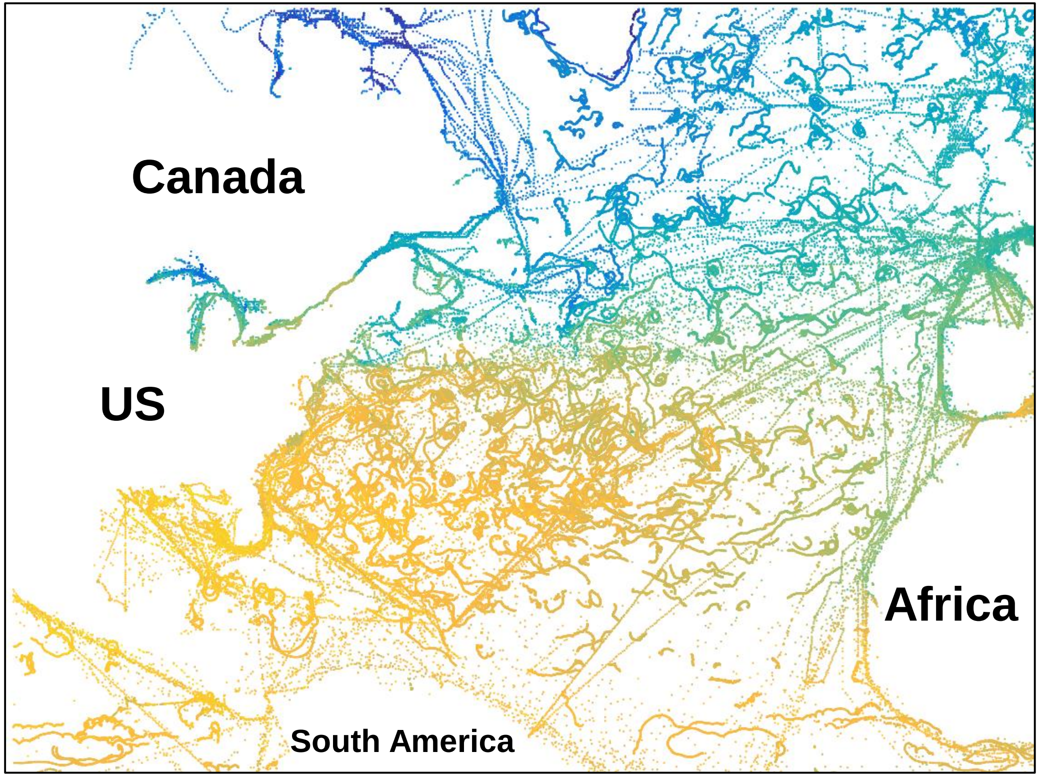
Fast spatial Gaussian process maximum likelihood estimation via skeletonization factorizations

Victor Minden*, Anil Damle, Ken L. Ho, Lexing Ying
*ICME, Stanford University,
2017 SIAM Annual Meeting

Real data: sea surface temperature (source: ICOADS)

- What is a good model for spatial data?
- Tobler's first law of geography
  - "Everything is related to everything else, but near things are more related than distant things"



The Gaussian process model (in 2D)

- Field is indexed by space, $\mathcal{Z}: \mathbb{R}^2 \to \mathbb{R}$
- Finite-dimensional distributions are multivariate normal

$$(z_1, \ldots, z_N)^T \sim N(\mu, \Sigma), \text{ with } \Sigma_{ij} = \sigma^2 k(x_i, x_j; \theta)$$

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

Applications of "kriging"

- Mining
- Hydrogeology
- Environmental science
- Natural resources

[Krige, 1951] [Zimmerman et al., 1998] [Bayraktar, 2005] [Goovaerts, 1997]

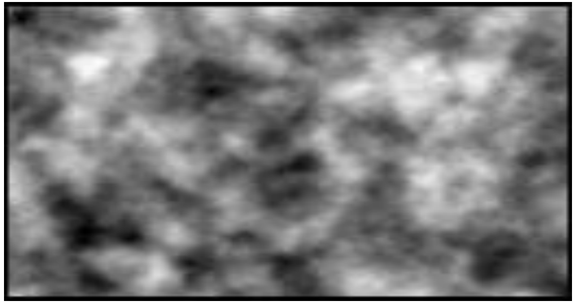
Parameters in GPs

Choice of kernel function is important to characterize field

- Squared-exponential: $k(x, y; \theta) = \exp(-|x-y|_\theta^2)$
- Matérn kernel (one such): $k(x, y; \theta) = \left(1 + \sqrt{3}|x-y|_\theta\right) \exp(-\sqrt{3}|x-y|_\theta)$

