Fast spatial Gaussian process maximum likelihood estimation via skeletonization factorizations

(with a focus on numerical linear algebra)

Victor Minden, Stanford ICME

Joint work with Ken L. Ho, Anil Damle, Lexing Ying

July, 2016

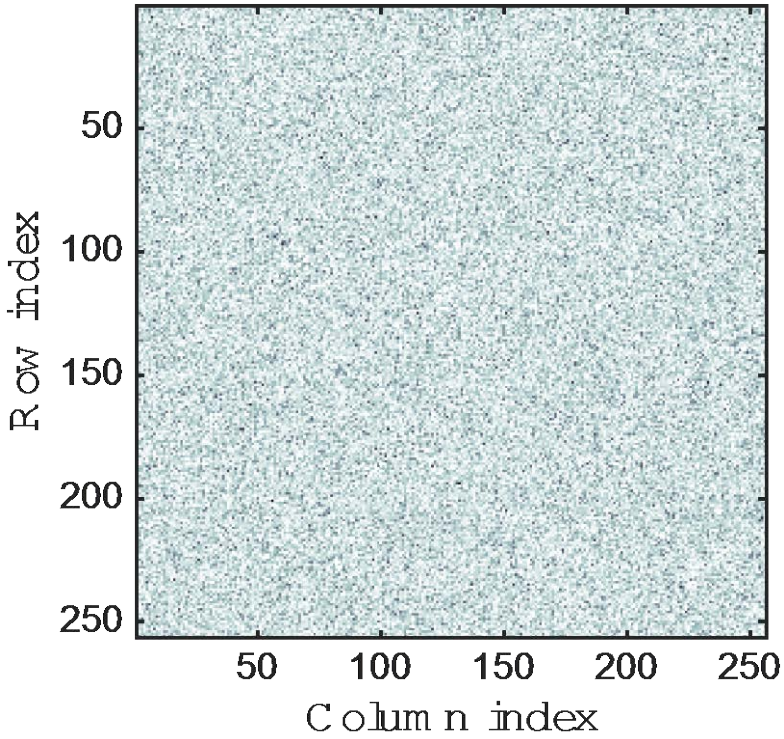
Prologue: some matrices have structure

- Non-example:

  A = randn(256,256);

- This arbitrary random matrix does not have the structure I am talking about.

- Solving $Ax=b$ with a matrix like this would be expensive.

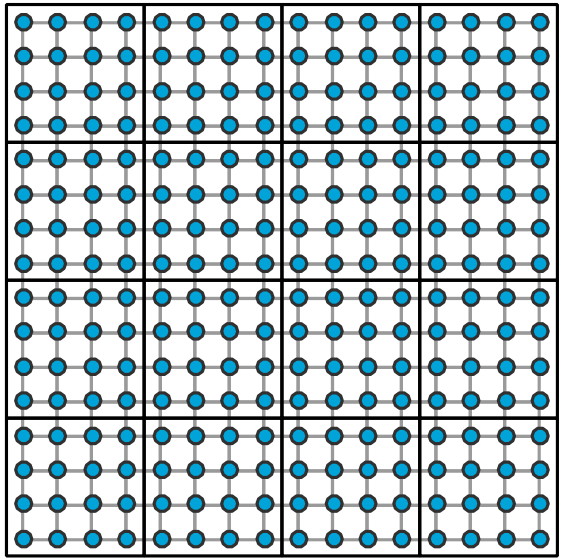


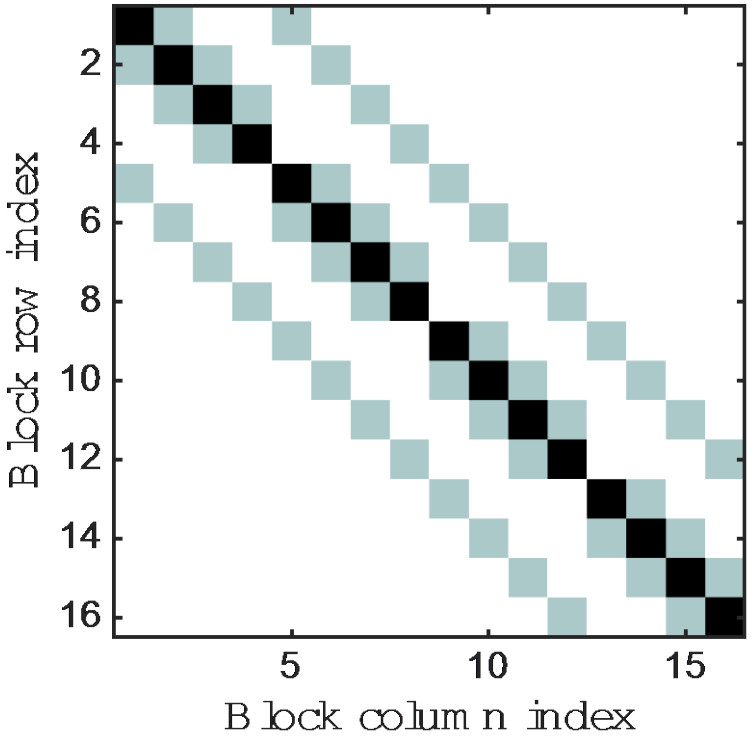
Prologue: some matrices have structure

- Example:

  Finite difference matrix (sparse)

