lot of memory, but they parallelize very well, can exploit (huge!) pre-computed look-up tables of quadratures, etc.