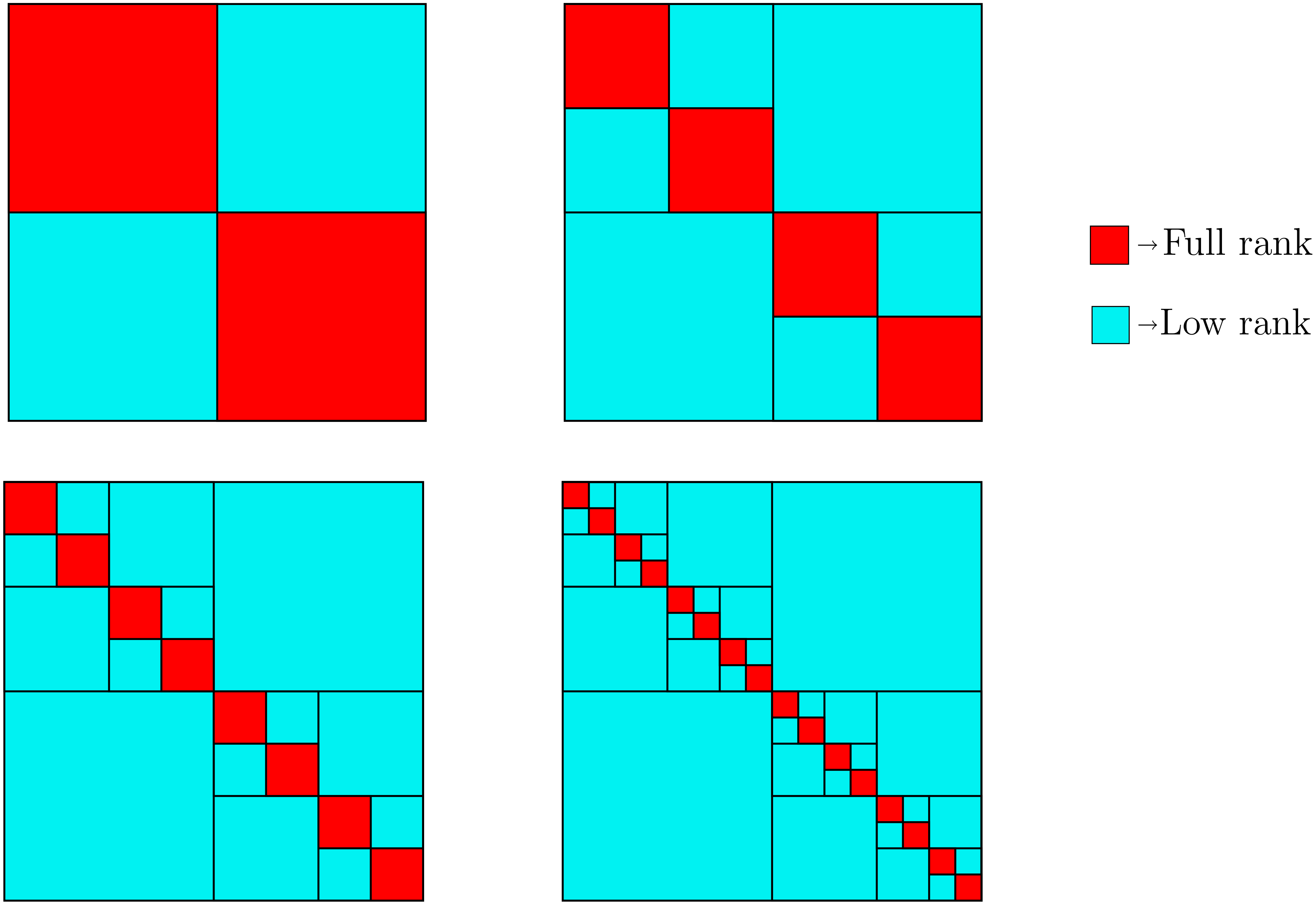
FLATIRON INSTITUTE
Center for Computational Mathematics

Fast algorithms for hierarchically compressible matrices

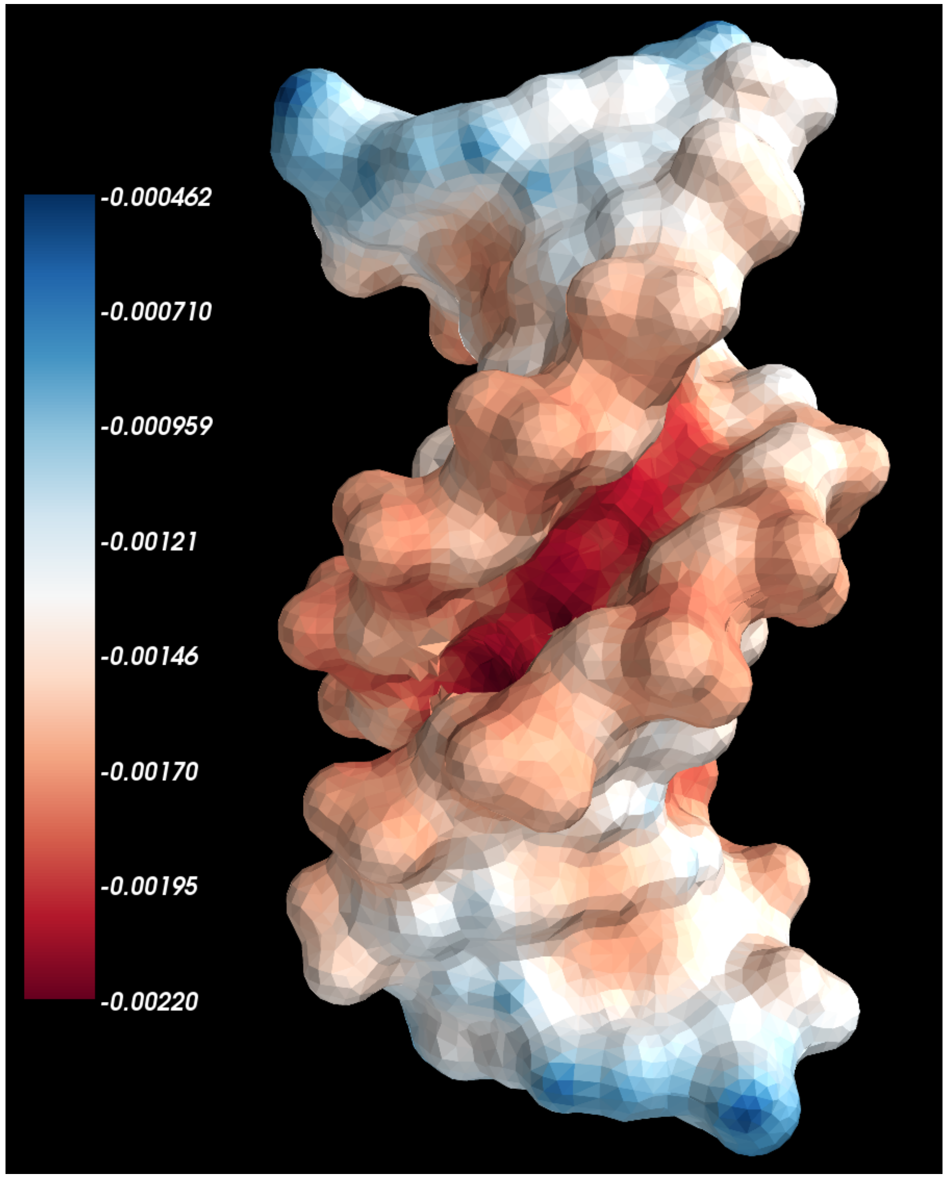
FWAM

Nov 1, 2019

What is a hierarchical matrix?



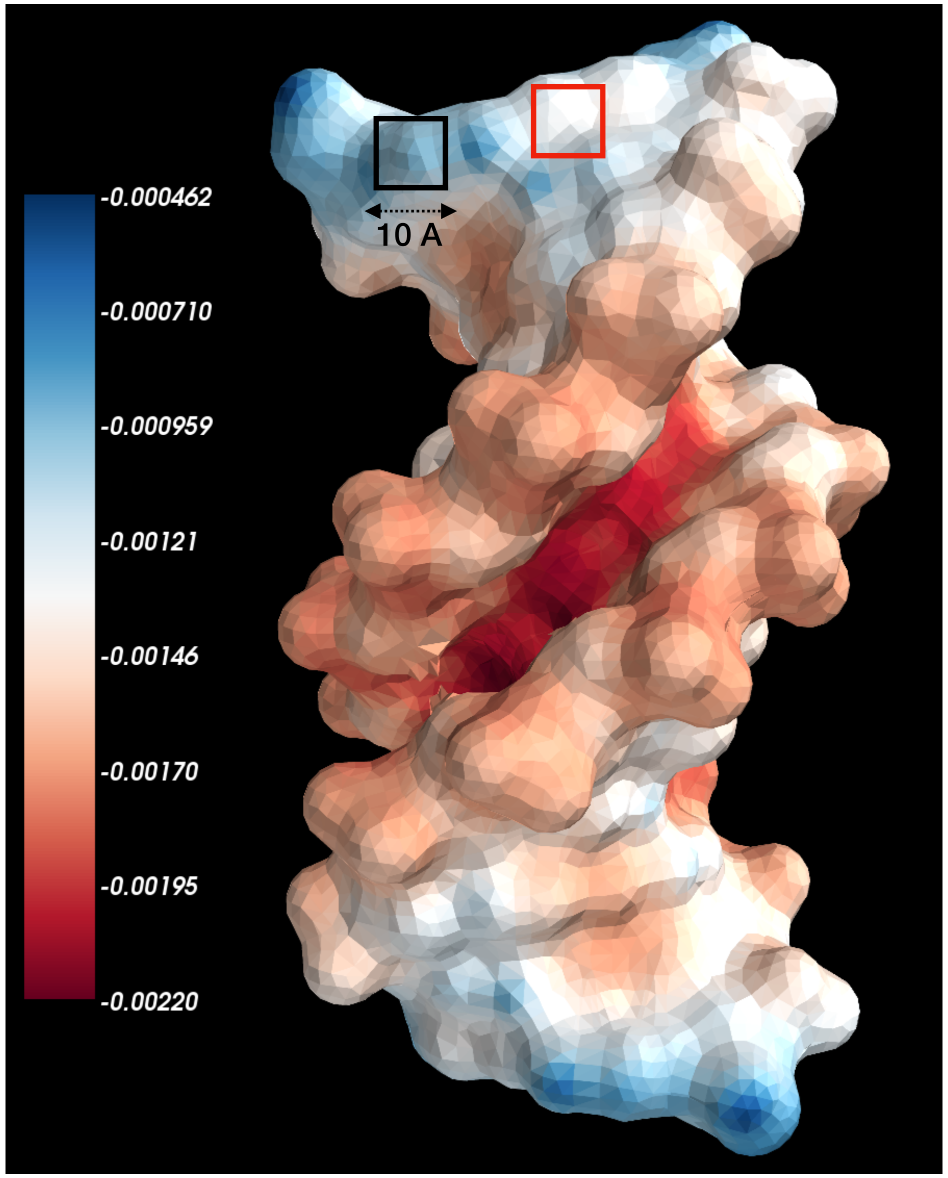
Applications - Molecular dynamics



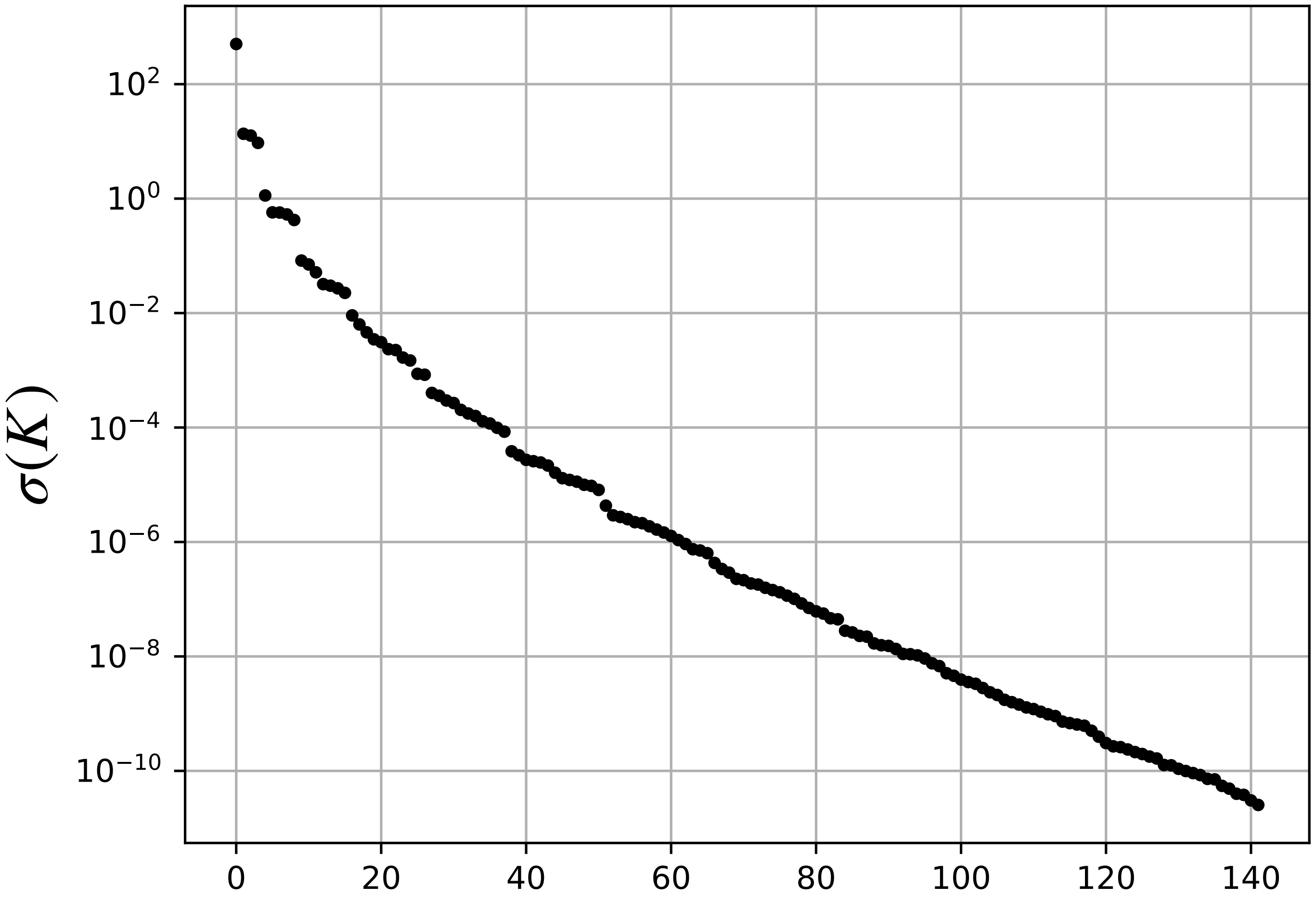
- pKa computation
- Docking

$$
\begin{aligned}
-\Delta\varphi &= 0 && \text{in } \Omega_0 \\
-\Delta\varphi &= \frac{1}{\varepsilon_1}\sum_i q_i \delta\left(\mathbf{r} - \mathbf{r}_i\right) && \text{in } \Omega_1 \\
[\varphi] = \left[\varepsilon \frac{\partial\varphi}{\partial\nu}\right] &= 0 && \text{on } \Sigma
\end{aligned}
$$