$$-\nabla \cdot \big(a(x)\nabla u(x)\big) + b(x)u(x) = f(x)$$

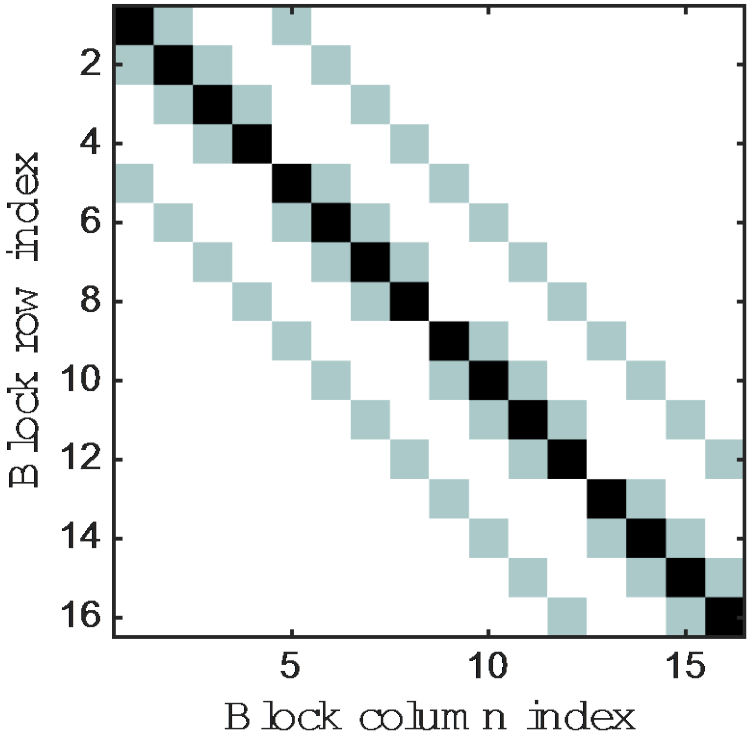




Prologue: some matrices have structure

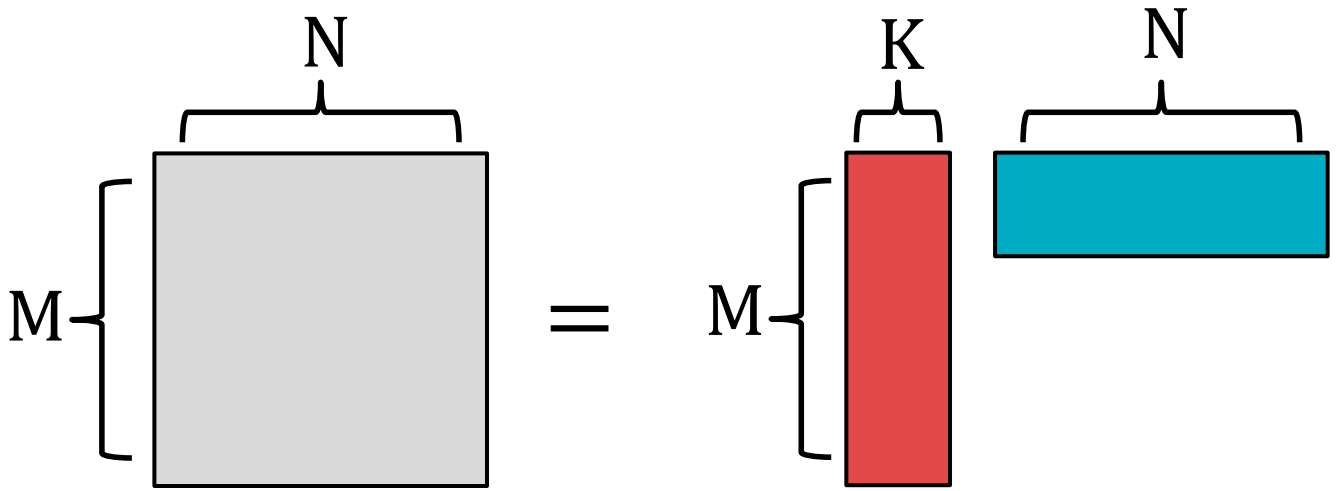
- When we have this structure, we have fast algorithms.
- Example: Nested dissection algorithm for solving Ax=b for A like this [George, 1973].
- New solve cost is much less expensive!

$\mathcal{O}(N^{1.5})$ or $\mathcal{O}(N^2)$ versus $\mathcal{O}(N^3)$



Prologue: dense matrices can have structure too

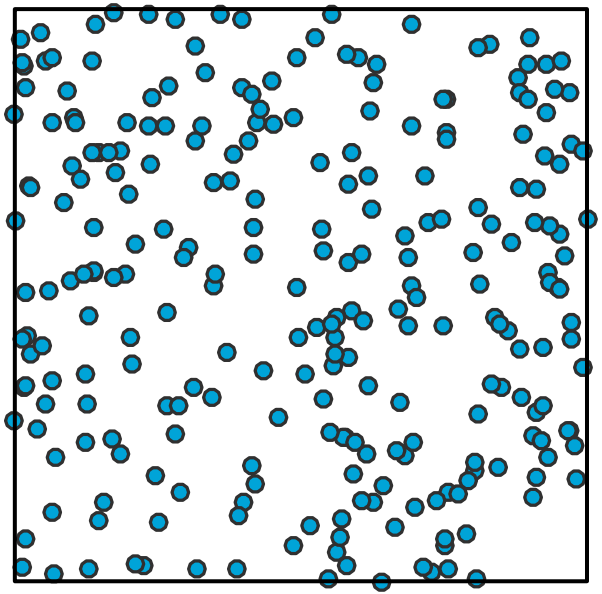
- Example: low-rank matrix (step 0 on our way to structure)
  - $\mathcal{O}(MN) \Longrightarrow \mathcal{O}(MK + NK)$

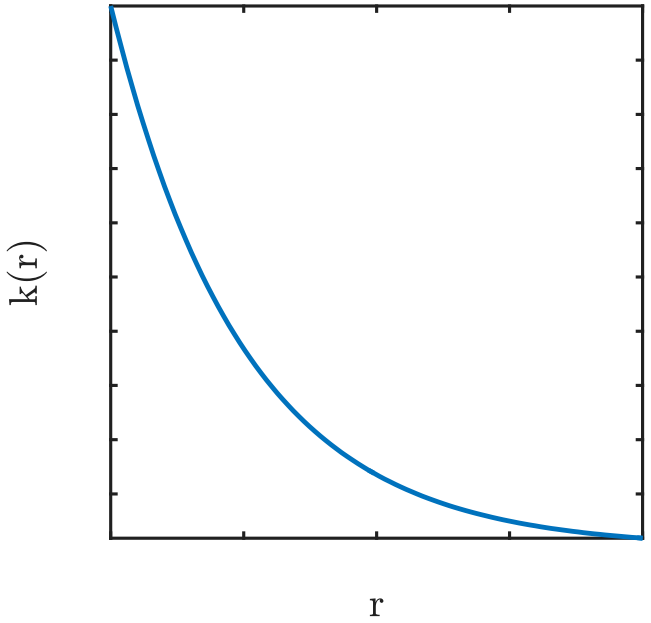


Prologue: dense matrices can have structure too

- Example: Start with scattered points and a kernel function

$$k(x_i, x_j) = k(|x_i - x_j|) = \exp(-|x_i - x_j|)$$

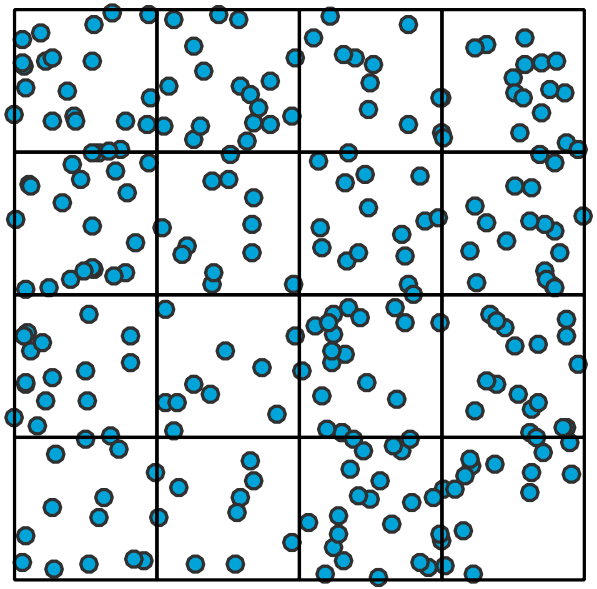


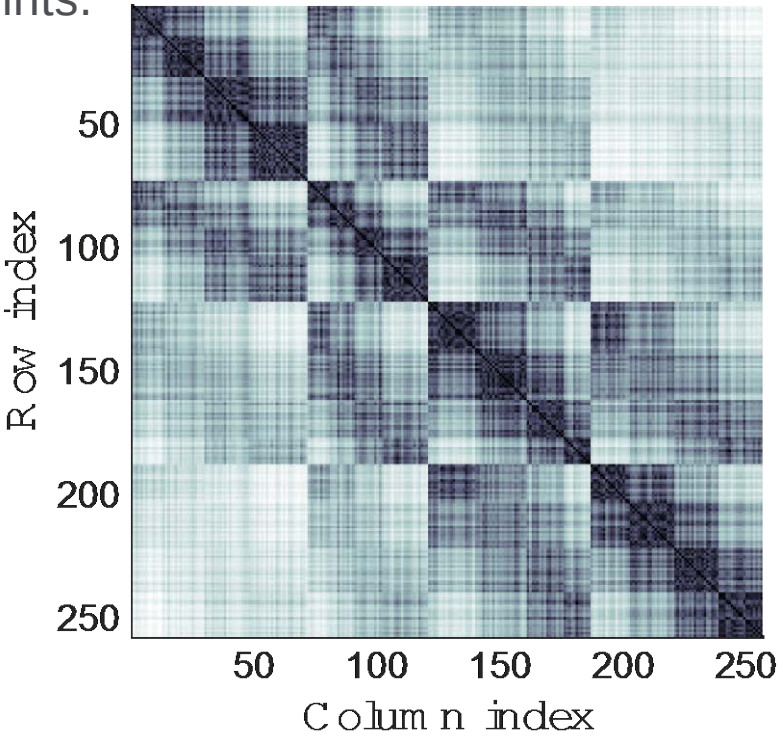


Prologue: dense matrices can have structure too

- Example: Kernel matrices on 2D points.