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 for GPs (1)

So, assume GP model makes sense [Stein, 1999] and:

- Kernel $k(x, y; \theta)$ is specified up to θ (so Σ depends on θ)
- Have observation vector z$= [z_1, \dots, z_N]^T$ with locations $\{x_i\} \subset \mathbb{R}^2$

- GP log-likelihood with $\Sigma = \sigma^2 K(\theta)$ (up to constants)

$$\ell\left(\theta, \mu, \sigma^2\right) = -(\mathbf{z} - \boldsymbol{\mu}\mathbf{1})^{\mathbf{T}} \boldsymbol{\Sigma}^{-\mathbf{1}} (\boldsymbol{z} - \boldsymbol{\mu}\mathbf{1}) - \mathbf{log}|\boldsymbol{\Sigma}|$$

- Optimize analytically over μ and σ^2, leading to the (log) profile likelihood (up to other constants)

$$\ell_p(\theta) = \ell\left(\theta, \hat{\mu}(\theta), \widehat{\sigma^2}(\theta)\right) = -\mathbf{log}|\mathbf{K}| - \boldsymbol{N}\,\mathbf{log}\left(\boldsymbol{z}^{\boldsymbol{T}}\left(\boldsymbol{K} + \mathbf{1}\mathbf{1}^{\boldsymbol{T}}\right)^{-\mathbf{1}} \boldsymbol{z}\right)$$

Maximum likelihood estimation for GPs (2)

Final goal: efficient method for finding maximum likelihood estimate where (up to an affine shift)

$$\ell_p(\theta) = -\log|\mathrm{K}| - N \log\left(z^T\left(K + 11^T\right)^{-1} z\right)$$

$$\frac{\partial \ell_p(\theta)}{\partial \theta_i} = -\mathrm{Tr}\left(K^{-1}\frac{\partial K}{\partial \theta_i}\right) + N\left(\frac{z^T\left(K+11^T\right)^{-1}\frac{\partial K}{\partial \theta_i}\left(K+11^T\right)^{-1} z}{z^T\left(K+11^T\right)^{-1} z}\right)$$

1. Log-determinant
2. Apply / solve with K and derivative
3. Product-trace

Efficient with skeletonization factorizations

Our approach: black-box MLE

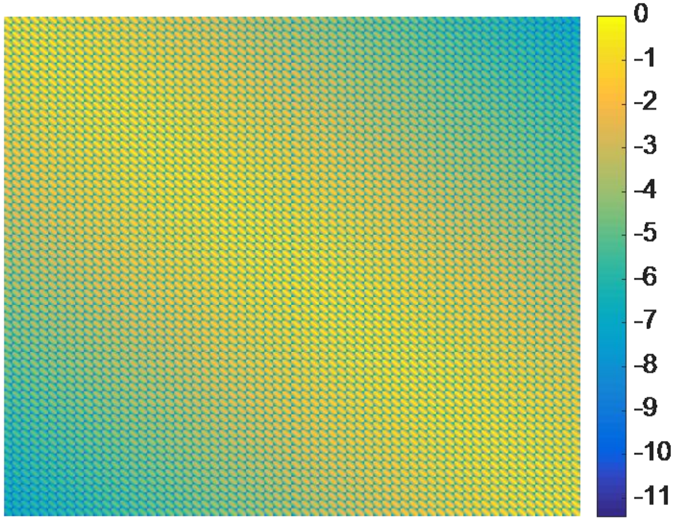
- We outline a simple efficient scheme for computing:
  - the log profile likelihood (including $\log|K|$)
  - its gradient (including product-trace)

- Then, use black-box optimization scheme (e.g., fminunc/fmincon)