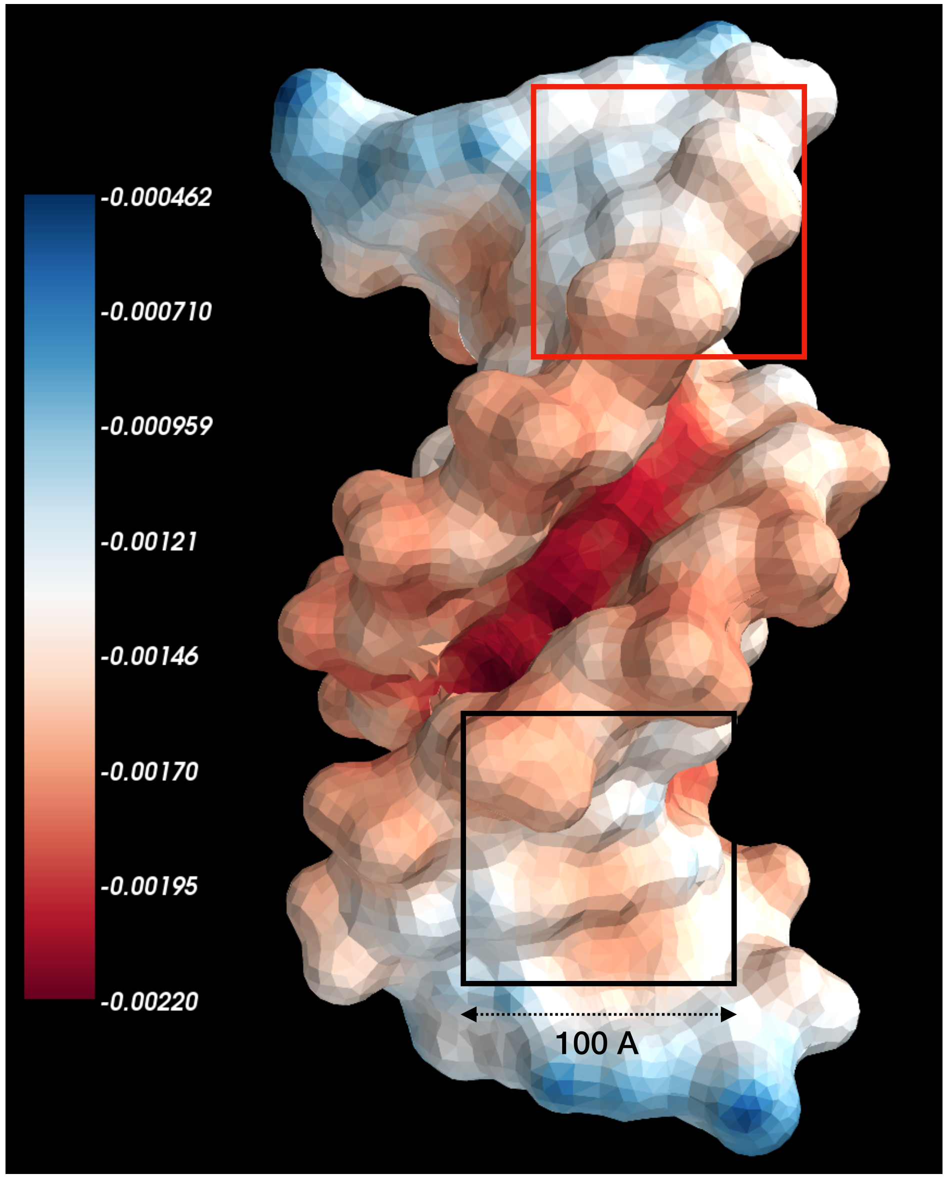
Applications - Molecular dynamics



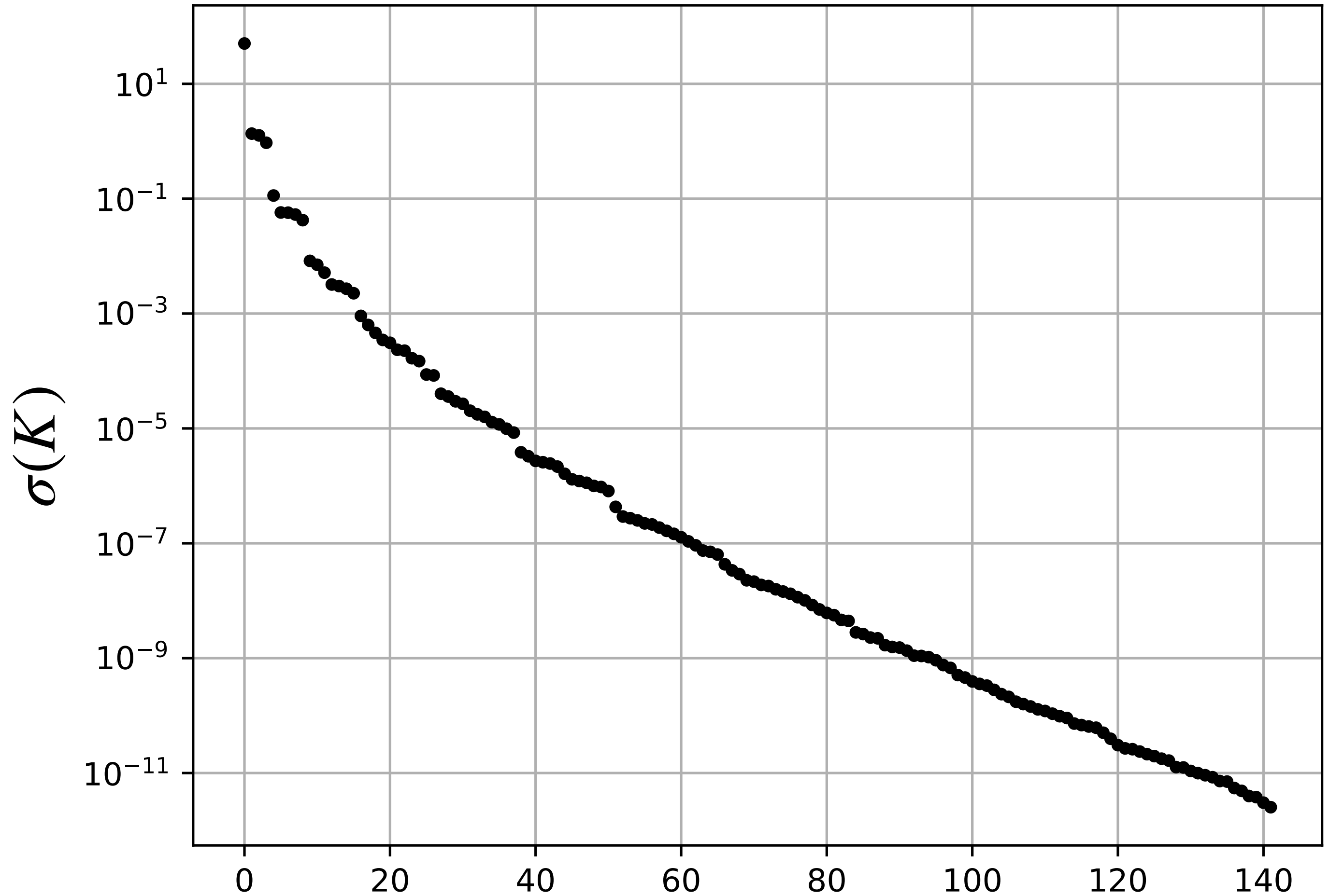
$$K_{i,j} = \frac{1}{|x_i - x_j|}$$



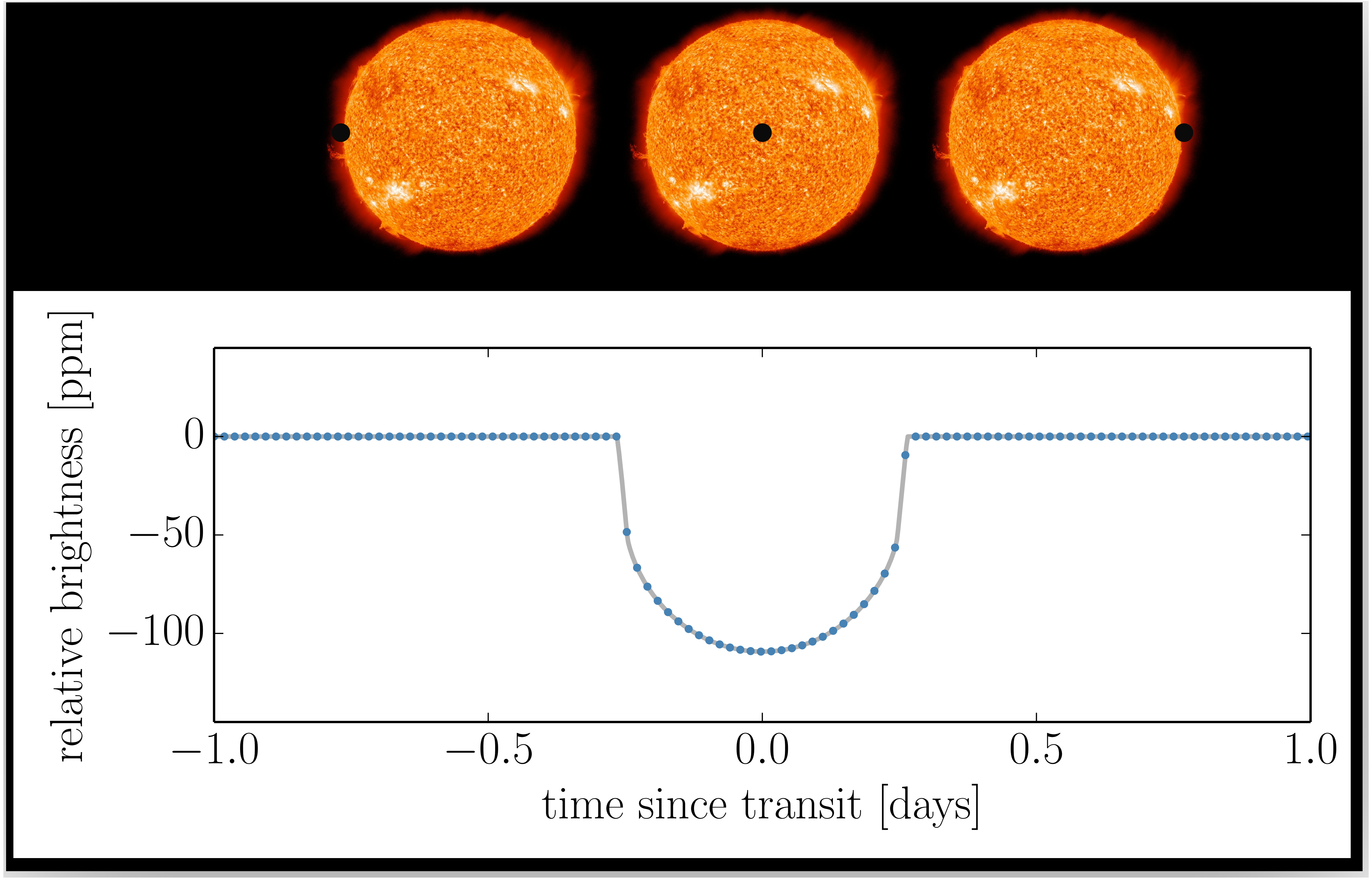
Applications - Molecular dynamics



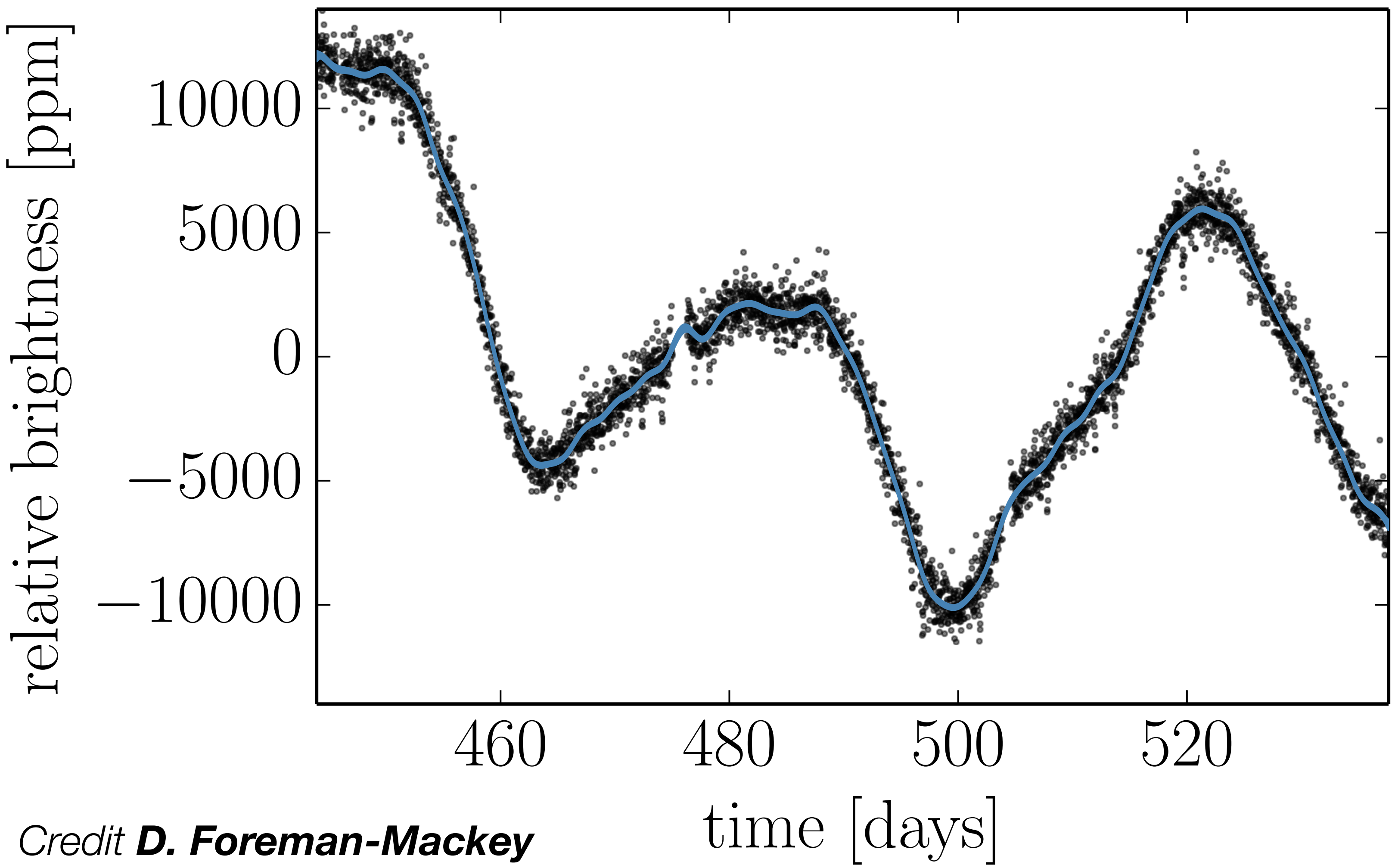
$$K_{i,j} = \frac{1}{|x_i - x_j|}$$



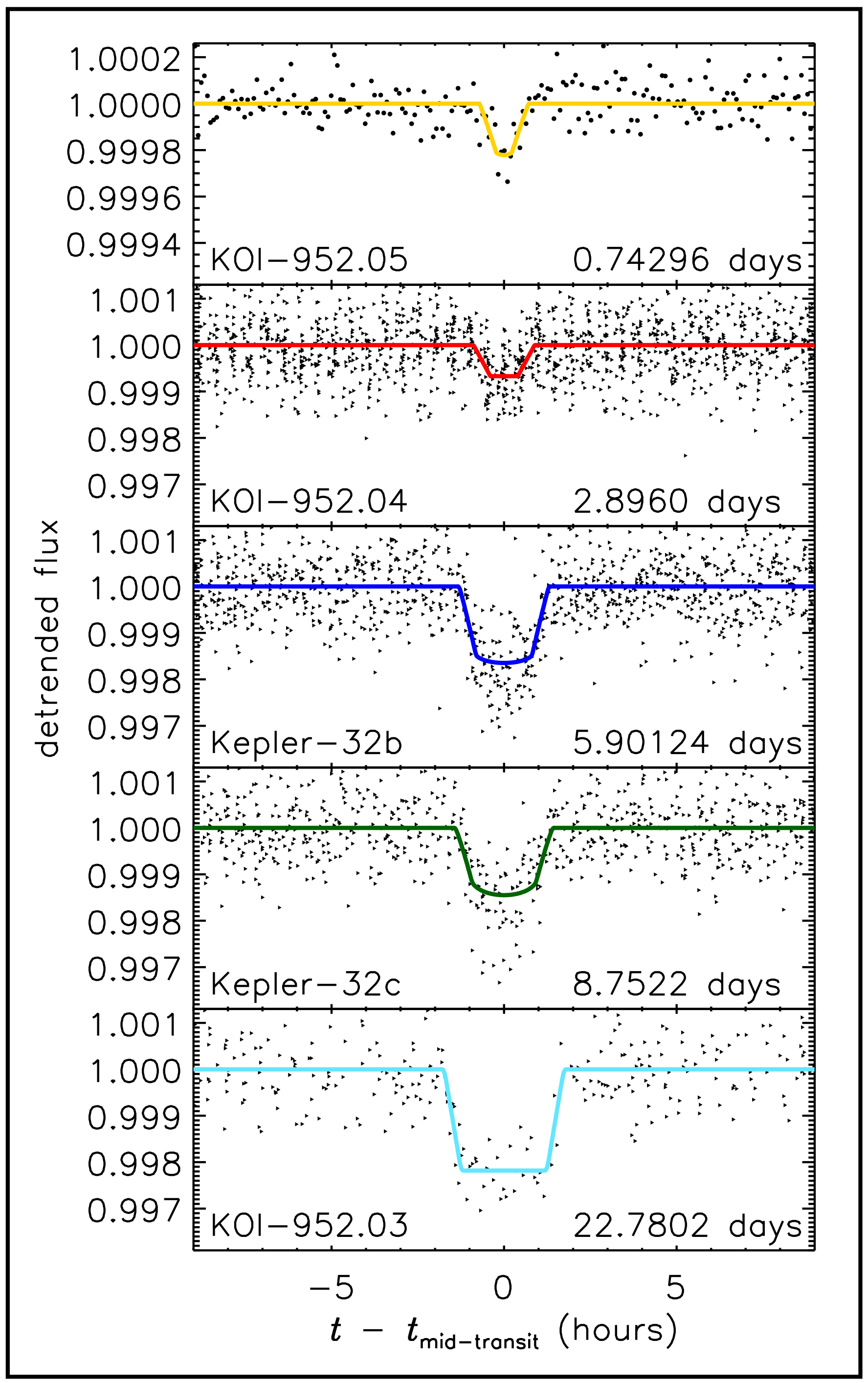
Exoplanet hunting using Gaussian Processes



Exoplanet hunting using Gaussian Processes



Credit ***D. Foreman-Mackey***



Credit **Fabrycky *et al.* (2012)**

Computational Task:

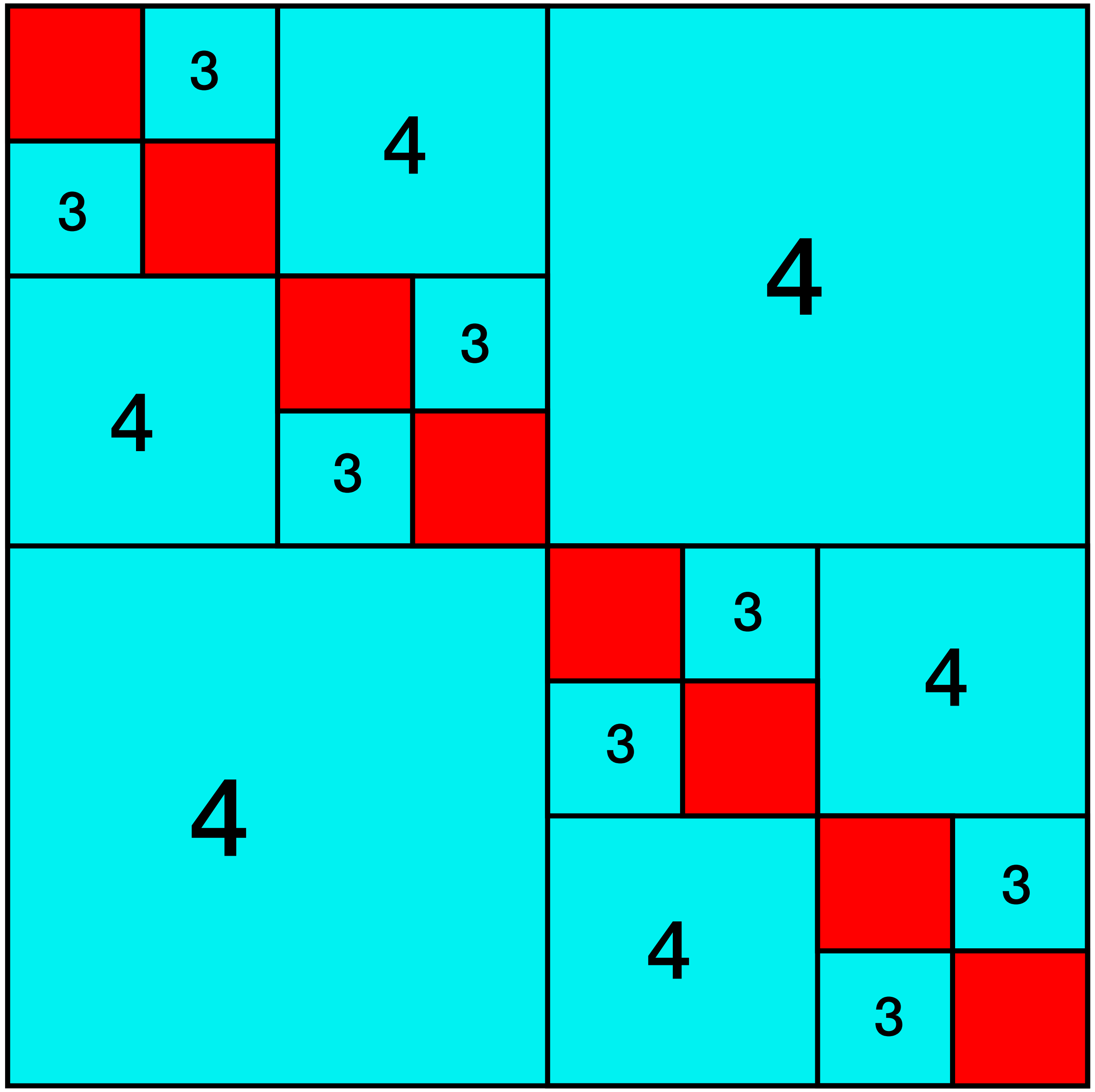
$$\text{argmax}_{\boldsymbol{\theta}} \mathscr{L}_{\boldsymbol{\theta}} \propto \frac{1}{\det C(\boldsymbol{t};\boldsymbol{\theta})^{1/2}} e^{-\frac{1}{2}\boldsymbol{y}^T C^{-1}(\boldsymbol{t};\boldsymbol{\theta})\boldsymbol{y}}$$