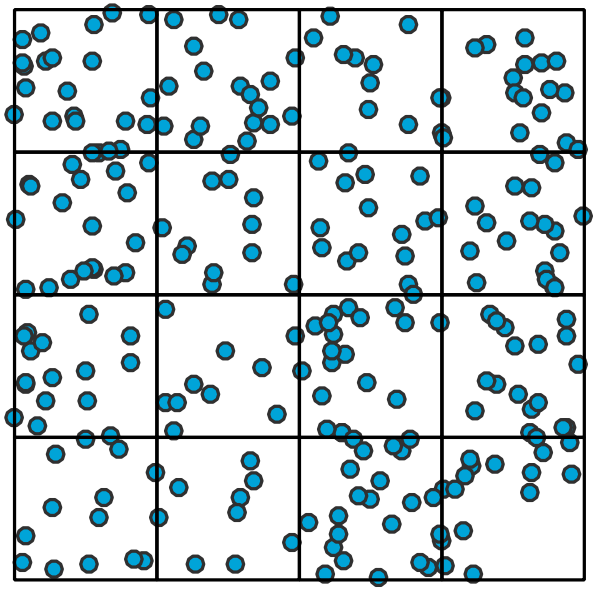
$$A_{ij} = k(x_i, x_j)$$

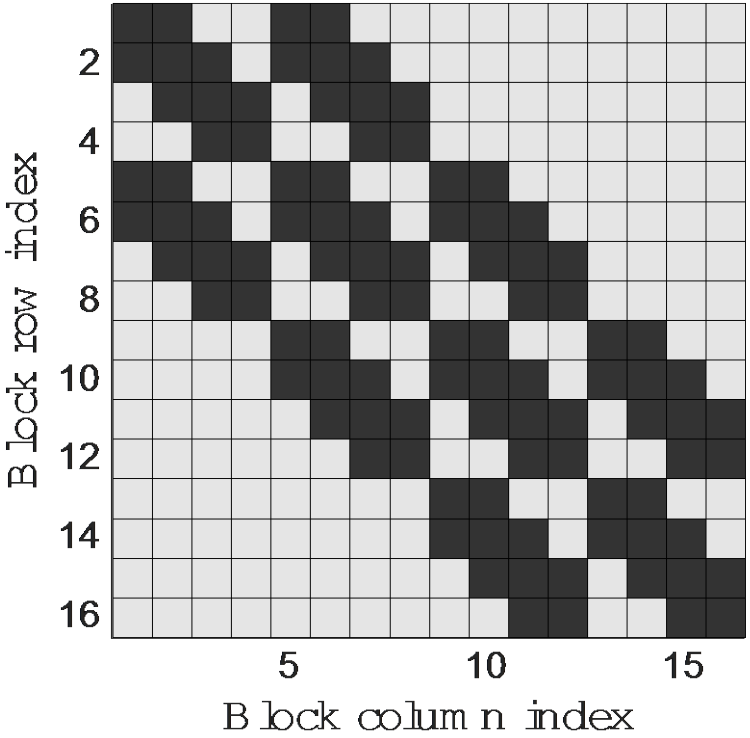




Prologue: dense matrices can have structure too

- Dark gray: full-rank
- Light gray: lower-rank





Wait, what does this have to do with Gaussian processes?

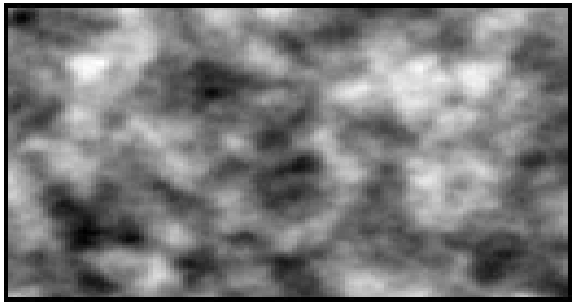
Gaussian process (GP) Model in 2D

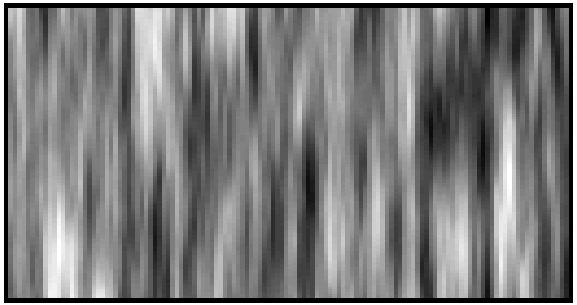
- Field is a function of space, $\mathcal{Z}: \mathbb{R}^2 \rightarrow \mathbb{R}$
- Finite-dimensional distributions are multivariate normal

$$(z_1, \ldots, z_N)^T \sim N(0, \Sigma)$$

for any collection of observations $\{z_i\} = \{\mathcal{Z}(x_i)\}$

Covariances given by some kernel function $\Sigma_{ij} = k(x_i, x_j; \theta)$





Using GPs to model functions of space

- "Kriging" in geostatistics

Real functions are (usually) not random, but can think of GP regression as:

- Bayesian prior over functions
- Radial basis function interpolation

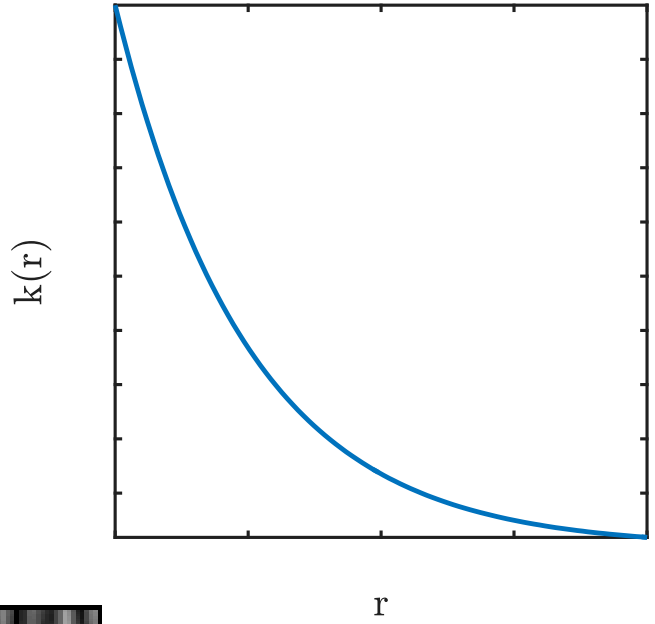
Best linear unbiased predictor (under suitable assumptions)

- Let $\mathcal{O} \subset \mathbb{R}^2$ be the set of finite observation locations with function values $\mathcal{Z}(\mathcal{O})$. Then for any finite $\mathcal{A} \subset \mathbb{R}^2$, we have the estimate based on conditional expectation

$$\hat{\mathcal{Z}}(\mathcal{A}) = \mathbb{E}[\mathcal{Z}(\mathcal{A}) \mid \mathcal{Z}(\mathcal{O})] = \Sigma_{\mathcal{A}\mathcal{O}}\Sigma_{\mathcal{O}\mathcal{O}}^{-1}\mathcal{Z}(\mathcal{O})$$