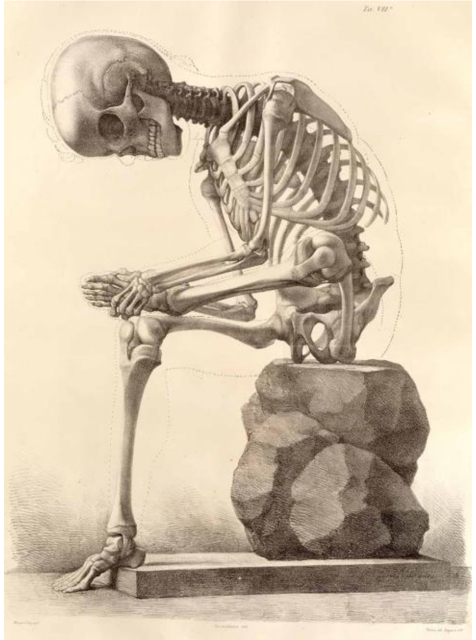
- A few other approaches:
  - Sample average approximation
  - (Block) composite likelihood
  - Approximate by Gaussian Markov random field
  - Covariance tapering
  - Multi-level preconditioning + tapering

  [Anitescu et al., 2012] [Eidsvik et al., 2014] [Vecchia, 1988] [Lindgren et al., 2011] [Furrer et al., 2006] [Castrillon-Candas et al., 2015]

- Previous work on hierarchical decompositions for Gaussian processes but no gradients
  [Ambikasaran et al., 2016] [Ambikasaran et al., ArXiV]
  [Borme & Garcke, 2007] [Khoromskij et al., 2008]