Exoplanet hunting using Gaussian Processes

Computational Task:

argmax$_{\boldsymbol{\theta}} \mathscr{L}_{\boldsymbol{\theta}} \propto \frac{1}{\det C(\boldsymbol{t};\boldsymbol{\theta})^{1/2}} e^{-\frac{1}{2}\boldsymbol{y}^T C^{-1}(\boldsymbol{t};\boldsymbol{\theta})\boldsymbol{y}}$

$$C(\boldsymbol{t},\boldsymbol{\theta}) = \sigma_\varepsilon^2 I + K(t,t';\boldsymbol{\theta})$$



$$K(t,t',\boldsymbol{\theta}) = \theta(0) + e^{-\frac{(t-t')^2}{2\theta(1)^2}}$$

$$\theta(0) = 1, \theta(1) = 3$$

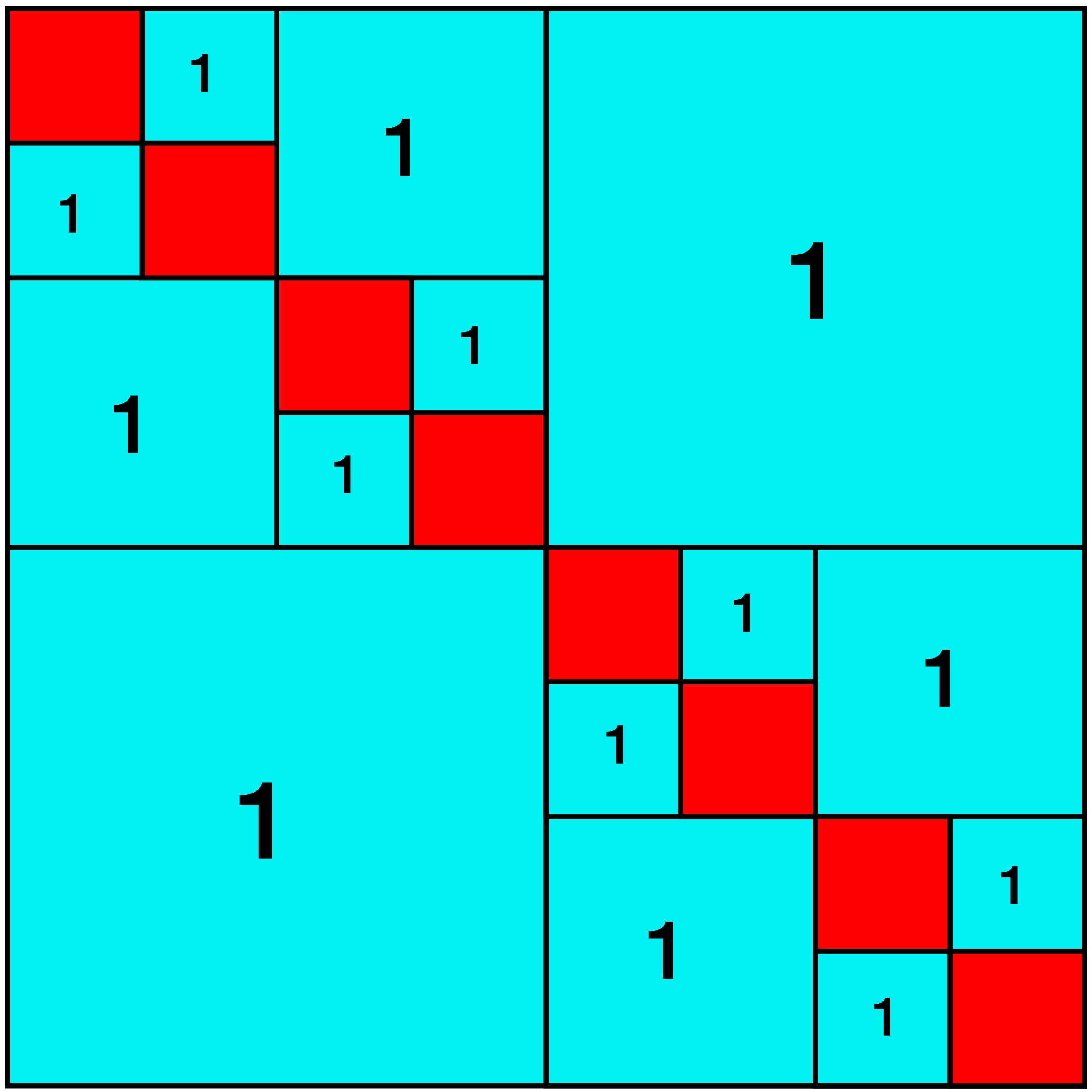
10k $t_i's$ uniformly distributed on $[0,1]$

Exoplanet hunting using Gaussian Processes

Computational Task:

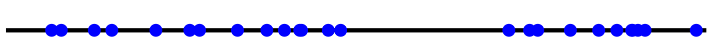
$$\text{argmax}_{\boldsymbol{\theta}} \mathscr{L}_{\boldsymbol{\theta}} \propto \frac{1}{\det C(\boldsymbol{t};\boldsymbol{\theta})^{1/2}} e^{-\frac{1}{2}\boldsymbol{y}^T C^{-1}(\boldsymbol{t};\boldsymbol{\theta})\boldsymbol{y}}$$

$$C(\boldsymbol{t},\boldsymbol{\theta}) = \sigma_\varepsilon^2 I + K(t,t';\boldsymbol{\theta})$$



$$K(t,t',\boldsymbol{\theta}) = \theta(0) + e^{-\theta(1)|t-t'|}$$

$$\theta(0) = 1, \theta(1) = 3$$



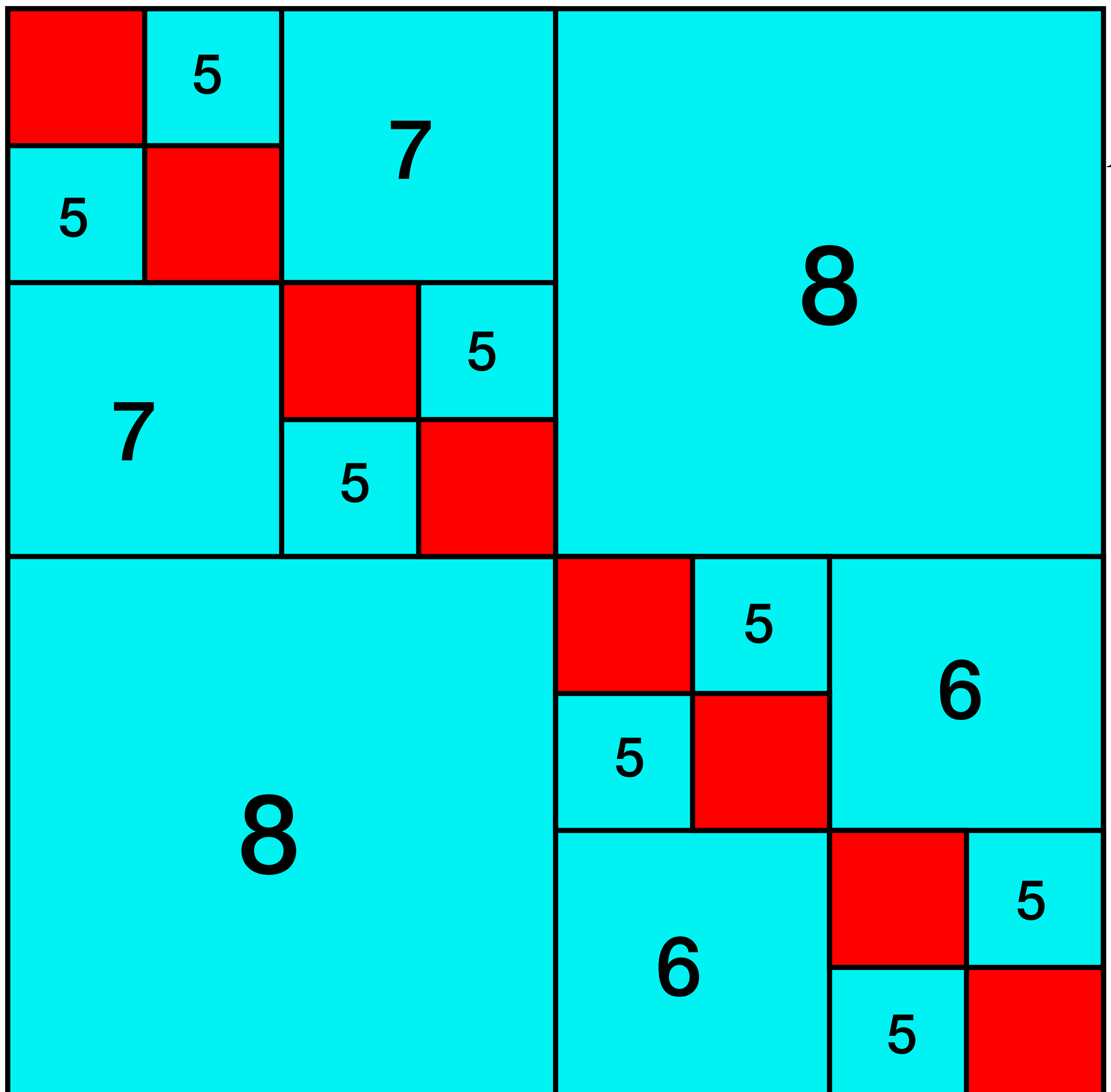
10k $t_i's$ uniformly distributed on $[0,1]$

Exoplanet hunting using Gaussian Processes

Computational Task:

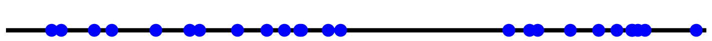
$$\text{argmax}_{\boldsymbol{\theta}} \mathscr{L}_{\boldsymbol{\theta}} \propto \frac{1}{\det C(\boldsymbol{t};\boldsymbol{\theta})^{1/2}} e^{-\frac{1}{2}\boldsymbol{y}^T C^{-1}(\boldsymbol{t};\boldsymbol{\theta})\boldsymbol{y}}$$

$$C(\boldsymbol{t},\boldsymbol{\theta}) = \sigma_\varepsilon^2 I + K(t, t'; \boldsymbol{\theta})$$



$$K(t,t',\boldsymbol{\theta}) = \theta(0) + \frac{1}{1+(x(i)-x(j))^2}$$

$$\theta(0) = 1$$



10k $t_i's$ uniformly distributed on $[0,1]$

What is fast?

Suppose $A \in \mathbb{R}^{n \times n}$, and, $\boldsymbol{v} \in \mathbb{R}^n$

- Matrix vector product (matvec) $A \cdot \boldsymbol{v}$: $O(n^2)$
- Inversion A^{-1} : $O(n^3)$
- Determinants $\det A$: $O(n^3)$

For a given task, an algorithm is fast if it's runtime beats the asymptotic complexity

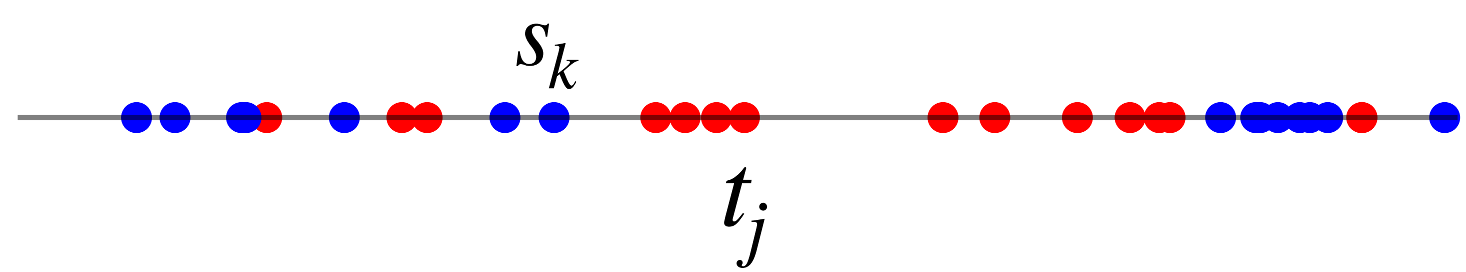
The dream: $O(n \log^s n)$

Examples

- Sparse matrices, matvecs in $O(kn)$, if well-conditioned, inverse in $O(kn)$
- FFT matrices, matvecs in $O(n \log n)$, inverse analytically known, and inverse application in $O(n \log n)$

Dense matrices $\neq$ Data dense

$$A_{j,k} = \delta_{j,k} + \cos(t_j - s_k)$$
$$= \delta_{j,k} + \cos(t_j)\cos(s_k) + \sin(t_j)\sin(s_k)$$



- **Matvec** $\boldsymbol{b} = A \cdot \boldsymbol{v}$: $O(n^2)$

Step 1:

$$W_1 = \sum_{k=1}^{n} \cos(s_k) v_k\,, \quad W_2 = \sum_{k=1}^{n} \sin(s_k) v_k$$

Step 2:

$$b_j = v_j + \cos(t_j) W_1 + \sin(t_j) W_2 \qquad O(n)!$$

$$A = I + UV^T, \quad U = \begin{bmatrix} \cos(t_1) & \sin(t_1) \\ \cos(t_2) & \sin(t_2) \\ \vdots & \vdots \\ \cos(t_n) & \sin(t_n) \end{bmatrix}, \quad V = \begin{bmatrix} \cos(s_1) & \sin(s_1) \\ \cos(s_2) & \sin(s_2) \\ \vdots & \vdots \\ \cos(s_n) & \sin(s_n) \end{bmatrix}$$

- **Inversion** A^{-1} : $O(n^3)$

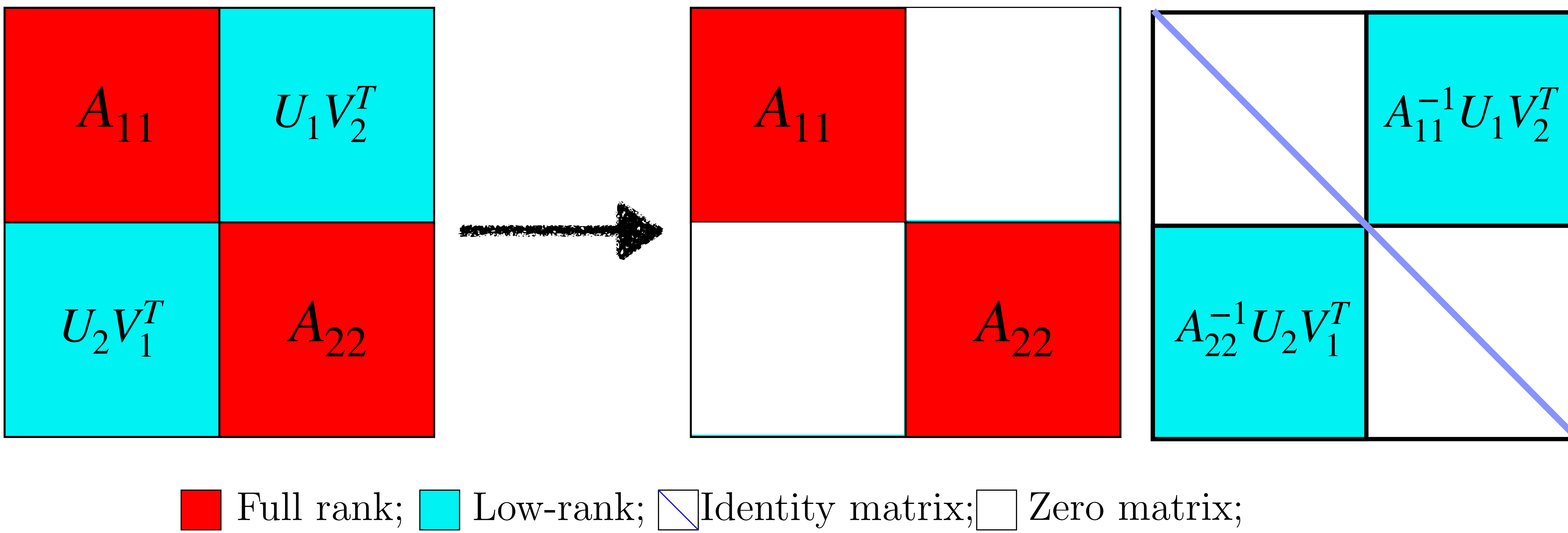
  Sherman Morrison Woodbury formula: $A^{-1} = I - \underset{n\times 2}{U}(\underset{2\times 2}{I_2 + V^T U})^{-1} \underset{2\times n}{V^T}$ $O(n)!$

- **Determinants** $\det A$: $O(n^3)$

  Slyvester formula formula: $\det A = \det(I_2 + V^T U)$ $O(n)!$

One level scheme - factorization

Assume: All off-diagonal blocks are rank r



Factorization tasks and costs:

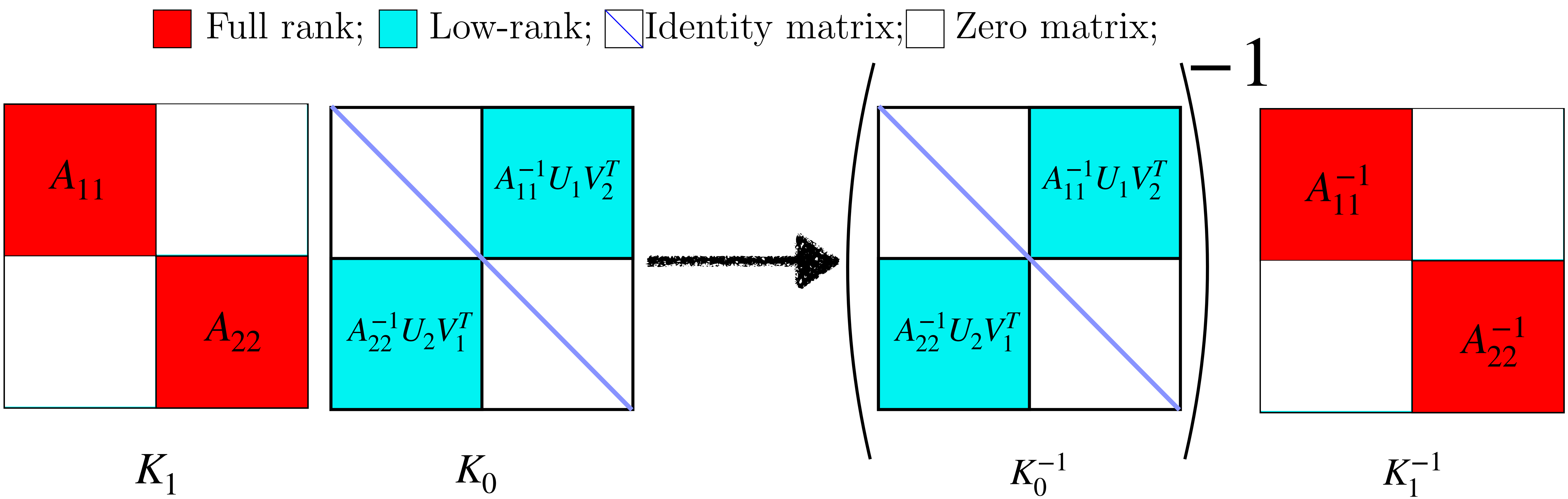
Compute $A_{12} = U_1V_2^T$, and $A_{21} = U_2V_1^T$: $2 \cdot O\left(\frac{n^2}{4} \cdot r\right)$

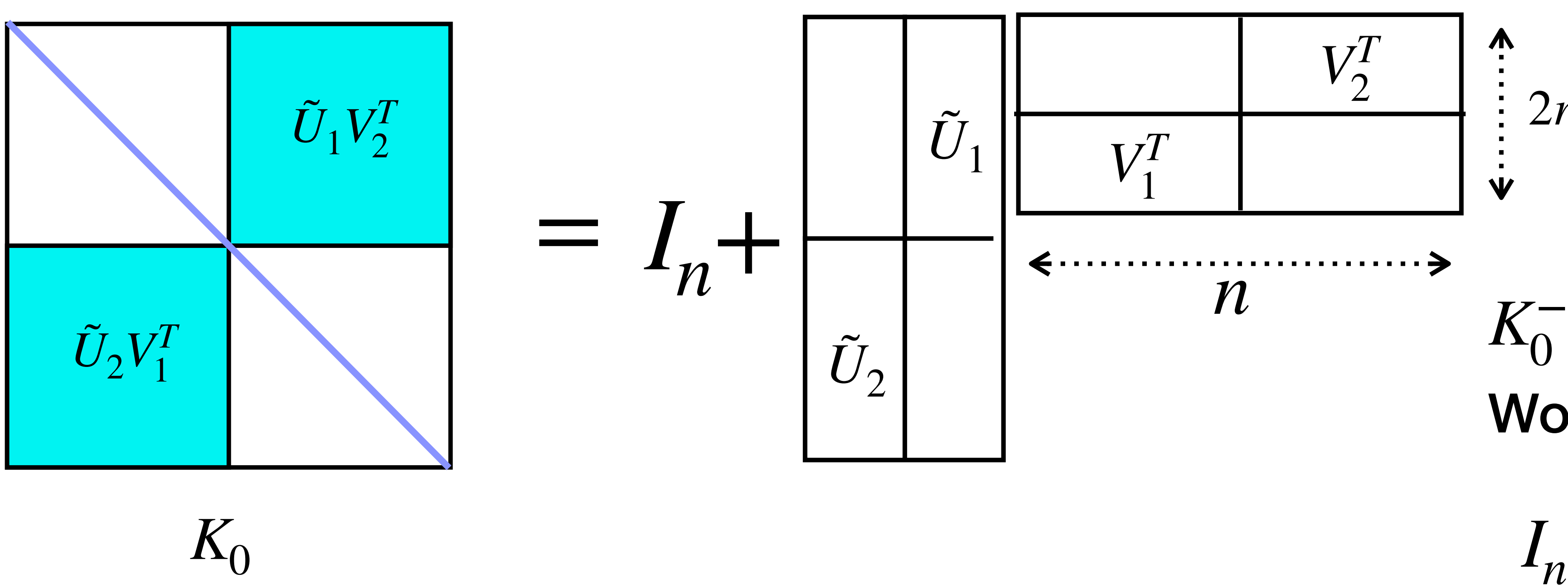
Compute A_{11}^{-1}, and A_{22}^{-1} : $2 \cdot \frac{n^3}{8}$