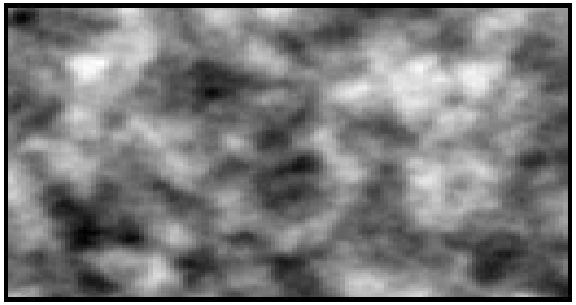
Parameterizing the covariance

Choice of kernel function is very important for GP regression

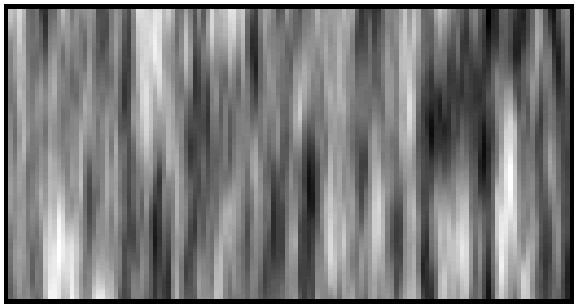


Parameterize kernel for flexibility, *e.g.*,

$$|x - y|_\theta^2 = \frac{(x_1 - y_1)^2}{\theta_1^2} + \frac{(x_2 - y_2)^2}{\theta_2^2}$$



$\theta = [7, 10]$



$\theta = [30, 3]$

Maximum likelihood estimation

- Goal: efficient method for finding maximum likelihood estimate for parameters of GP

$$\hat{\theta}_{MLE} = \operatorname{argmax} \ell(\theta)$$

- To use gradient-based optimization, need to compute a few things with kernel matrices

$$\ell(\theta) = -\mathrm{z}^{\mathrm{T}}\Sigma^{-1}z \; - \log|\Sigma|$$

$$\partial_{\theta_i}\ell(\theta) = z^T\Sigma^{-1}\Sigma_i\Sigma^{-1}z \; - \mathrm{Tr}(\Sigma^{-1}\Sigma_i)$$

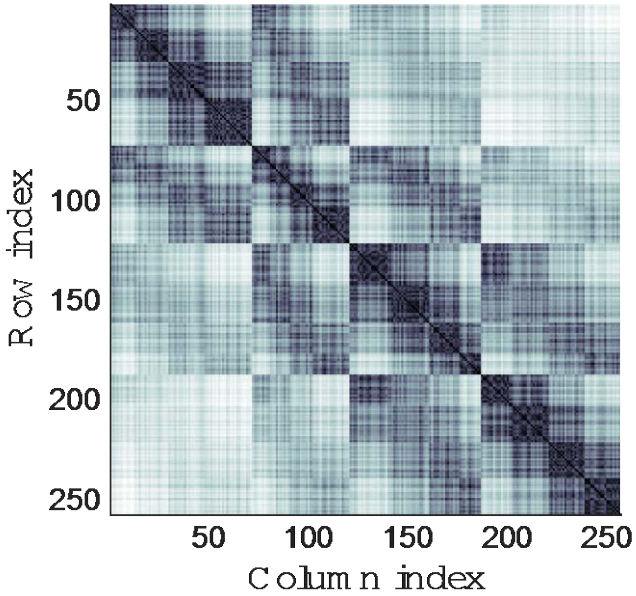
Typically $O(N^3)$ using naïve approach, but can also

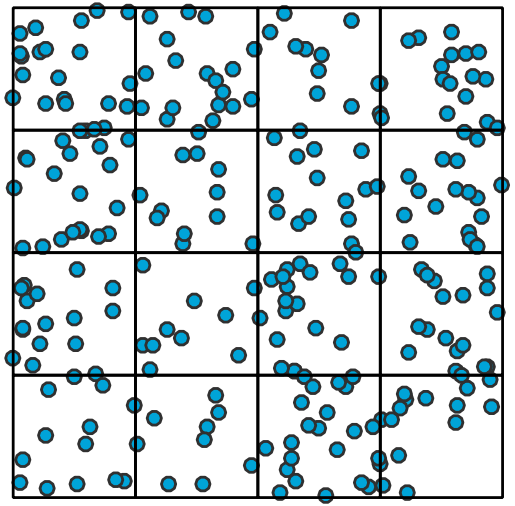
- "Taper" (threshold) Σ [Furrer et al., 2006]
- Write Σ as sparse (or diagonal) plus low-rank [Cressie & Johannesson, 2008] [Sang & Huang, 2012]
- Use a different statistical estimator [Anitescu et al., 2012] [Stein et al., 2013] [Vecchia et al.,1988]

Our approach (others exist)

At each iteration of a black-box optimization routine, use rank-structured matrix algorithms to efficiently compute likelihood and gradient.

(See also recent work by Litvinenko, Genton, Keyes and collaborators, and specifically [Castrillón-Candás et al., 2015] and [Sun & Stein, 2016])





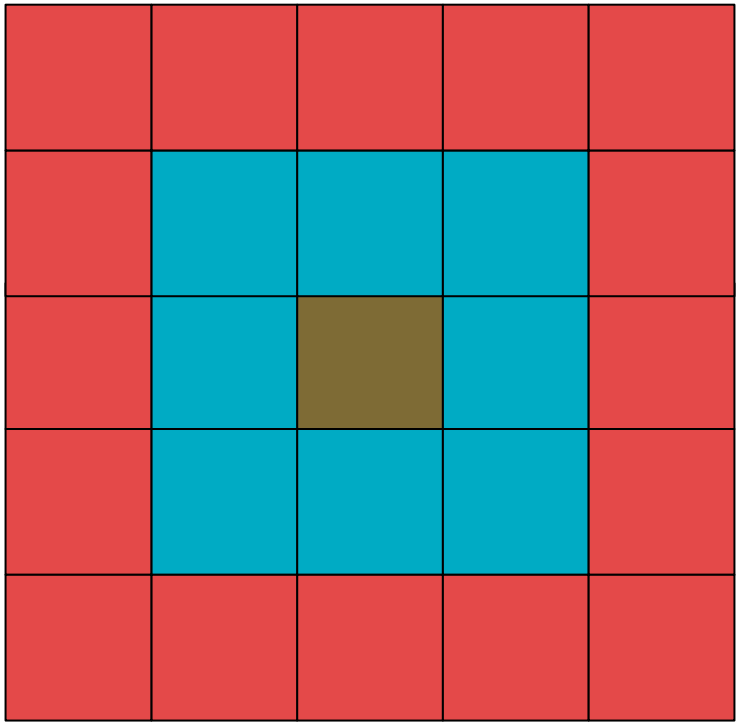
Factorization: “There are many like it, but this one is ours”

Structured matrix algorithms have a rich history.

Important examples:

- Hierarchically semi-separable / block-separable (HSS / HBS) matrices
  (see [Chandrasekaran et al., 2006 & 2007], [Xia et al., 2012], [Martinsson & Rokhlin, 2005], and [Gillman et al., 2012])
- Hierarchically off-diagonal low rank (HODLR) matrices
  (see [Martinsson, 2008] and [Ambikasaran & Darve, 2013])
- General $\mathcal{H}$-matrix algebra of Hackbusch and collaborators
  (see book [Bebendorf, 2008] for thorough treatment)
- Inverse fast multipole method (IFMM)
  (see [Coulier et al., 2015])

What does an algorithm look like? (bear with me)



AKA “how does Victor spend his days?”