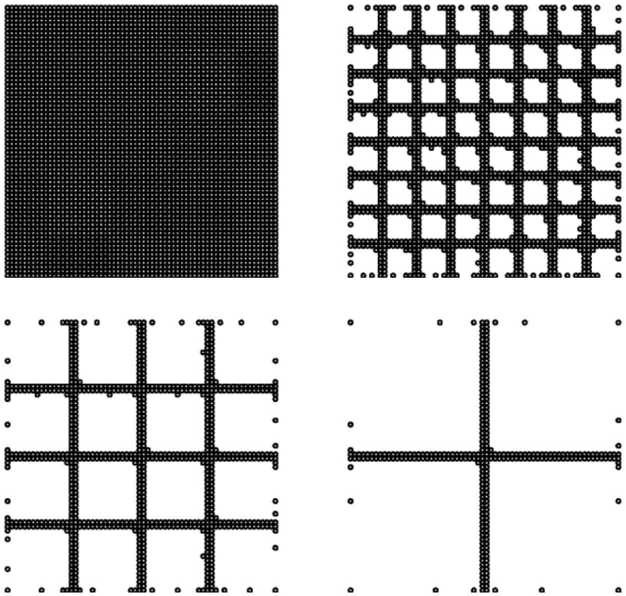
(Part 1) Skeletonization factorizations



Hierarchical representations from skeletonization

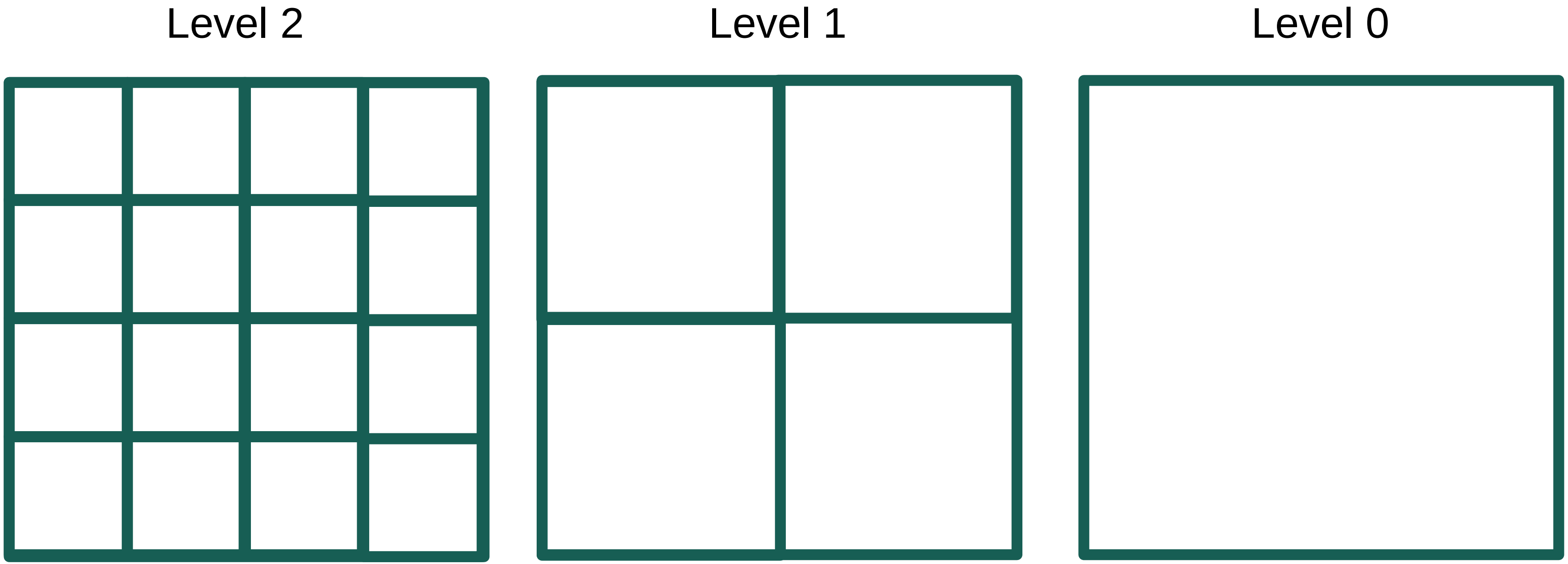
- “Recursive skeletonization” and related literature has led to many nice hierarchical representations based on “weak admissibility”

- HSS / HBS matrices
  - [Martinsson & Rokhlin, 2005]
  - [Chandrasekaran et al., 2006 & 2007]
  - [Ho & Greengard, 2012]
  - [Xia et al., 2012]
  - [Gillman et al., 2012]
- HODLR matrices
  - [Martinsson, 2008]
  - [Ambikasaran & Darve, 2013]

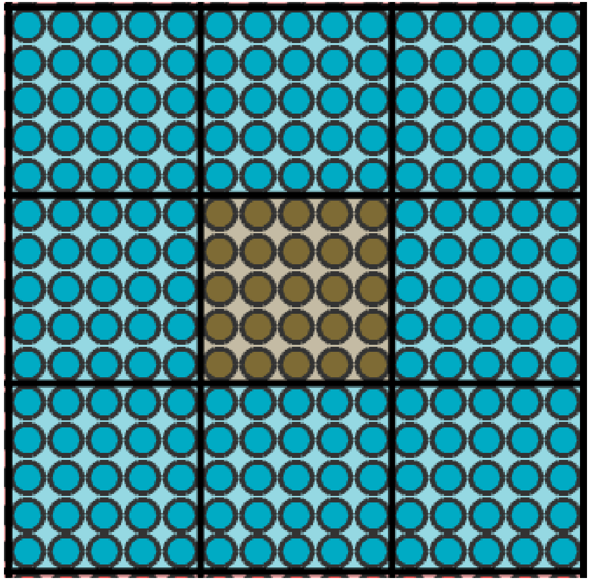


Use a tree decomposition of space

- For visualization, assume point distribution is uniform in a 2D square



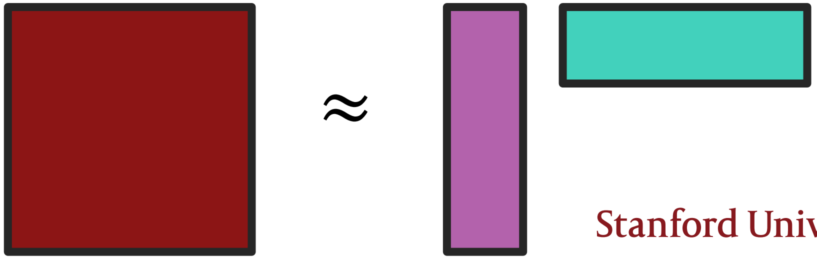
Bottom level of quadtree



- **Brown**: box b
- **Blue**: complement of b
  (lots of these)

$$\left[\begin{array}{c|c} K_{bb} & K_{bc} \\ \hline K_{cb} & K_{cc} \end{array}\right]$$