Factorization Cost:

$\frac{n^3}{4}$

One level scheme - Inversion



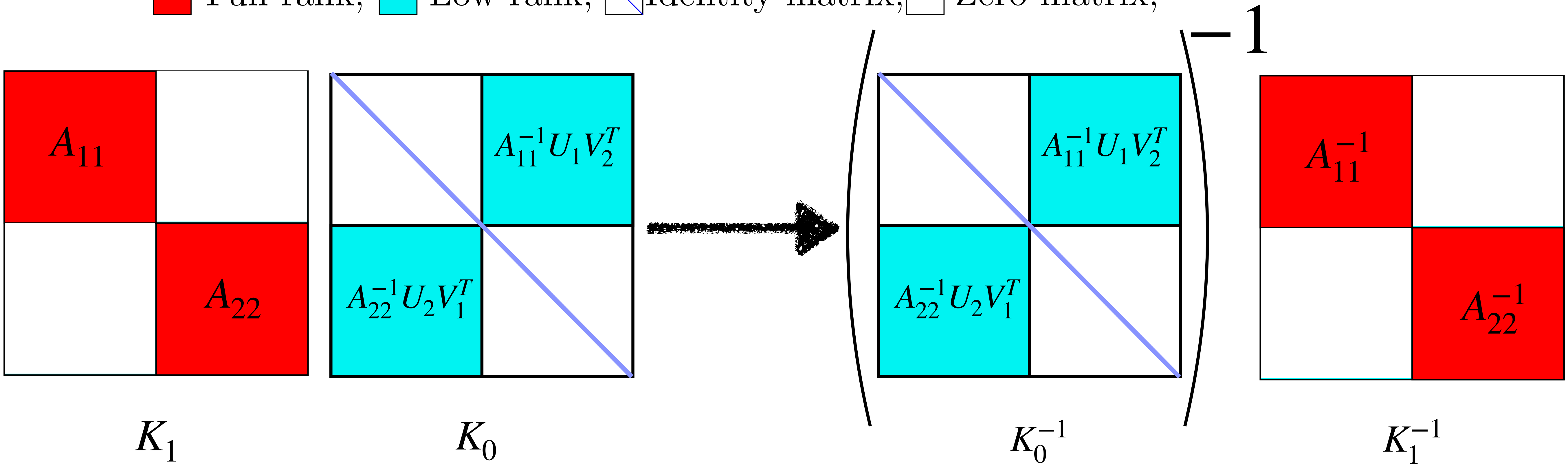


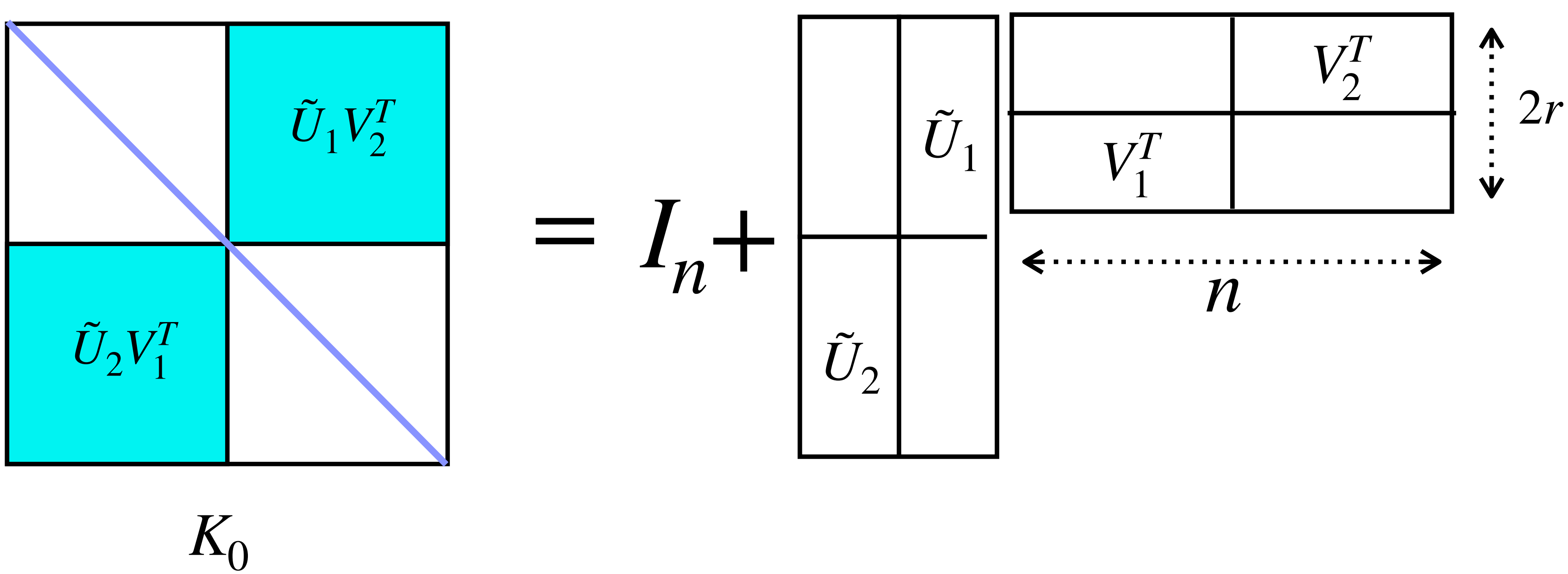
K_0^{-1} using Sherman Morrison Woodbury formula in $O\left(4nr^2\right)$

$$I_n - U\left(I_{2r} + V^TU\right)^{-1}V^T$$

One level scheme - Inversion

Full rank; Low-rank; Identity matrix; Zero matrix;





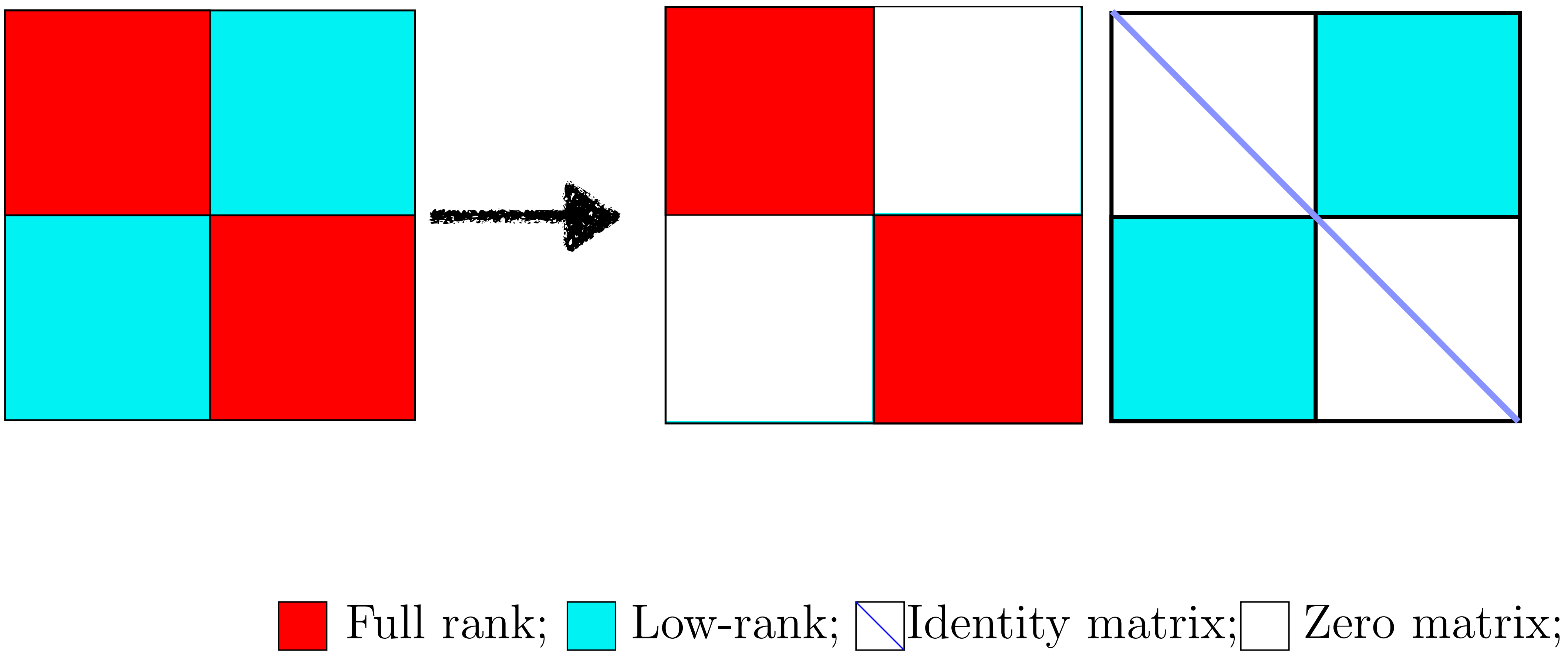
Inversion Cost:

$n^3 \rightarrow \frac{n^2}{2}$

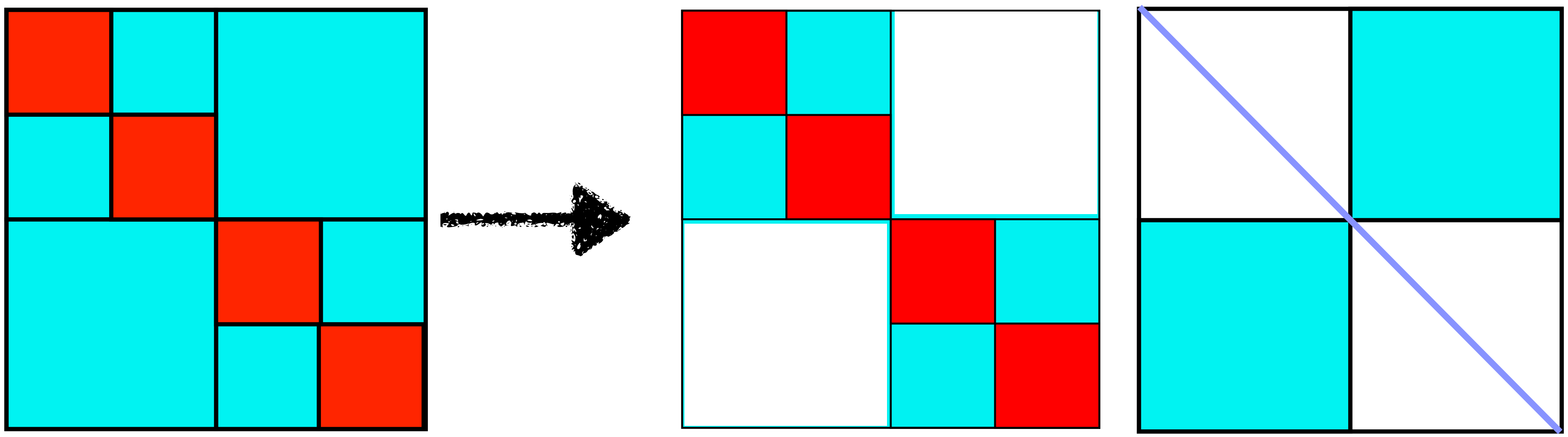
Factorization Cost:

$\frac{n^3}{4}$

Can we induct?

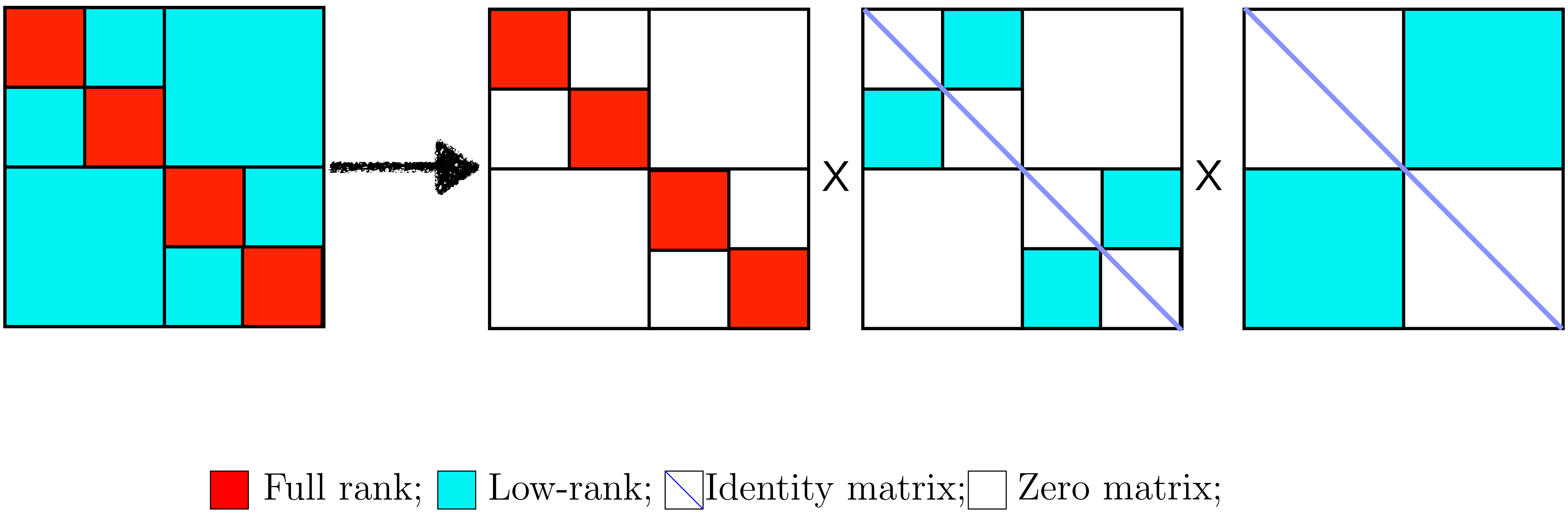


Can we induct?

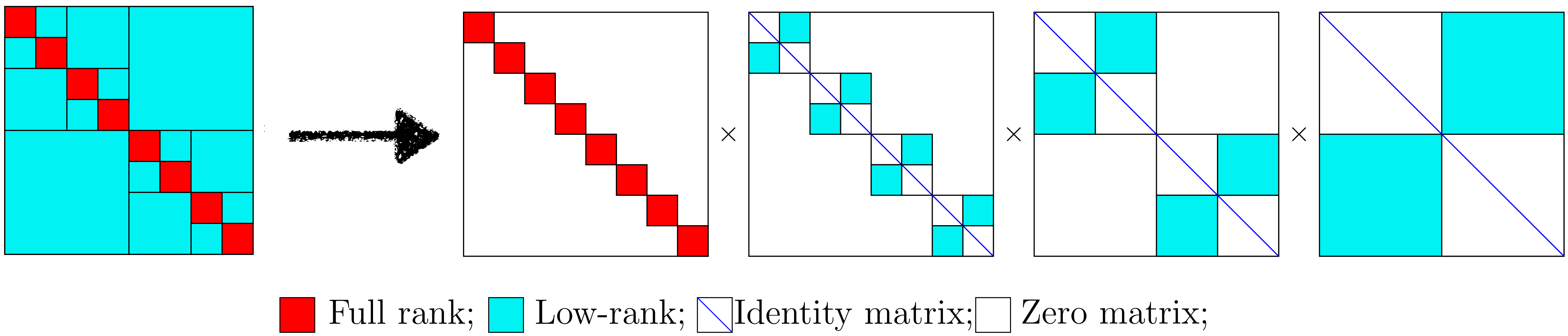


Full rank; Low-rank; Identity matrix; Zero matrix;

Can we induct?



Algorithm and factorization costs



1. Compute all low-rank factorizations of off-diagonal blocks at all levels $O(n^2 r)$

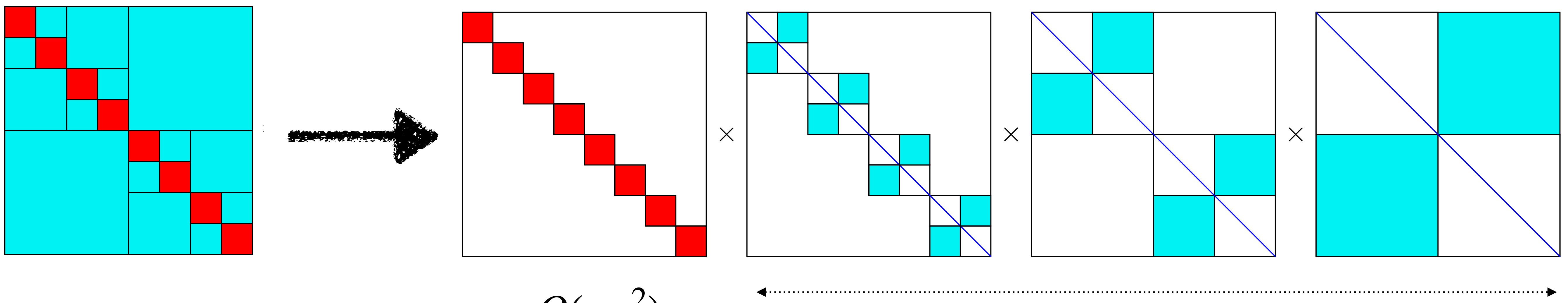
2. Compute inverses of n/p , $p \times p$ matrices at the finest level $O(np^2)$

3. Loop over levels $j = \kappa - 1, \ldots 1$
   a. Update the inverses of the coarser diagonal blocks
   b. Update the off-diagonal low rank factors using the computed inverses

   $O(npr \log n)$

Factorization cost: $O(n^2 r + np^2 + npr \log^2 n)$

Apply/Inversion/Determinant cost



$O(np^2)$

Each matrix is of the form $(I + UV^T)$

where rank of U,V is $2r$

Cost of applying/inverting/computing determinants

at each level: $O(nr^2)$

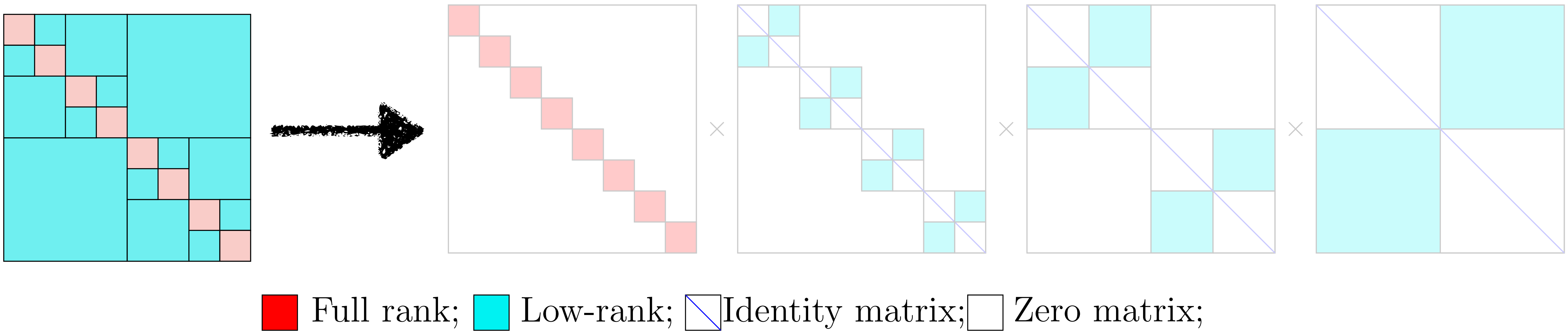
Post factorization

Inversion cost: $O(n)$

Determinant cost: $O(n)$

Full rank; Low-rank; Identity matrix; Zero matrix;