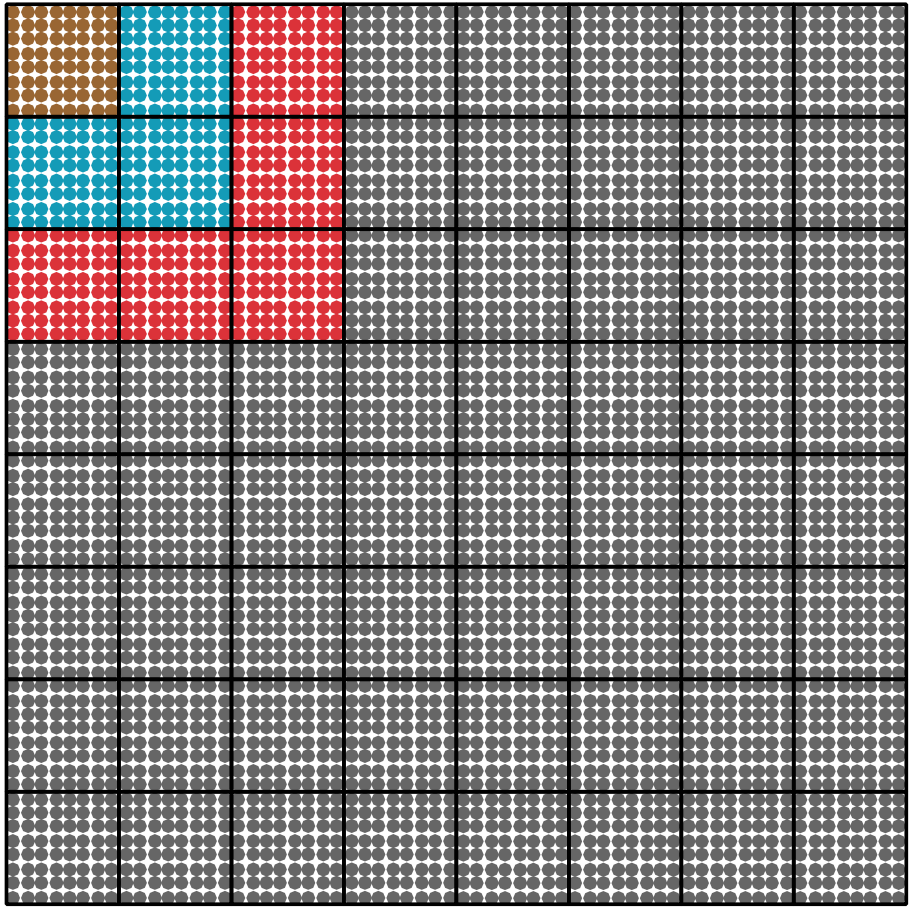
- **Brown**: box b
- **Blue**: near-field neighbors of b
- **Red**: far-field neighbors of b

$$\left[\begin{array}{c|c|c} A_{bb} & A_{bn} & A_{bf} \\ \hline A_{nb} & A_{nn} & A_{nf} \\ \hline A_{fb} & A_{fn} & A_{ff} \end{array}\right]$$

Nested dissection-like

- Box 1 (before)

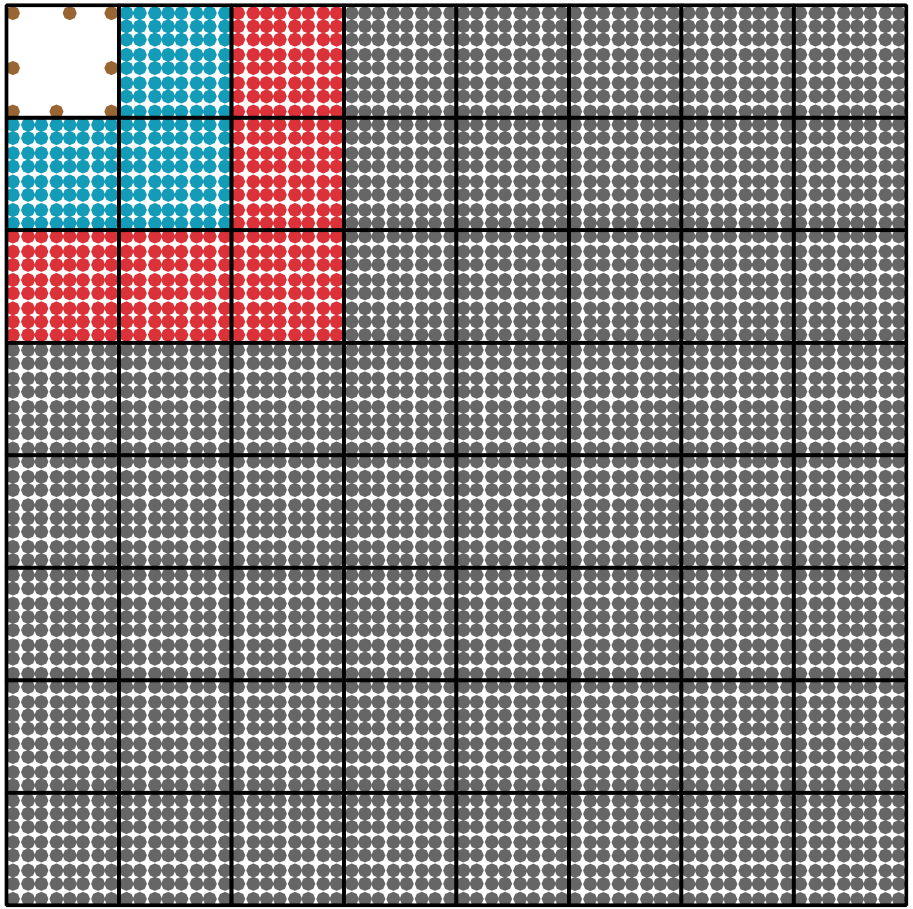
Domain:



Nested dissection-like

- Box 1 (after)

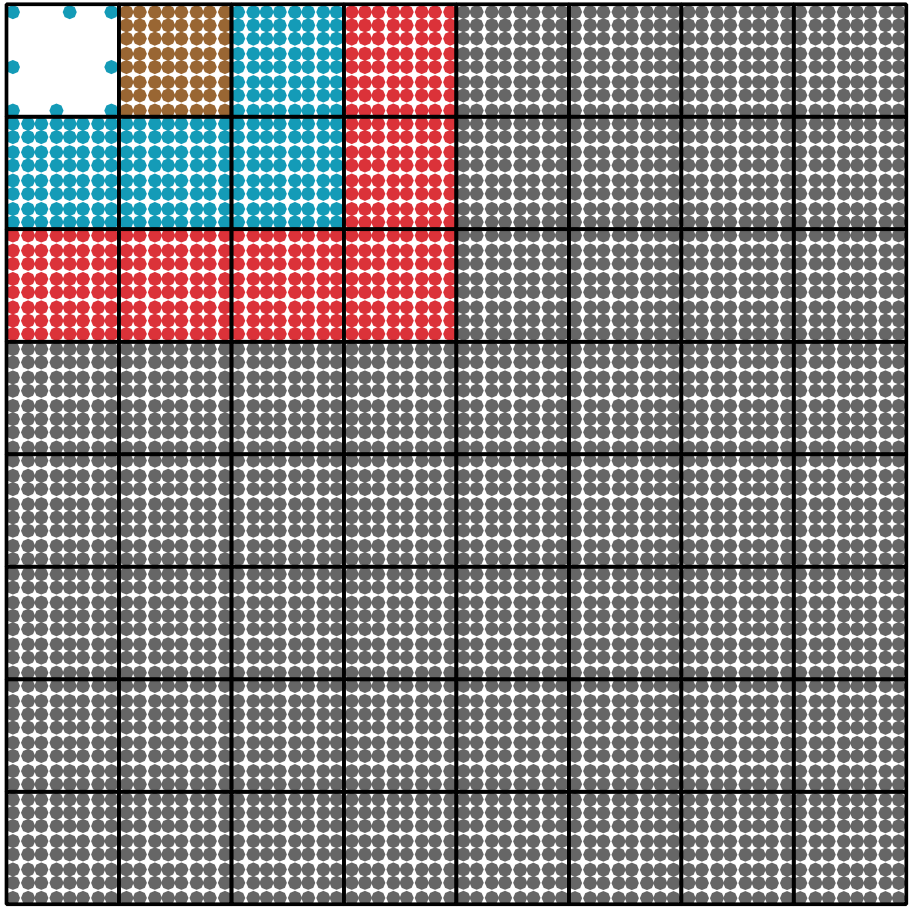
Domain:



Nested dissection-like

- Box 2 (before)

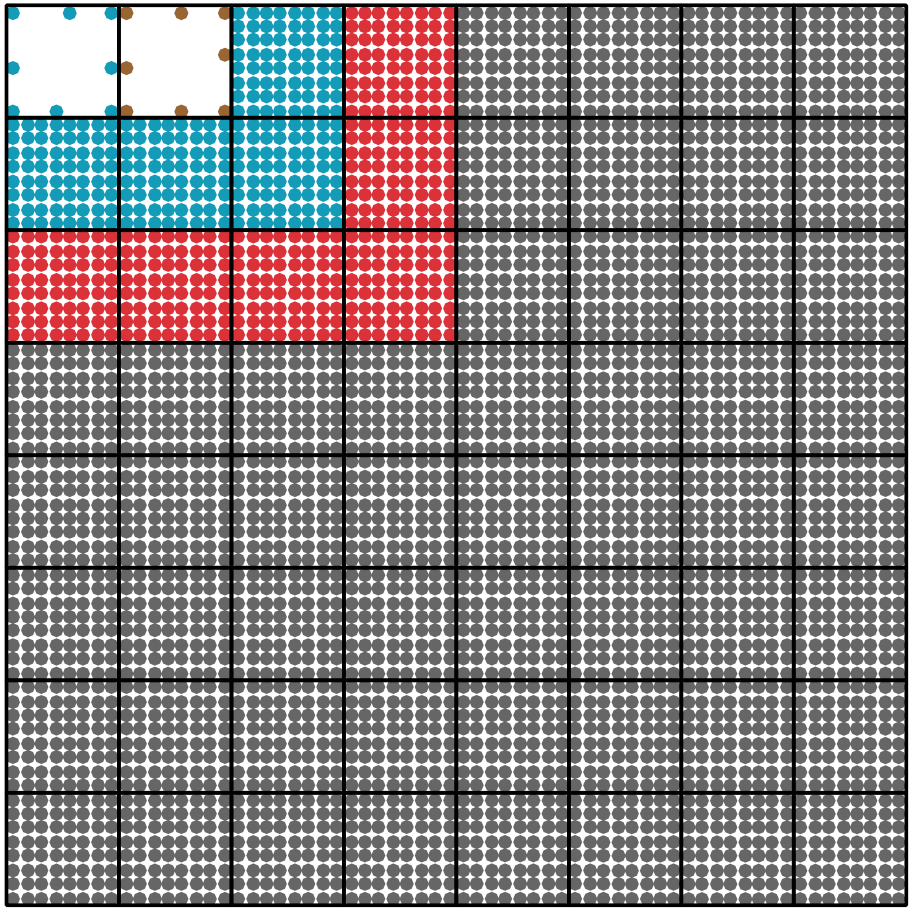
Domain:



Nested dissection-like

- Box 2 (after)

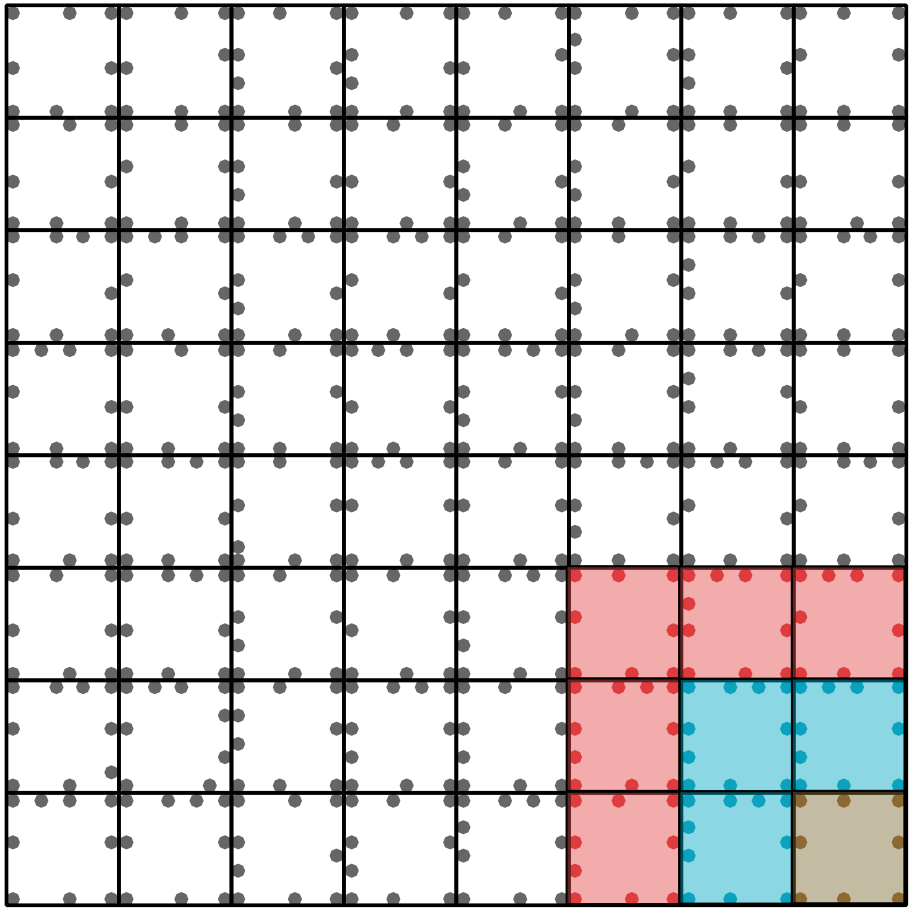
Domain:



Nested dissection-like

- …Box 64 (after)

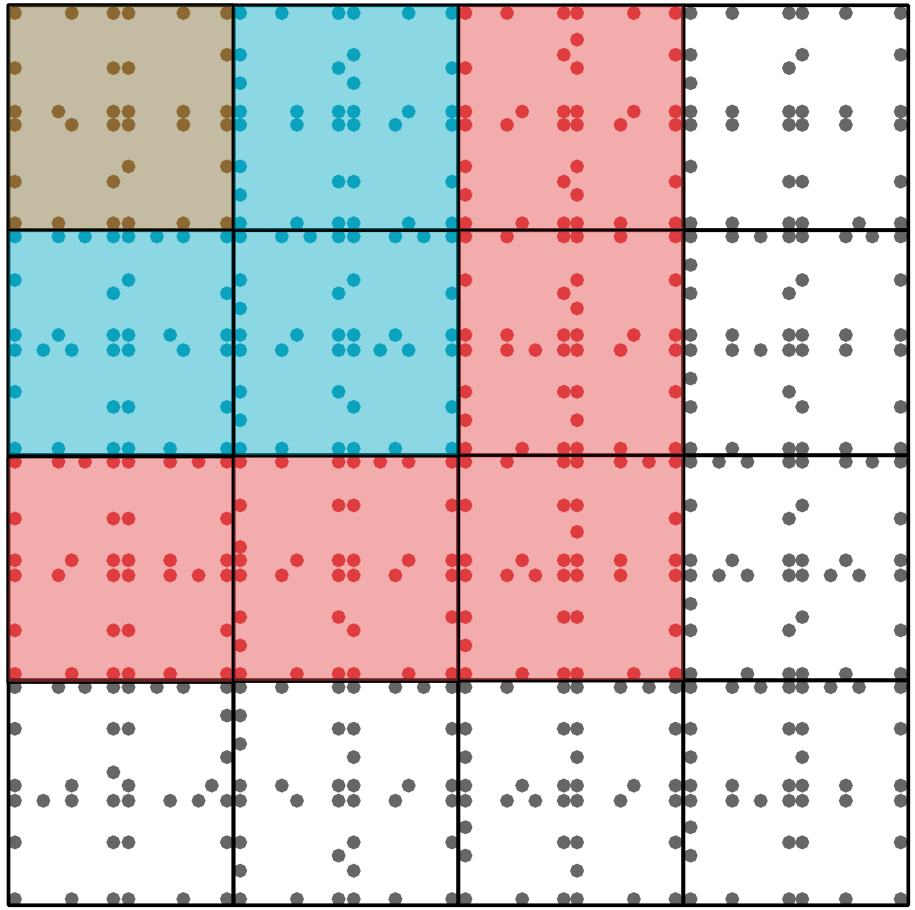
Domain:



Nested dissection-like

- Kicker: hierarchical!
- Box 1 (before)

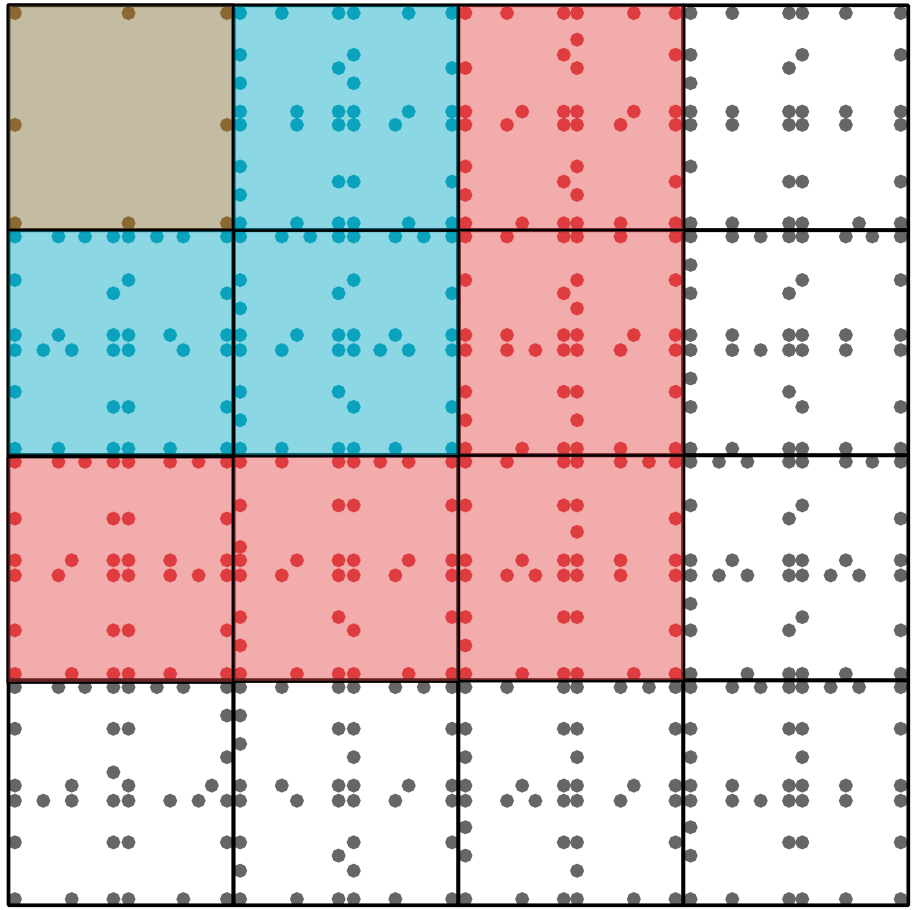
Domain:



Nested dissection-like

- Kicker: hierarchical!
- Box 1 (after)

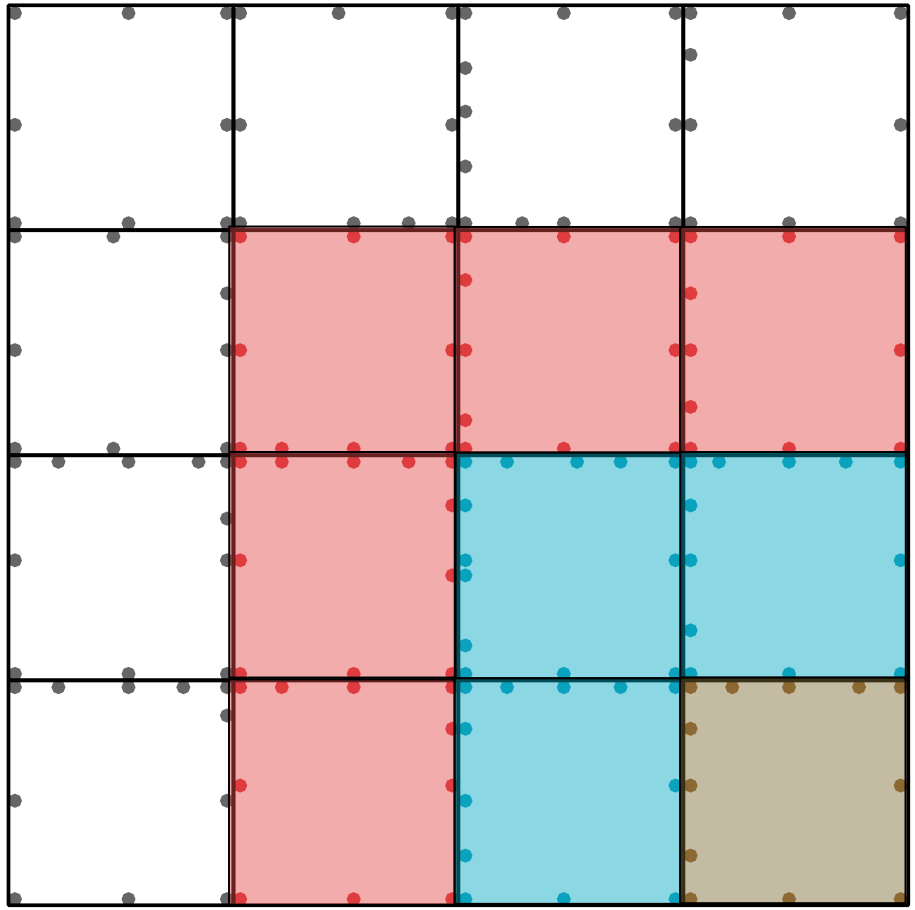
Domain:



Nested dissection-like

- Kicker: hierarchical!
- ... end

Domain:



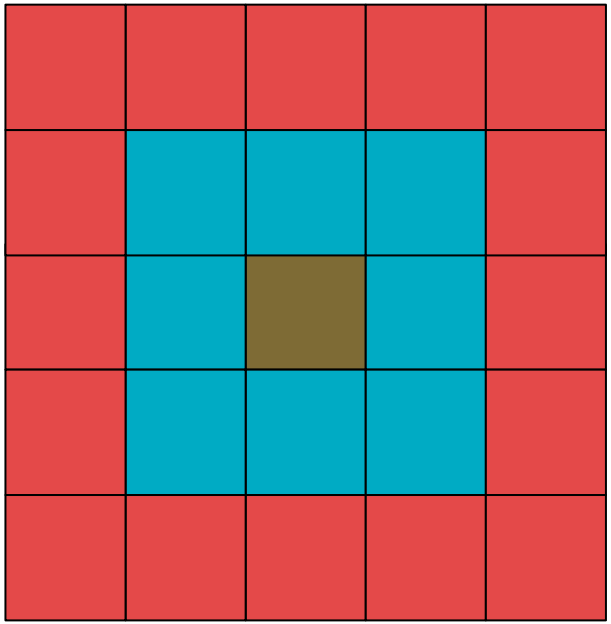
Complexity sketch

- For each box from smallest to biggest:
  1. Compress box-to-far
  2. Perform block elimination