$$K_{bc} \approx UV^T$$



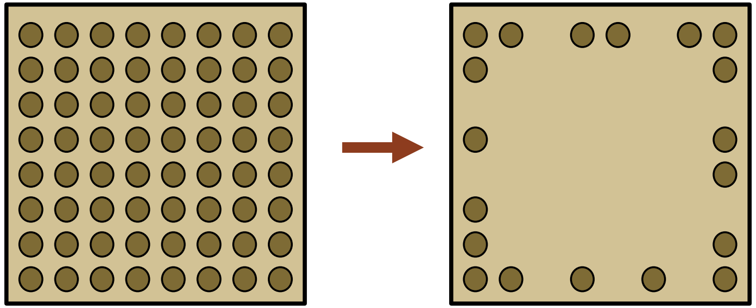
Interpolative decomposition

- By assumption, the interactions between **box b** and **"the rest of the world"** are low-rank.

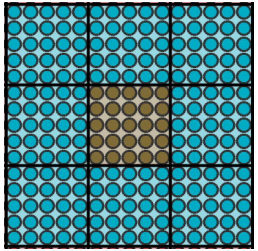
- Compress these with an *interpolative decomposition* [Cheng et al., 2005], where box **b** is partitioned into small **"skeleton set" s** and larger **"redundant set" r**

$$\left[\begin{array}{c|c} K_{bb} & K_{bc} \\ \hline K_{cb} & K_{cc} \end{array}\right]$$

$$\boxed{\begin{aligned} b &= s \cup r \\ K_{cr} &\approx K_{cs} T \end{aligned}}$$



(1) Break box into skeleton and redundant

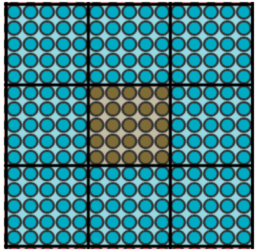


$$\left[\begin{array}{c|c} K_{bb} & K_{bc} \\ \hline K_{cb} & K_{cc} \end{array}\right] = \left[\begin{array}{cc|c} K_{rr} & K_{rs} & K_{rc} \\ K_{sr} & K_{ss} & K_{sc} \\ \hline K_{cr} & K_{cs} & K_{cc} \end{array}\right]$$

$$b = s \cup r$$

$$K_{cr} \approx K_{cs} T$$

(2) Insert low-rank approximation

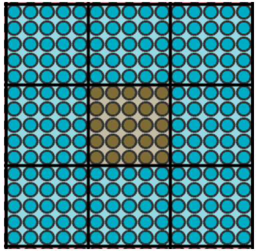


$$\left[\begin{array}{c|c} K_{bb} & K_{bc} \\ \hline K_{cb} & K_{cc}\end{array}\right] \approx \left[\begin{array}{cc|c} K_{rr} & K_{rs} & T^T K_{sc} \\ K_{sr} & K_{ss} & K_{sc} \\ \hline K_{cs}T & K_{cs} & K_{cc}\end{array}\right]$$

$$b = s \cup r$$

$$K_{cr} \approx K_{cs} T$$

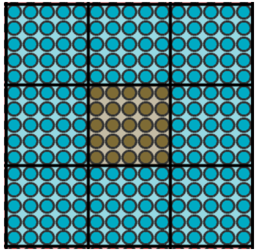
(3) Block sparse elimination



$$\left[\begin{array}{cc|c} K_{rr} & K_{rs} & T^T K_{sc} \\ K_{sr} & K_{ss} & K_{sc} \\ \hline K_{cs}T & K_{cs} & K_{cc} \end{array}\right] \longrightarrow \left[\begin{array}{cc|c} X_{rr} & X_{rs} & \\ X_{sr} & K_{ss} & K_{sc} \\ \hline & K_{cs} & K_{cc} \end{array}\right]$$

- Subtract multiple of second column from first column (and second row from first row)