Low rank factorizations of off-diagonal blocks

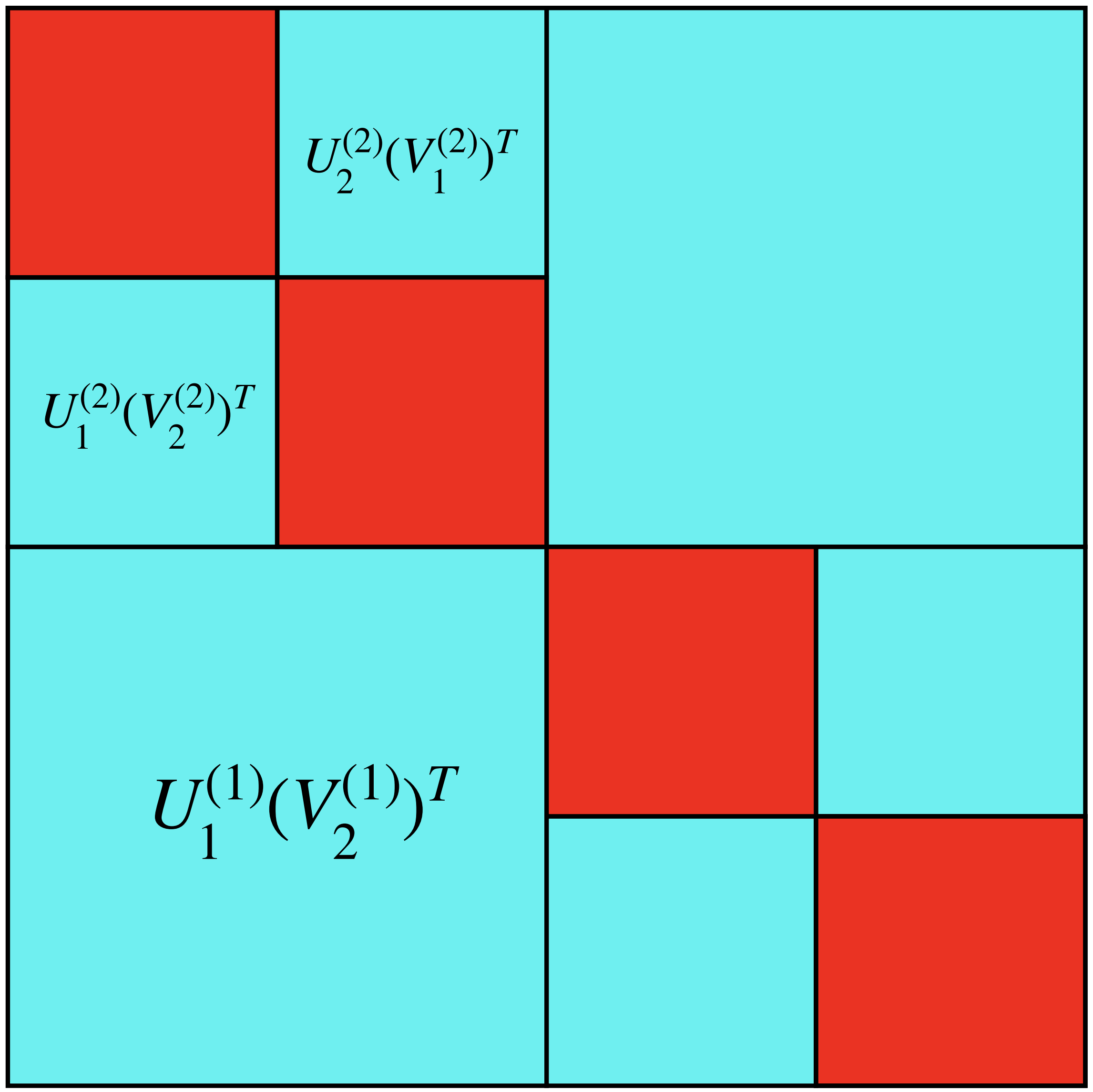


Compute all low-rank factorizations of off-diagonal blocks at all levels $O(n^2 r)$

Options:

1. Analysis — Need different expansions per kernel!
    - Compute analytical low rank decompositions of the kernel
    - $e^{-(s-t)^2/2} = \sum_{n=0}^{r} \frac{(t-c)^n}{n!} h_n(s-c) + O(\varepsilon)$
2. Linear algebra — Can be unstable sometimes!
    - Partial pivoted LU
    - Adaptive Cross Approximation
3. Using nested basis — Can be unstable sometimes - but instabilities known!

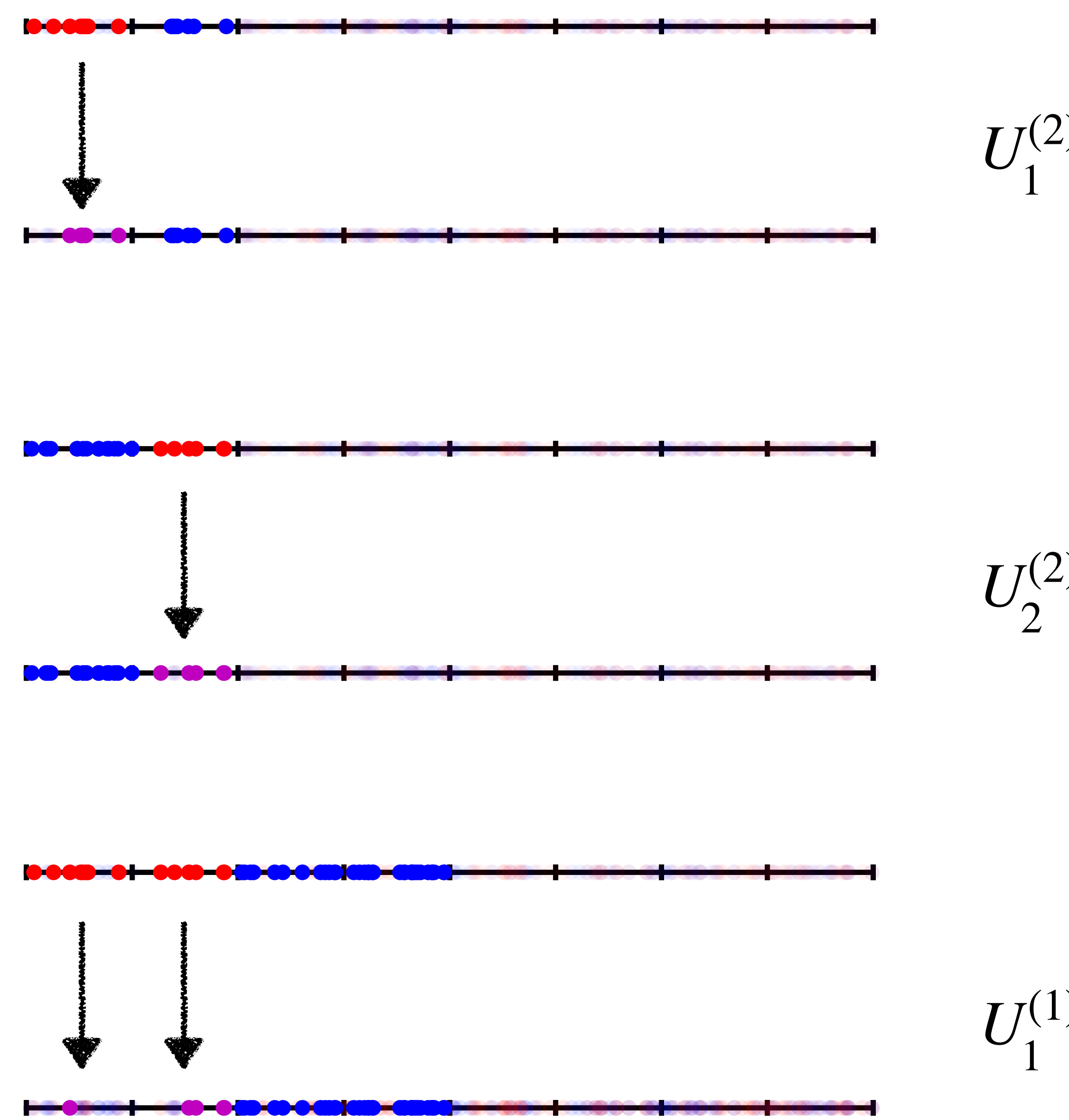
Nested basis



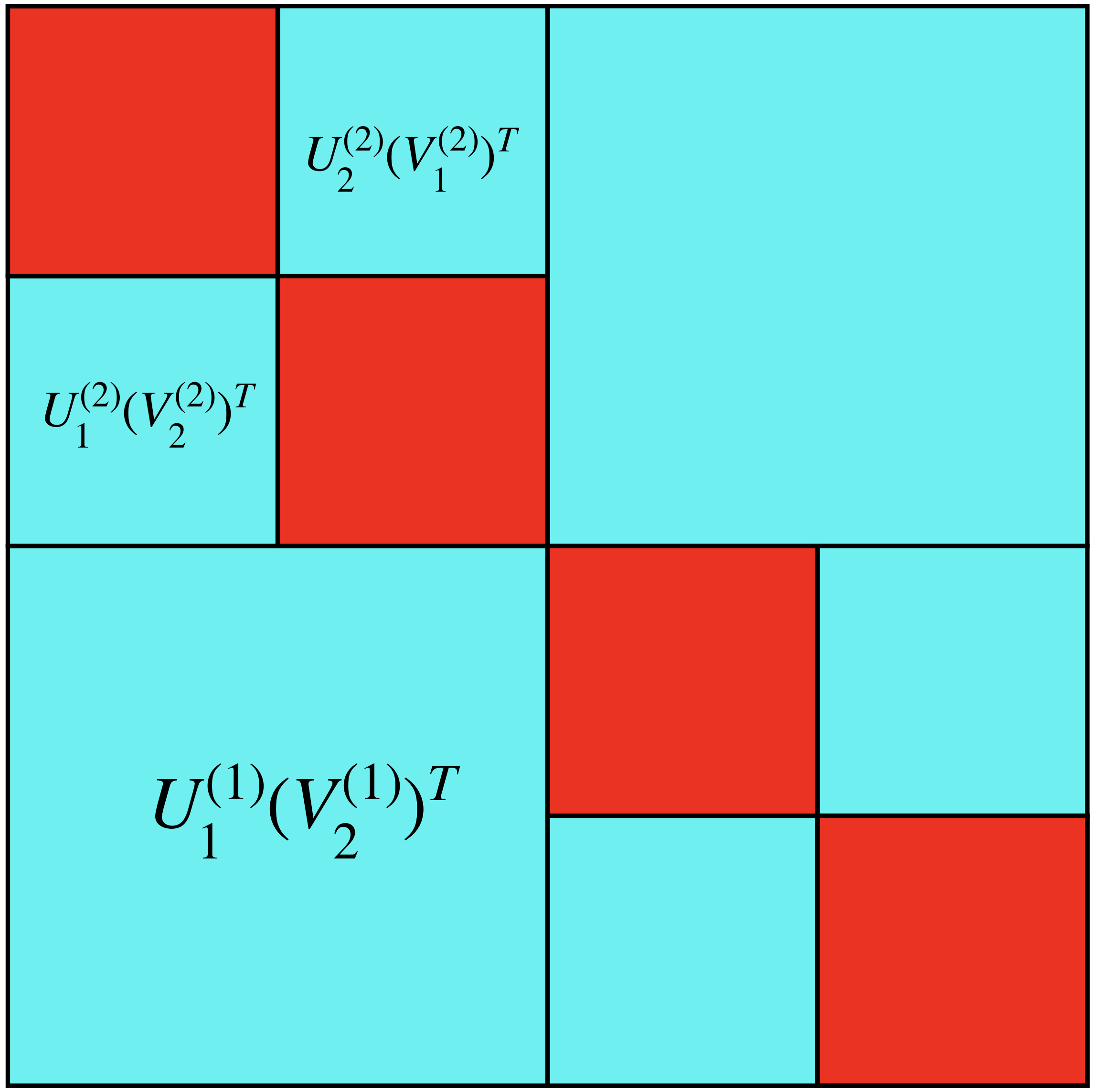
Q: Given $U_1^{(2)}$, $U_2^{(2)}$, can we compute $U_1^{(1)}$?

In general: No!

$U_i^{(j)}$ can be thought of as subset of columns of original matrix



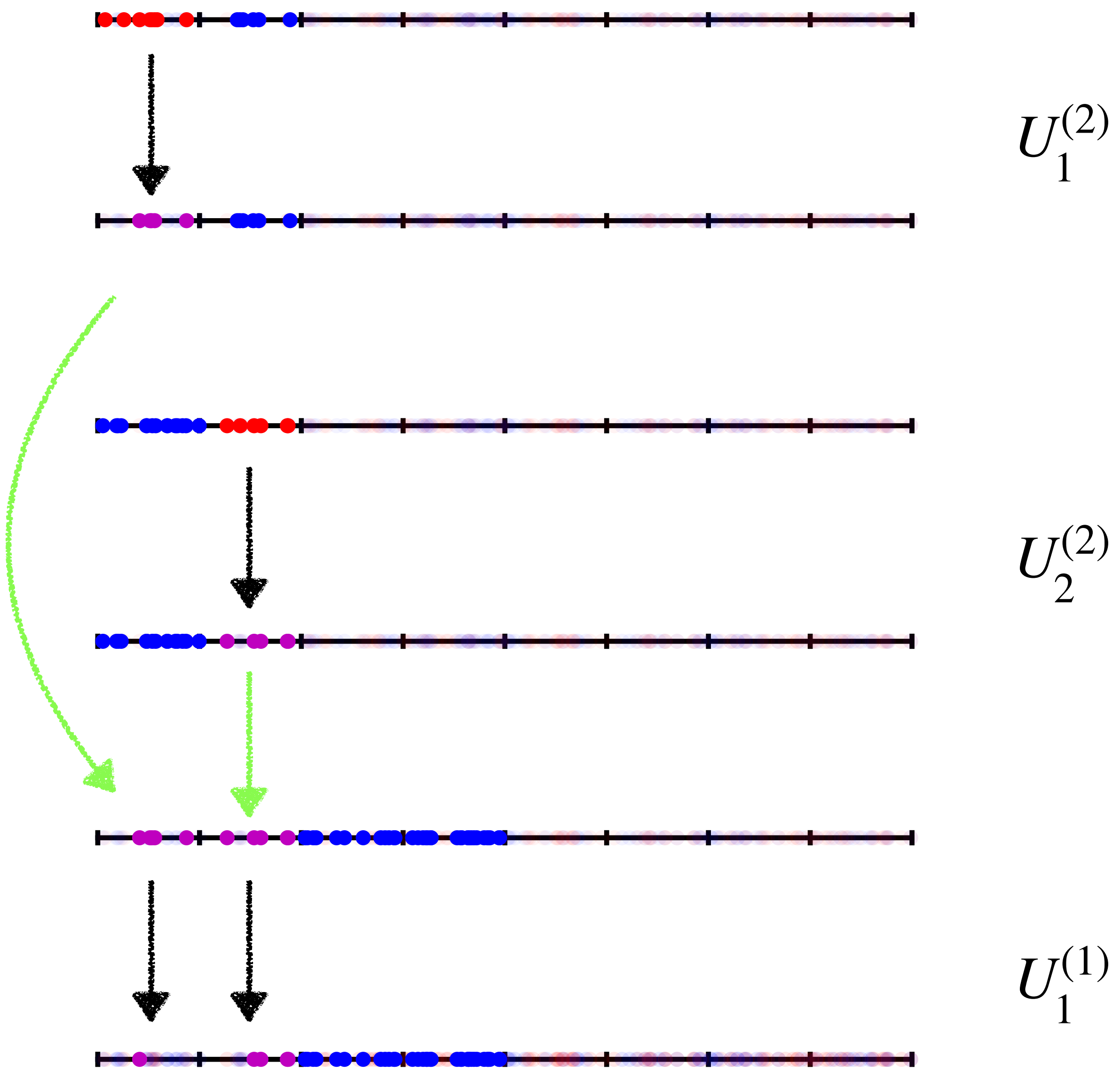
Nested basis



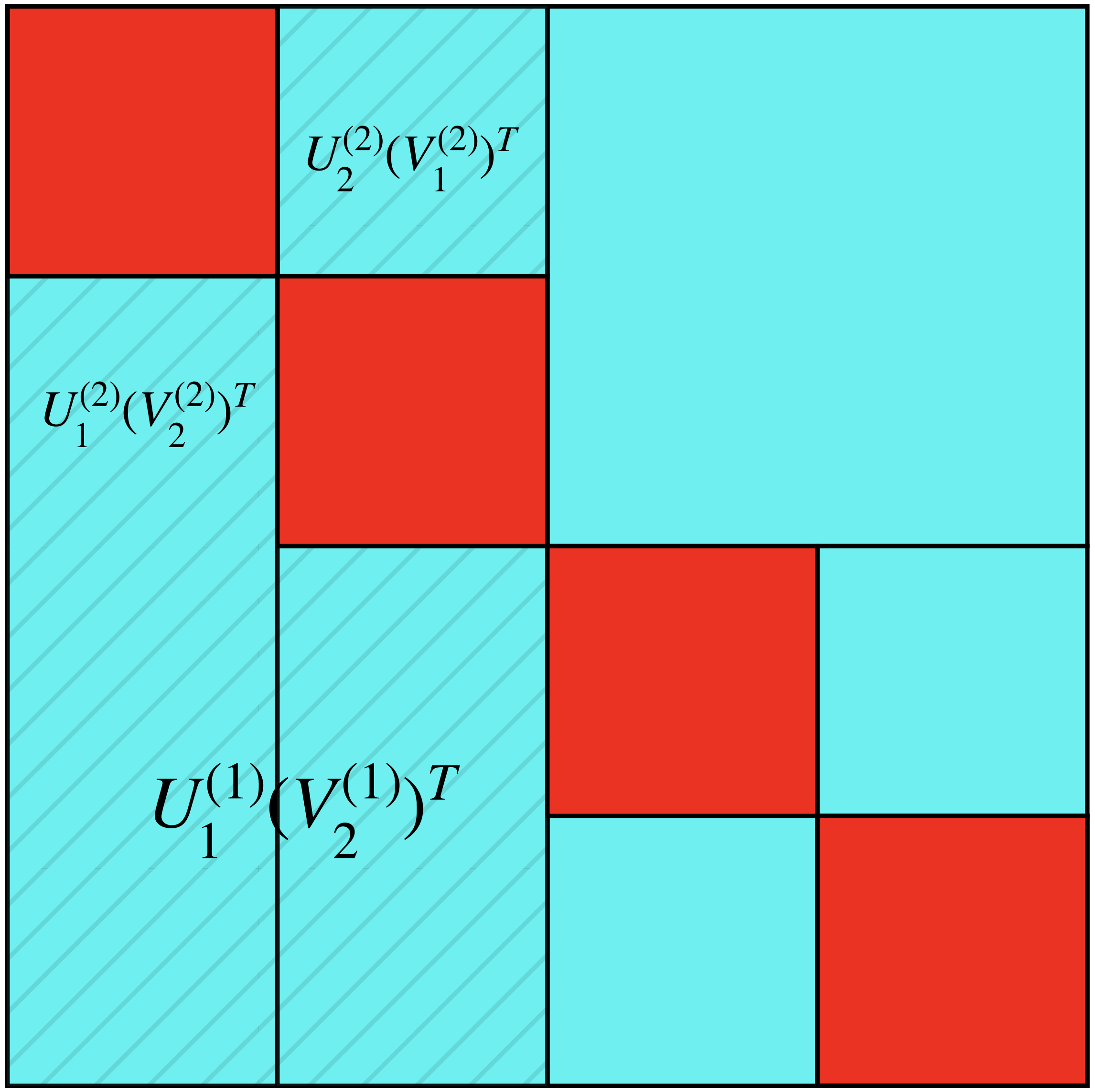
Q: Given $U_1^{(2)}$, $U_2^{(2)}$, can we compute $U_1^{(1)}$?