- If proxy surface used and constant rank:

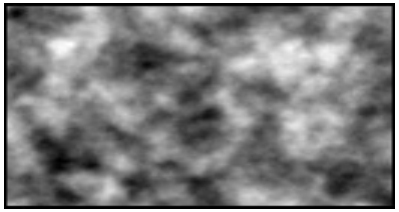
  $\mathcal{O}(N)$ (linear complexity)

- Important: final factorization gives log-det, sqrt, solve, etc. but not trace-of-product

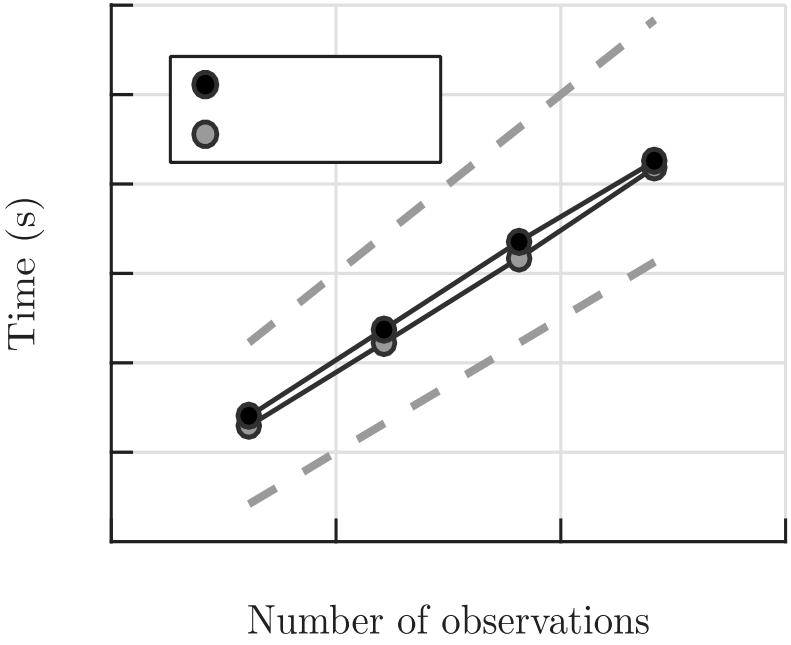


Example: synthetic data on grid $[0,100]^2$

GP:



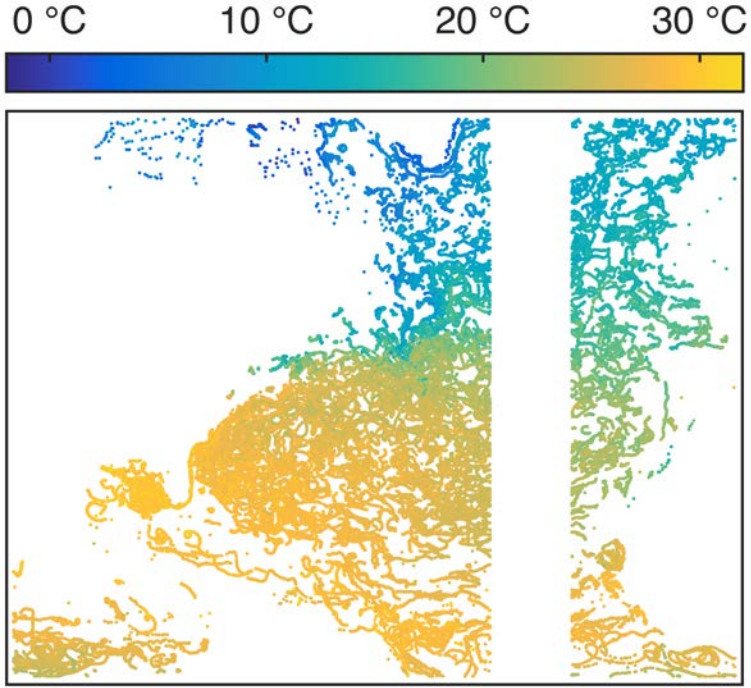
- This example uses a simpler factorization algorithm due to [Martinsson & Rokhlin, 2005].
- Runtime for 1 iteration of quasi-Newton
  (1 objective and 1 gradient evaluation)
- About 5 hours / iteration for 262,144 points

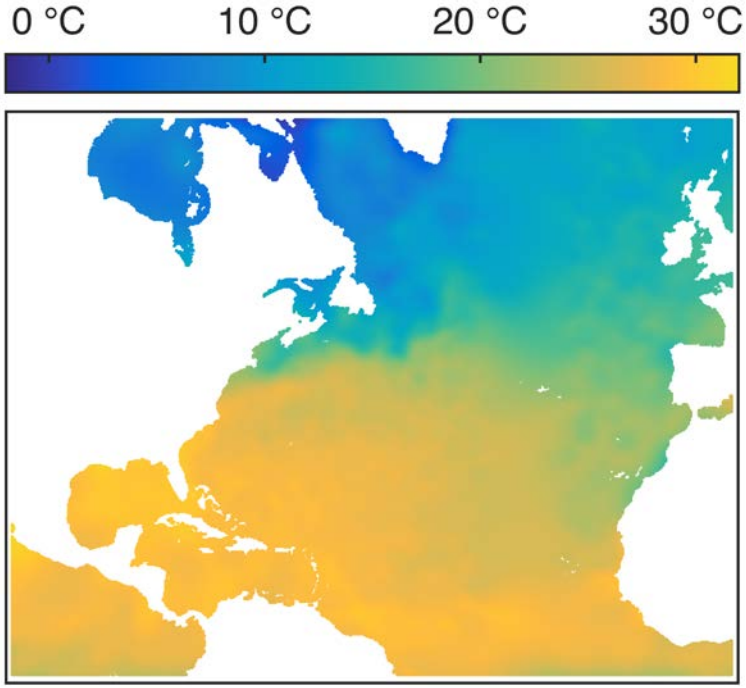


(MATLAB R2015a on a quad-socket Intel Xeon E5-4640 processor @ 2.4 GHz)

Example: ICOADS ocean data

National Climatic Data Center/NESDIS/NOAA/U.S. Department of Commerce, Data Support Section/Computational and Information Systems Laboratory/National Center for Atmospheric Research/University Corporation for Atmospheric Research, Earth System Research Laboratory/NOAA/U.S. Department of Commerce, and Cooperative Institute for Research in Environmental Sciences/University of Colorado. 1984, updated monthly. *International Comprehensive Ocean-Atmosphere Data Set (ICOADS) Release 2.5, Monthly Summaries*. Research Data Archive at the National Center for Atmospheric Research, Computational and Information Systems Laboratory. http://dx.doi.org/10.5065/D6CF9N3F. Accessed 11 Nov 2015.





Final thoughts

Take-away: Matrices in practice are frequently “easy”, spatial Gaussian processes are but one example.

- If you have rank-structure, exploit it!

Applications and extensions

- Maximum-likelihood estimation for Gaussian processes
- Preconditioners and direct solvers for low-frequency scattering and potential problems
- Updating factorizations after geometry changes for optimization
- Interpolation with radial basis functions
- Parallel C++ implementation (joint with Yingzhou Li)

Acknowledgments

- Thanks to the Department of Energy Computational Science Graduate Fellowship for supporting my research and conference travel.
- Thanks to all at the Krell Institute for all the help and for fostering a great community.

Thanks for listening!

Questions?

1. Why can't I do this for 1000-dimensional Gaussian processes?
2. What are the challenges for parallelizing these approaches?
3. But Victor, these are just pictures! What are the nitty-gritty details?
4. How do you compute the trace of the product?
5. Can I use your code?
   - https://github.com/klho/FLAM [Ken L. Ho]