In general: No!

$U_i^{(j)}$ can be thought of as subset of columns of original matrix



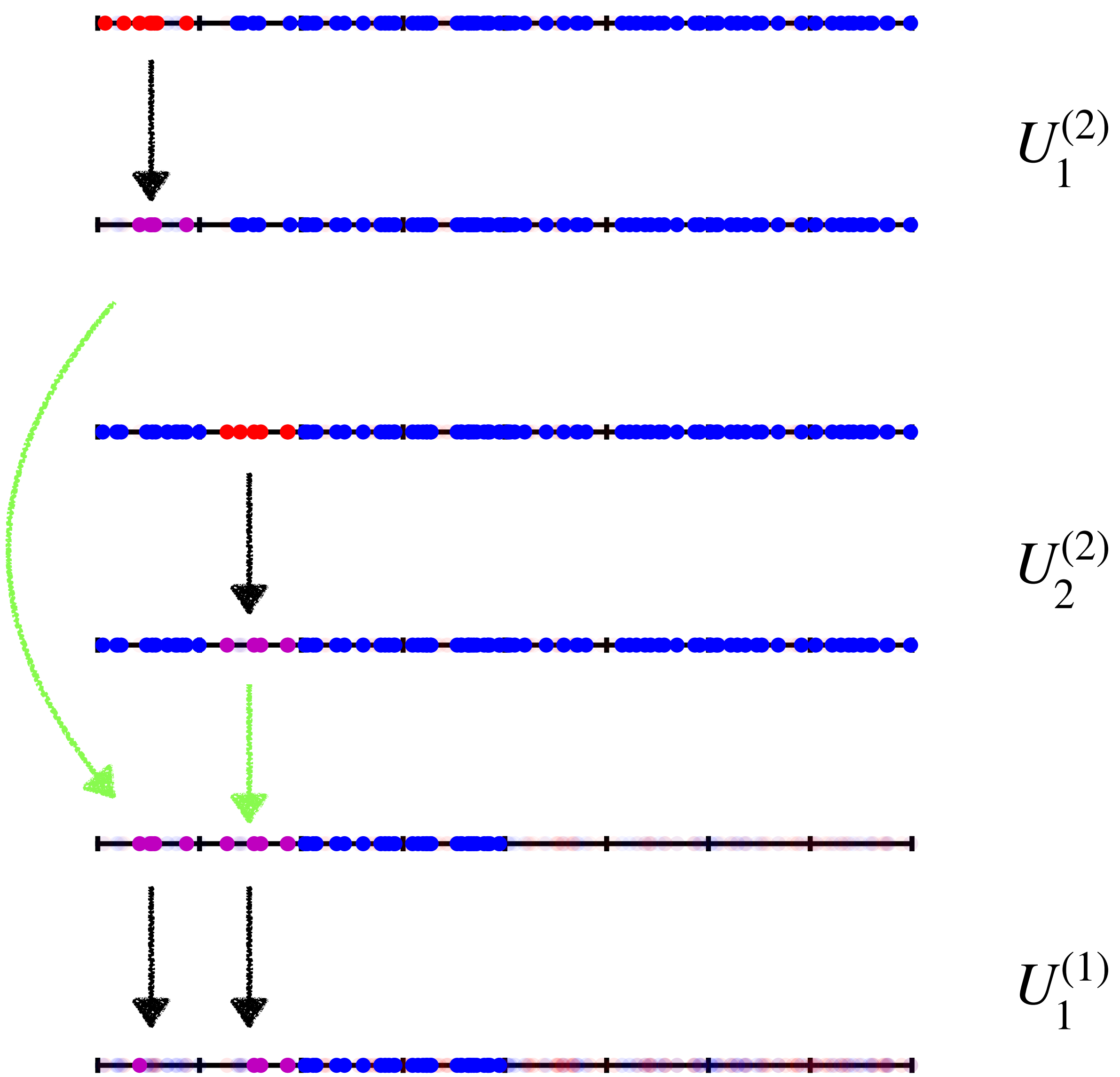
Nested basis



Q: Given $U_1^{(2)}$, $U_2^{(2)}$, can we compute $U_1^{(1)}$?

Yes, but factorization cost still $O(n^2)$

$U_i^{(j)}$ can be thought of as subset of columns of original matrix

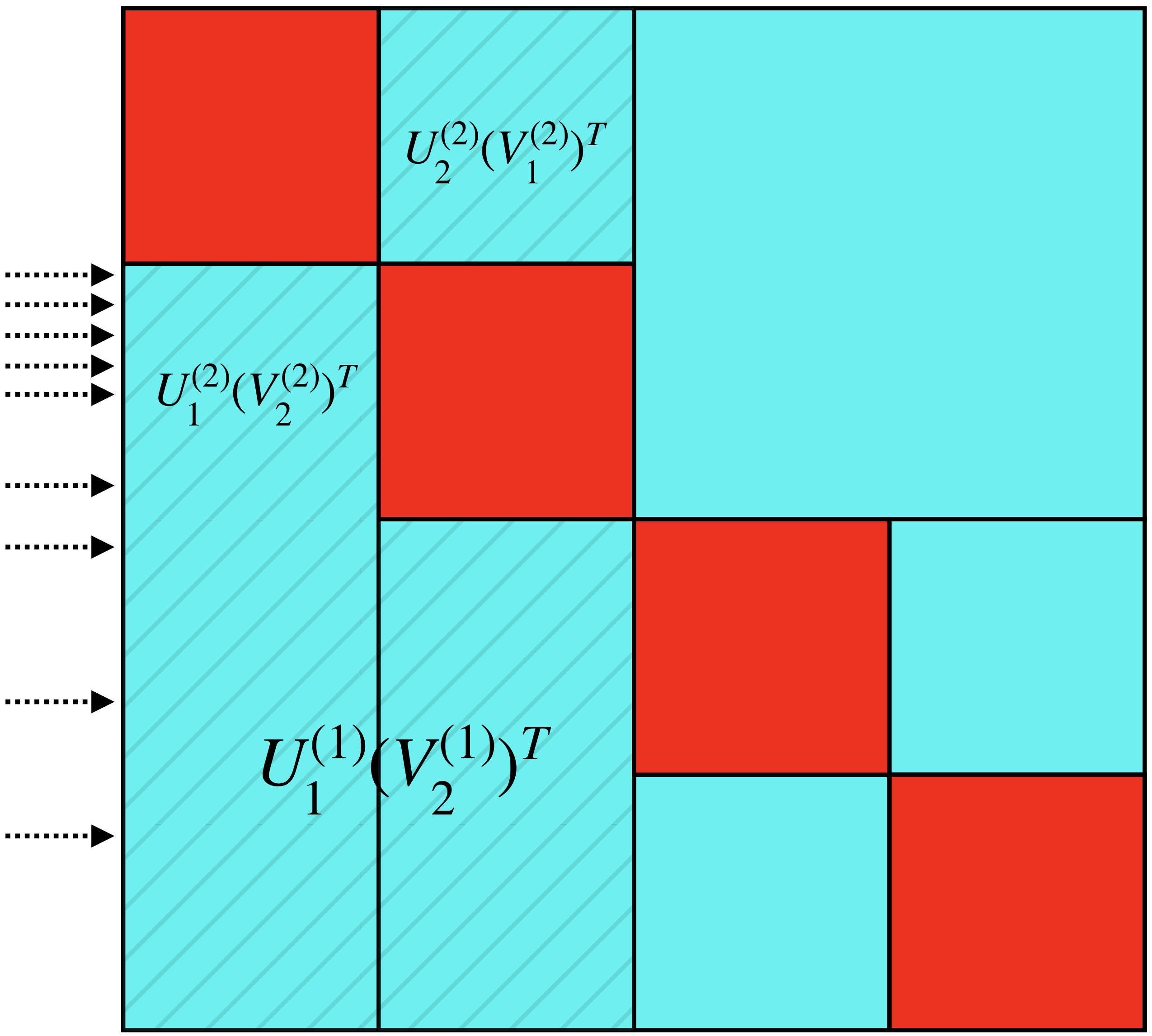


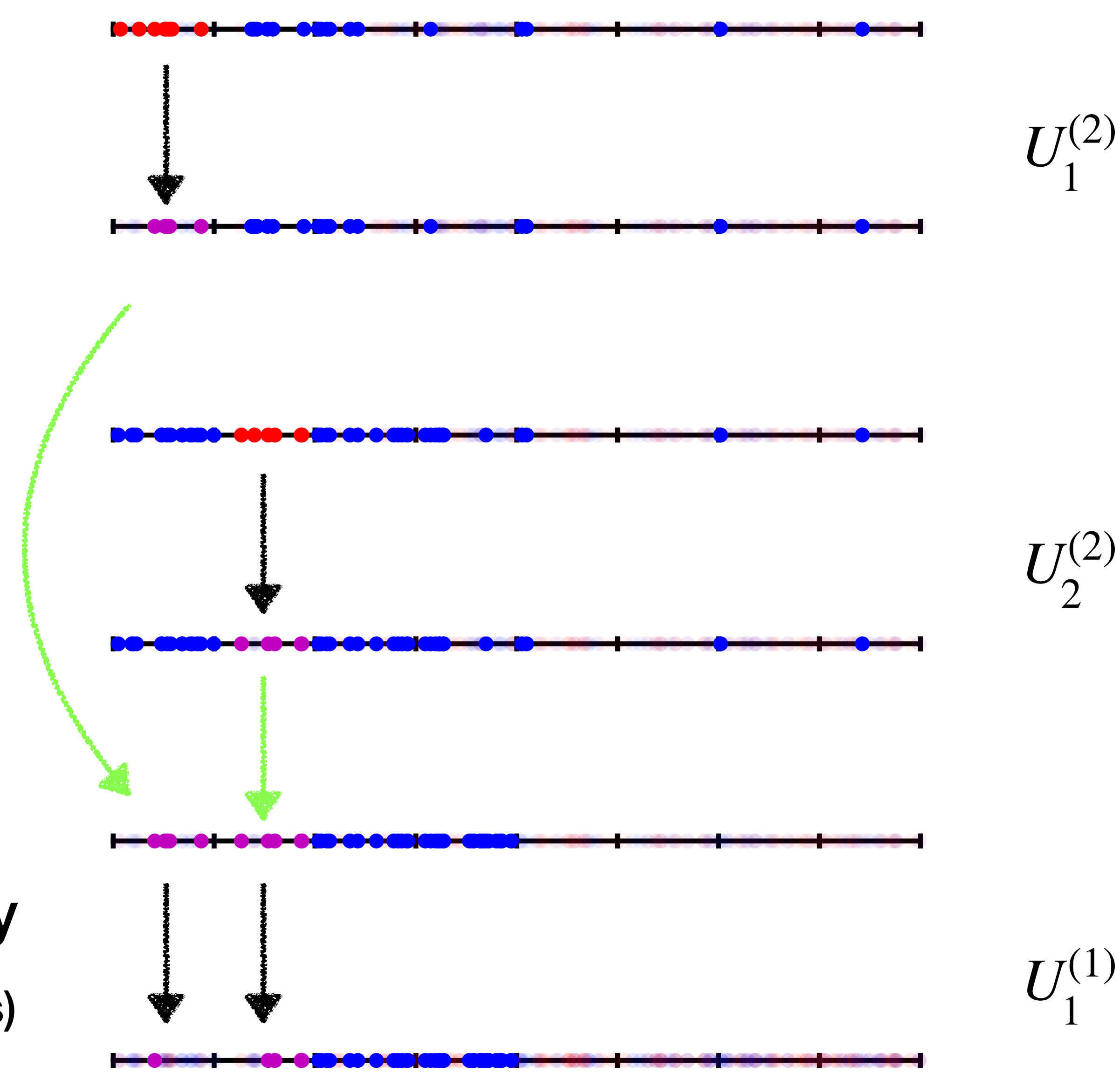
Nested basis

Q: Given $U_1^{(2)}$, $U_2^{(2)}$, can we compute $U_1^{(1)}$?

Use proxy points! Factorization cost $O(n)$

$U_i^{(j)}$ can be thought of as subset of columns of original matrix





Proofs for kernels satisfying Green's identity

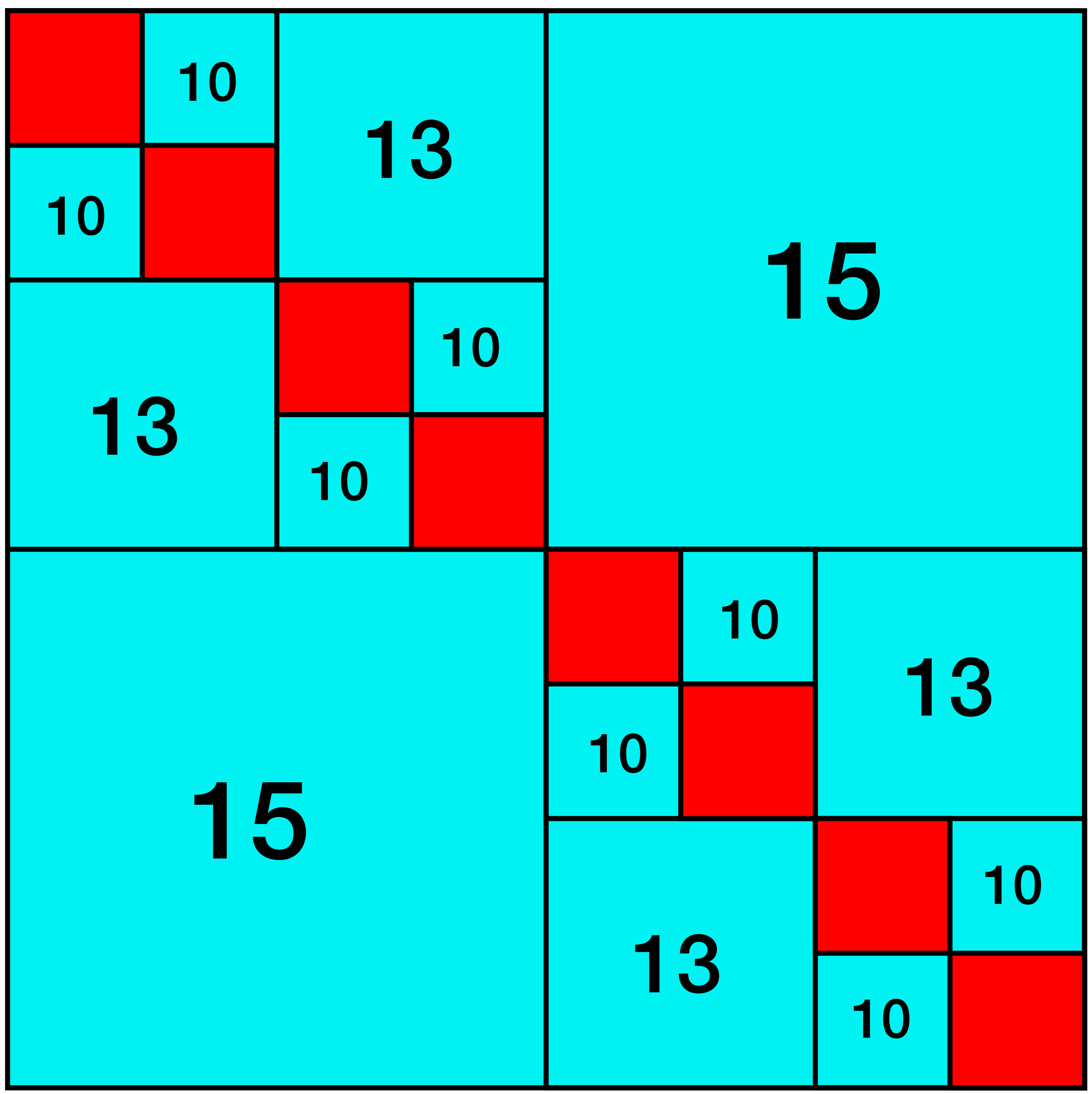
Heuristic works for larger family of kernels (e.g. Gaussians)

The need for order

Computational Task:

$$\text{argmax}_{\boldsymbol{\theta}} \mathscr{L}_{\boldsymbol{\theta}} \propto \frac{1}{\det C(\boldsymbol{t};\boldsymbol{\theta})^{1/2}} e^{-\frac{1}{2}\boldsymbol{y}^T C^{-1}(\boldsymbol{t};\boldsymbol{\theta})\boldsymbol{y}}$$

$$C(\boldsymbol{t},\boldsymbol{\theta}) = \sigma_\varepsilon^2 I + K(t,t';\boldsymbol{\theta})$$



$$K(t,t',\boldsymbol{\theta}) = \theta(0) + \frac{1}{\sqrt{\theta(1) + (t-t')^2}}$$

$$\theta(0) = 1, \theta(1) = 0.01$$

10k $t_i's$ uniformly distributed on $[0,1]$

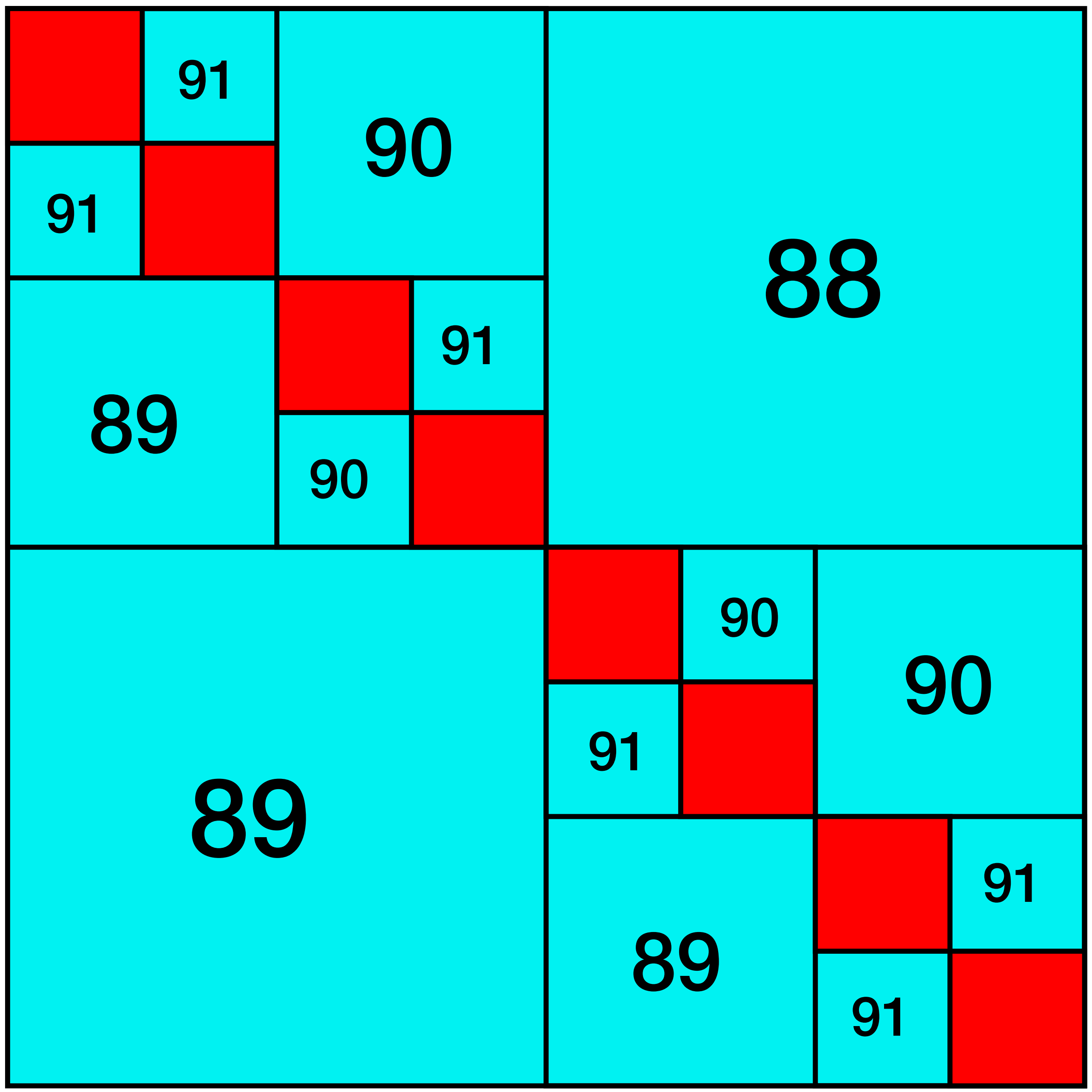
Rank structure exists only if points are sorted

The need for order

Computational Task:

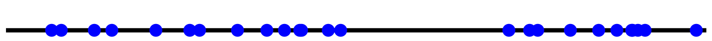
$$\text{argmax}_{\boldsymbol{\theta}} \mathscr{L}_{\boldsymbol{\theta}} \propto \frac{1}{\det C(\boldsymbol{t};\boldsymbol{\theta})^{1/2}} e^{-\frac{1}{2}\boldsymbol{y}^T C^{-1}(\boldsymbol{t};\boldsymbol{\theta})\boldsymbol{y}}$$

$$C(\boldsymbol{t},\boldsymbol{\theta}) = \sigma_\varepsilon^2 I + K(t, t'; \boldsymbol{\theta})$$



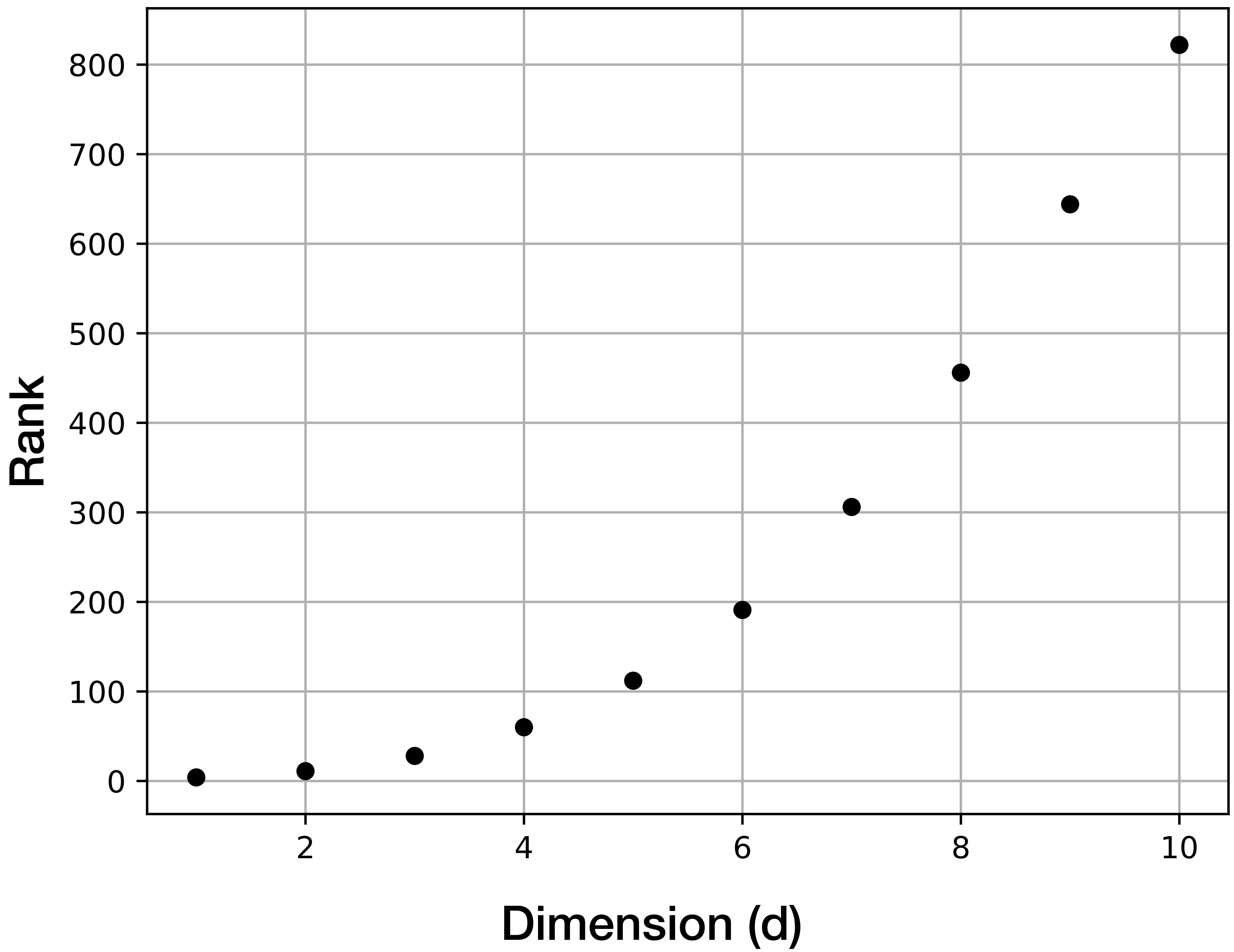
$$K(t, t', \boldsymbol{\theta}) = \theta(0) + \frac{1}{\sqrt{\theta(1) + (t - t')^2}}$$

$$\theta(0) = 1, \theta(1) = 0.01$$



10k $t_i's$ uniformly distributed on $[0,1]$

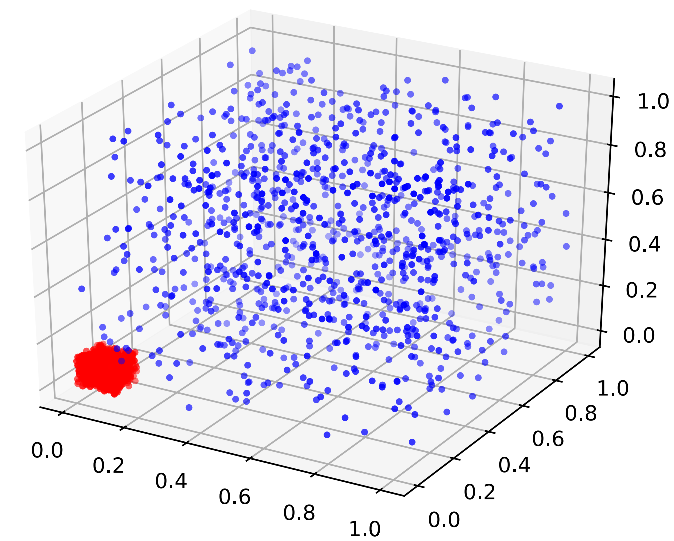
The curse of dimensionality



$$K(t, t', \boldsymbol{\theta}) = \theta(0) + e^{-\frac{|t-t'|^2}{2\theta(1)^2}}$$

$$\theta(0) = 1, \theta(1) = 3$$

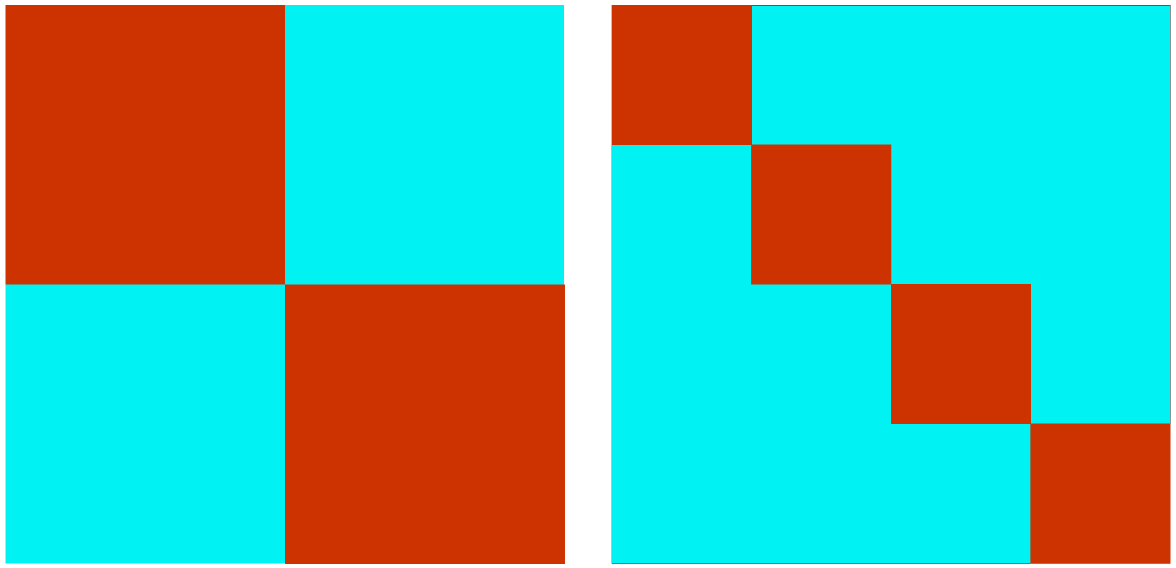
$$t, t' \in \mathbb{R}^d, \quad t' \in [0,0.125]^d, \quad t \in [0,1]^d \backslash [0,0.125]^d$$



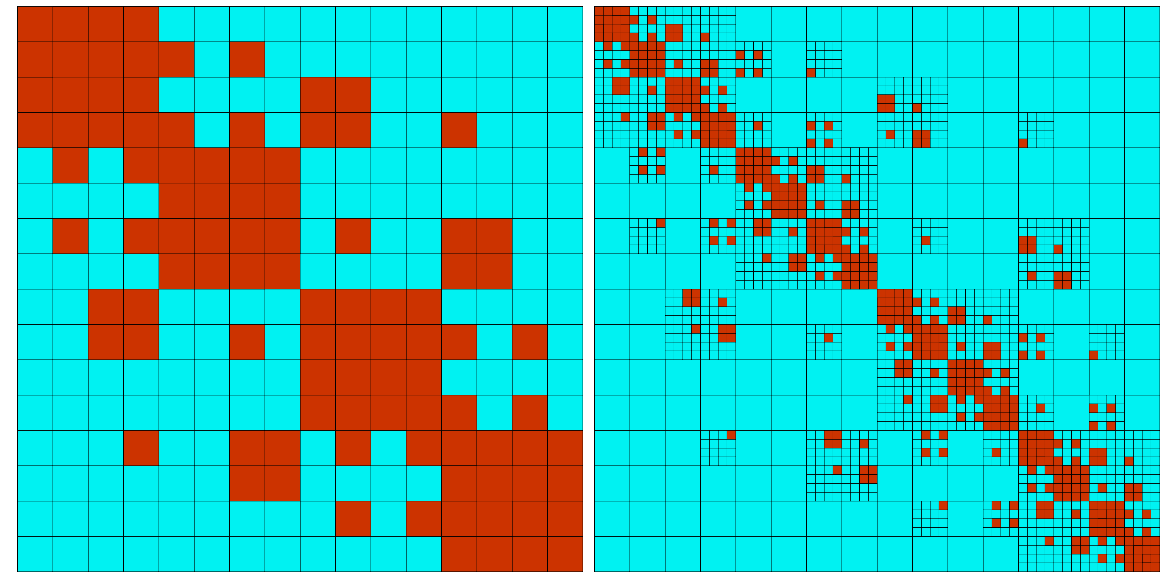
Sample distribution in 3d

Fast direct solvers in action

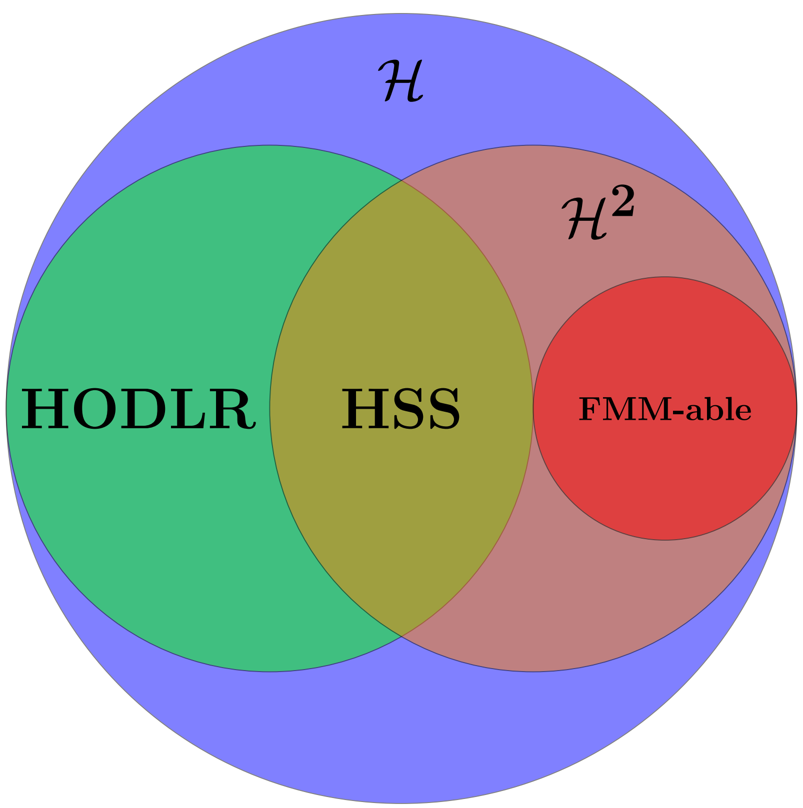
The zoo of matrix factorizations



HODLR/HSS matrices



FMM/$\mathscr{H}^2$ matrices



Butterfly/FFT matrices

Low-rank structure ↓ / Nested basis →	No	Yes
Strong	HODLR	HSS
Weak	$\mathcal{H}$	$\mathcal{H}^2$

Other applications - Neural networks

A multiscale neural network based on hierarchical nested bases

Yuwei Fan,[*] Jordi Feliu-Fabà,[†] Lin Lin,[‡] Lexing Ying,[§] Leonardo Zepeda-Núñez[¶]

Using $\mathscr{H}^2$ in layers of locally connected networks

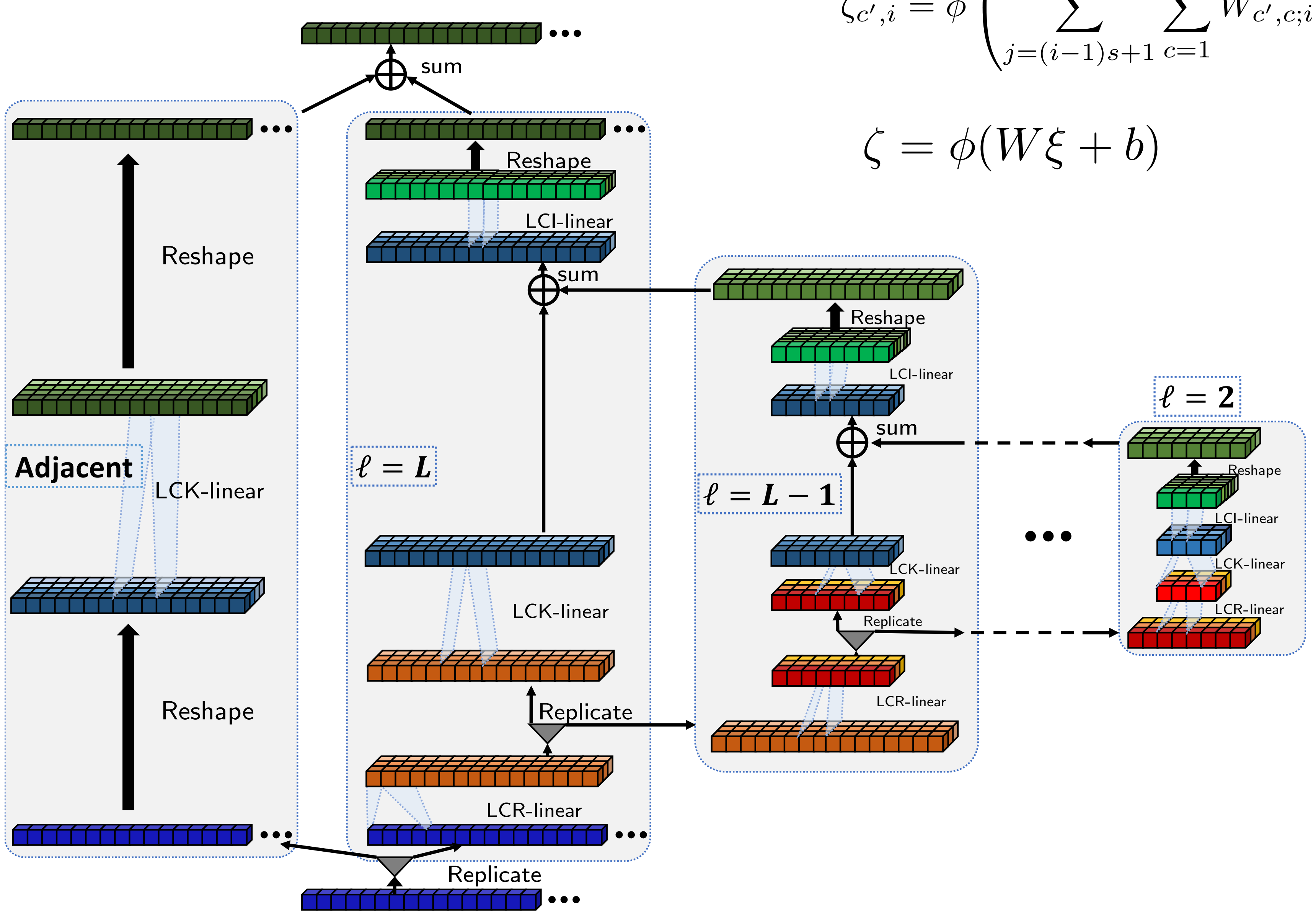
A multiscale neural network based on hierarchical matrices

Yuwei Fan,[*] Lin Lin,[†] Lexing Ying,[‡] Leonardo Zepeda-Núñez[§]

Using $\mathscr{H}$ in layers of locally connected networks

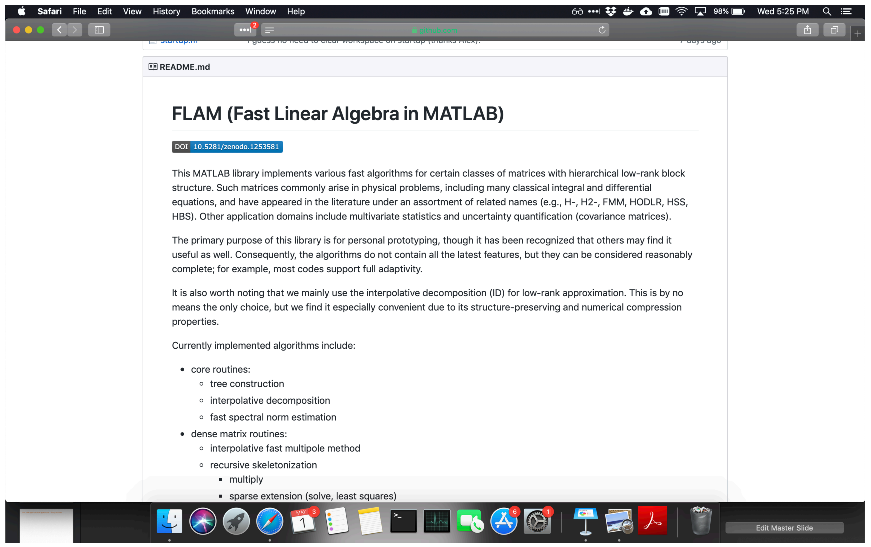
$$\zeta_{c',i} = \phi\left(\sum_{j=(i-1)s+1}^{(i-1)s+w} \sum_{c=1}^{\alpha} W_{c',c;i,j}\xi_{c,j} + b_{c',i}\right), \quad i = 1, \ldots, N_x', \; c' = 1, \ldots, \alpha'$$

$$\zeta = \phi(W\xi + b)$$

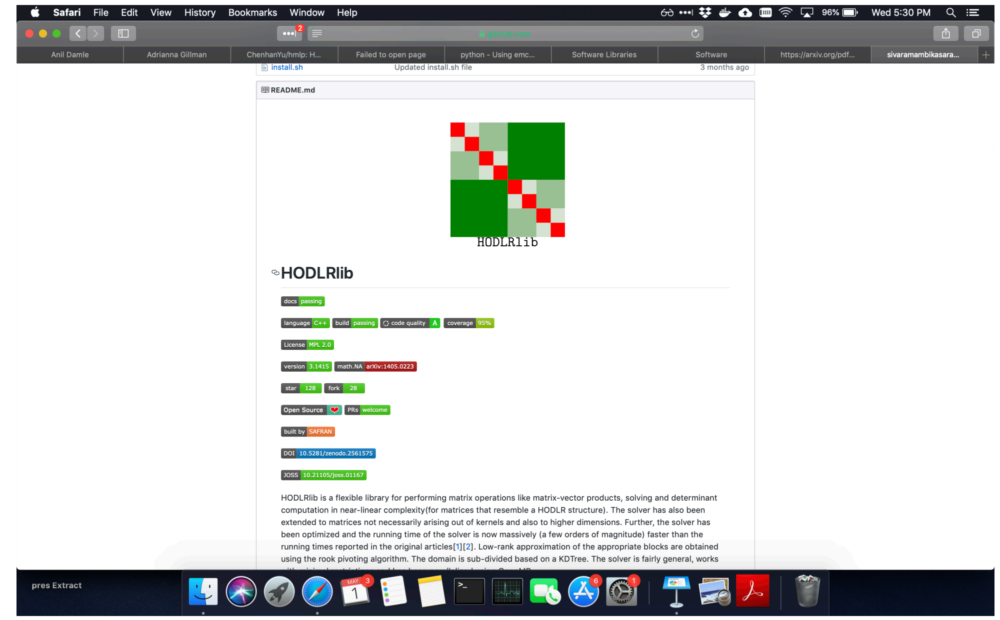


Affords fewer number of parameters in Neural net representation and fast application of the forward network

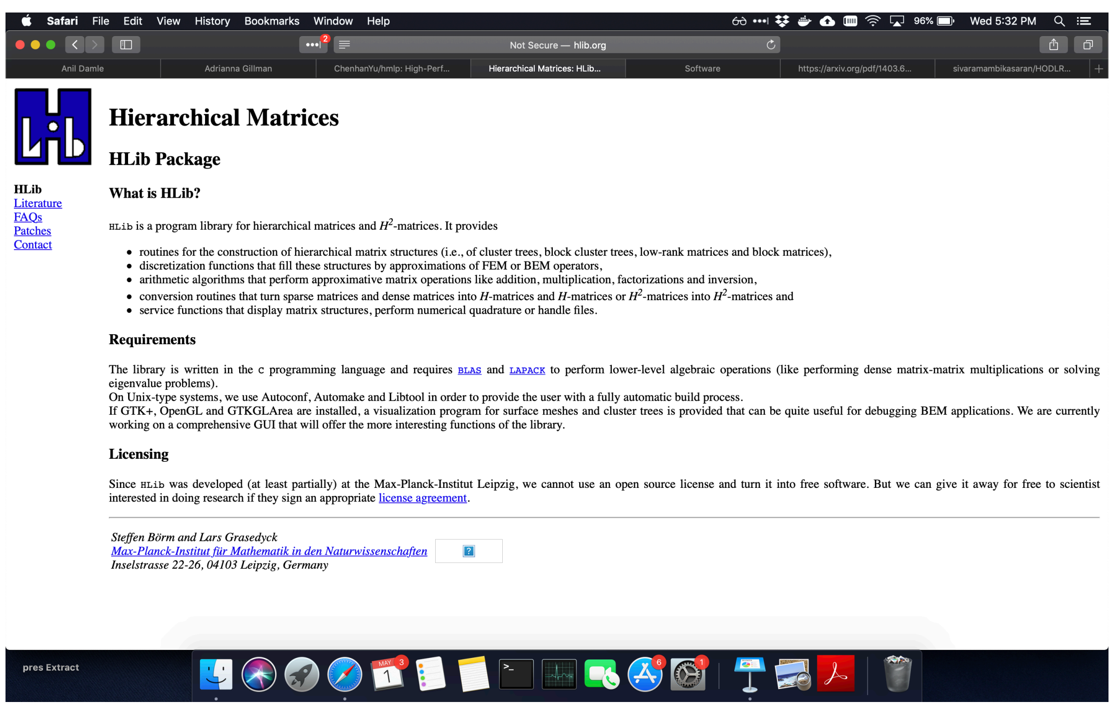
Software



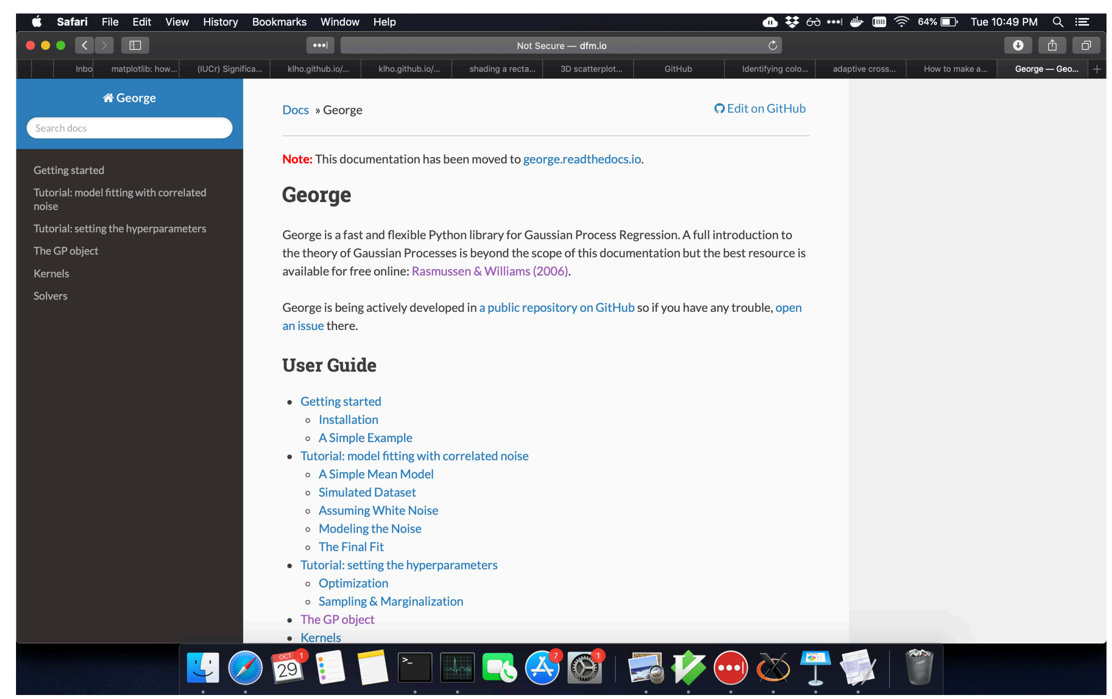
https://github.com/klho/FLAM



https://github.com/sivaramambikasaran/HODLR

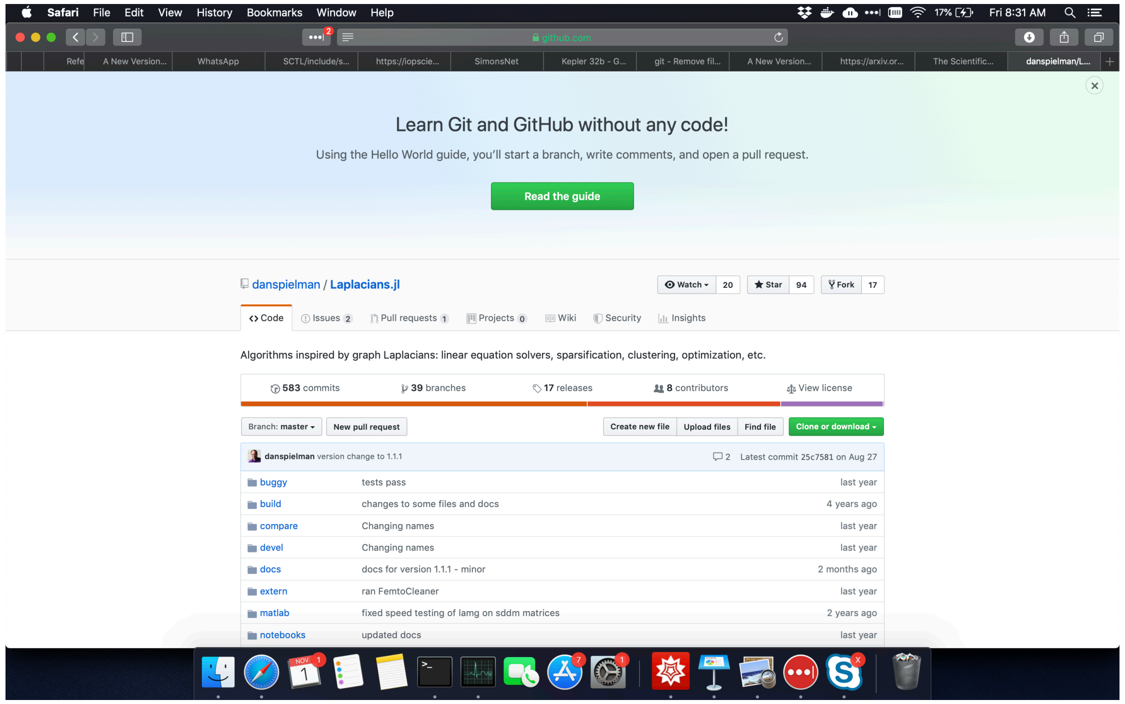


http://www.hlib.org

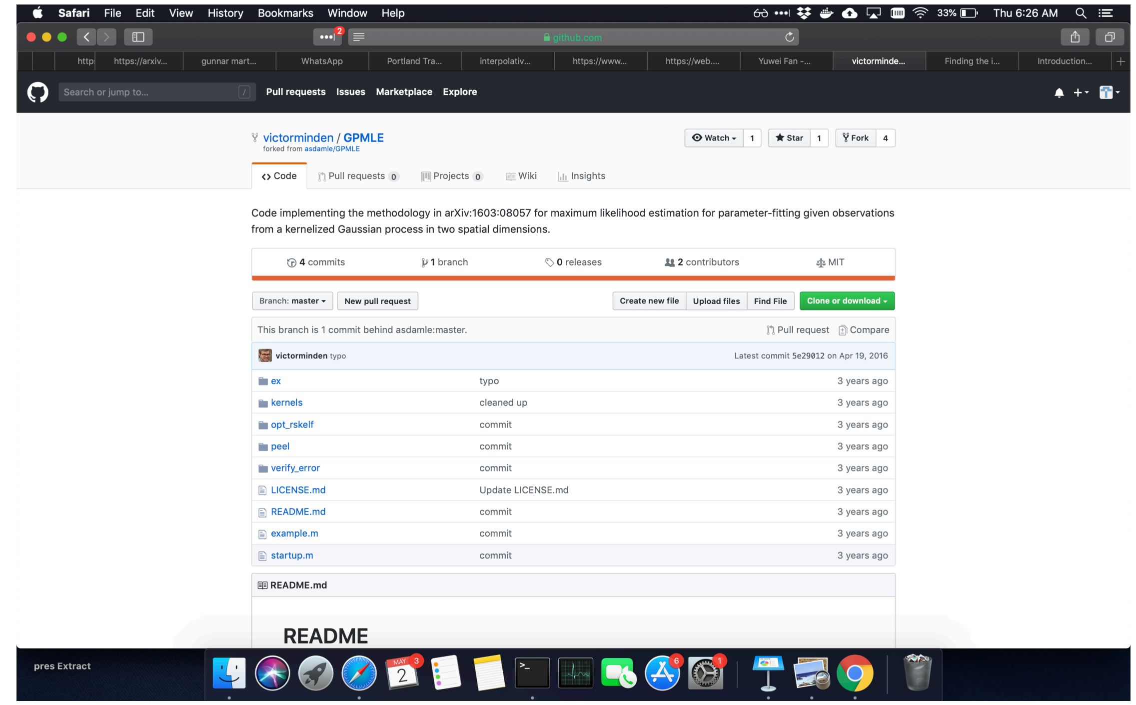


http://dfm.io/george/current/

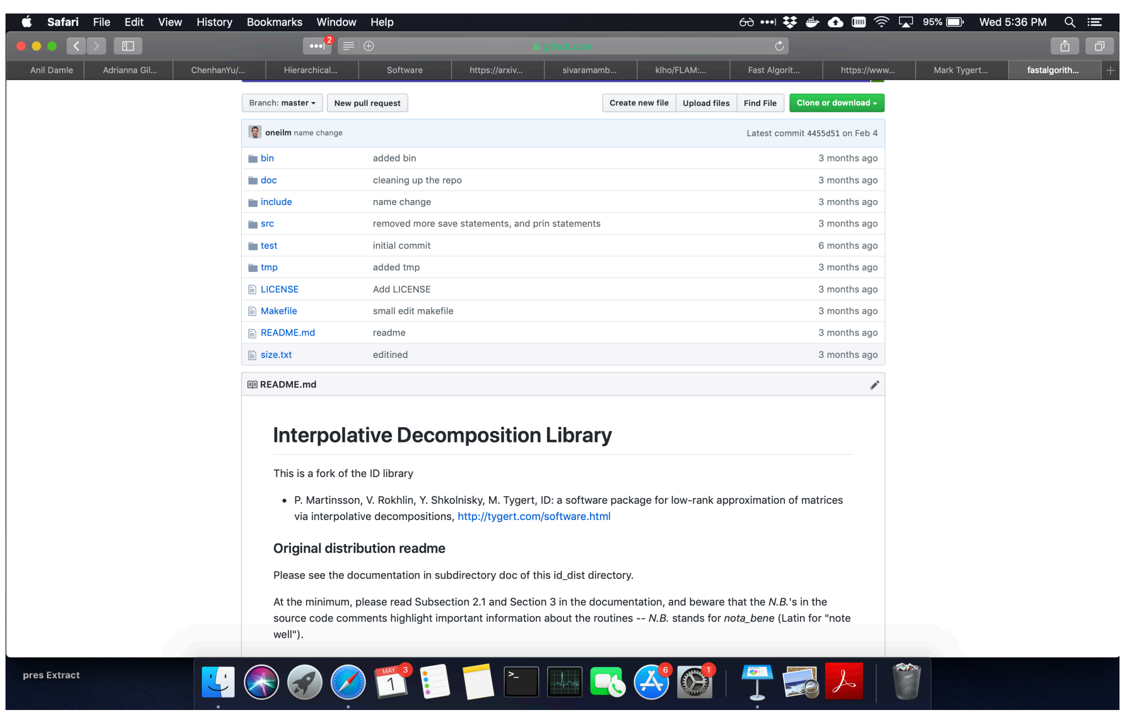
More resources



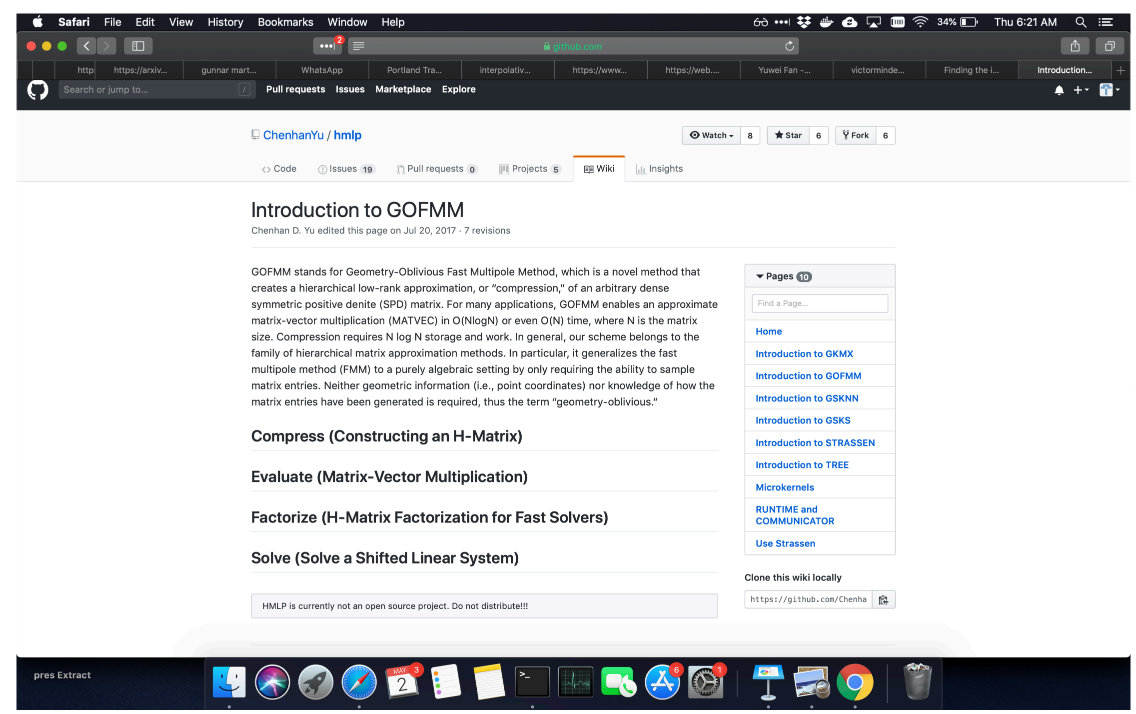
https://github.com/danspielman/Laplacians.jl



https://github.com/victorminden/GPMLE



https://github.com/fastalgorithms/libid



https://github.com/ChenhanYu/hmlp/wiki/Introduction-to-GOFMM

More resources

- Video lectures by Gunnar - https://www.youtube.com/playlist?list=PLPDZ9rcIfxyOrlpcu_D1PRcyK-o2iofwW

- Excellent review article on randomized methods for low rank approximations - Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions: https://arxiv.org/pdf/0909.4061.pdf

- Some of the illustrations courtesy: Sivaram Ambikasaran, Dan Foreman Mackey, David Hogg, Mike O'Neil, Per-Gunnar Martinsson, Ken Ho, Lesliie Greengard, Lexing Ying, Adrianna Gillman

Even more references

- Ho, Kenneth L., and Leslie Greengard. "A fast direct solver for structured linear systems by recursive skeletonization." *SIAM Journal on Scientific Computing* 34.5 (2012): A2507-A2532.
- Yu, Chenhan D., et al. "Geometry-oblivious FMM for compressing dense SPD matrices." *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017.
- Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D. W., & O'Neil, M. (2016). Fast direct methods for Gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, *38*(2), 252-265.
- Gillman, A., Young, P. M., & Martinsson, P. G. (2012). A direct solver with O (N) complexity for integral equations on one-dimensional domains. *Frontiers of Mathematics in China*, *7*(2), 217-247.
- Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, *53*(2), 217-288.
- Greengard, L., & Rokhlin, V. (1991). On the numerical solution of two−point boundary value problems. *Communications on Pure and Applied Mathematics*, *44*(4), 419-452.
- Fan, Y., Lin, L., Ying, L., & Zepeda-Núnez, L. (2018). A multiscale neural network based on hierarchical matrices. *arXiv preprint arXiv:1807.01883*.
- Minden, V., Damle, A., Ho, K. L., & Ying, L. (2017). Fast spatial gaussian process maximum likelihood estimation via skeletonization factorizations. *Multiscale Modeling & Simulation*, *15*(4), 1584-1611.
- Martinsson, P. G., Rokhlin, V., Shkolnisky, Y., & Tygert, M. (2008). ID: A software package for low-rank approximation of matrices via interpolative decompositions, Version 0.2.
- Corona, E., Martinsson, P. G., & Zorin, D. (2015). An O (N) direct solver for integral equations on the plane. *Applied and Computational Harmonic Analysis*, *38*(2), 284-317.
- Gimbutas, Z., & Rokhlin, V. (2003). A generalized fast multipole method for nonoscillatory kernels. *SIAM Journal on Scientific Computing*, *24*(3), 796-817.

Not an exhaustive list

Thank you!

Questions?