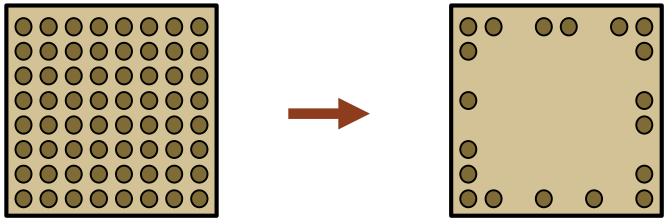
(3) Block sparse elimination



$$\left[\begin{array}{cc|c} X_{rr} & X_{rs} & \\ X_{sr} & K_{ss} & K_{sc} \\ \hline & K_{cs} & K_{cc}\end{array}\right] \longrightarrow \left[\begin{array}{cc|c} X_{rr} & & \\ & X_{ss} & K_{sc} \\ \hline & K_{cs} & K_{cc}\end{array}\right]$$

- Subtract multiple of first column from second column (and first row from second row)

Result: redundant DOFs are decoupled

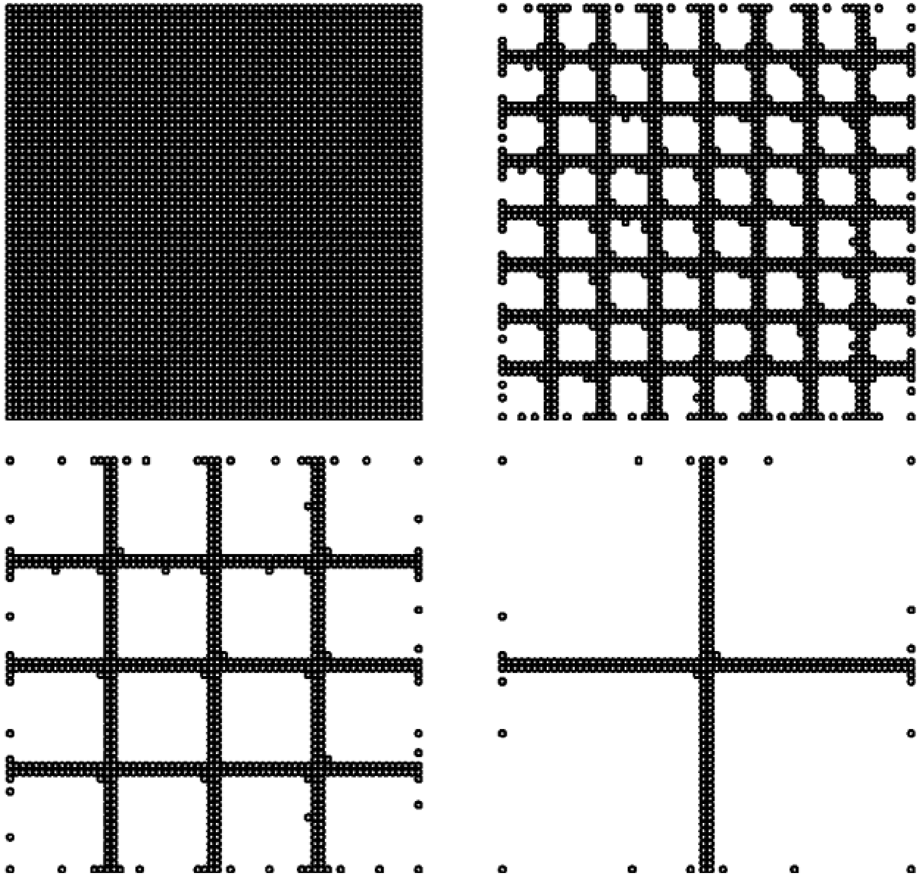


$$K = \left[\begin{array}{cc|c} K_{rr} & K_{rs} & K_{rc} \\ K_{sr} & K_{ss} & K_{sc} \\ \hline K_{cr} & K_{cs} & K_{cc} \end{array}\right]$$

Unit determinant

$$\approx \left[\begin{array}{cc|c} V_{rr} & V_{rs} & \\ V_{sr} & V_{ss} & \\ \hline & & I \end{array}\right] \left[\begin{array}{cc|c} X_{rr} & & \\ & X_{ss} & K_{sc} \\ \hline & K_{cs} & K_{cc} \end{array}\right] \left[\begin{array}{cc|c} V_{rr} & V_{rs} & \\ V_{sr} & V_{ss} & \\ \hline & & I \end{array}\right]^T$$

Repeat for each box at each level of the hierarchy



- For each box from bottom to top, eliminate redundant DOFs interior to that box.
- At each level, ignore redundant DOFs from the previous level (they are decoupled already)

What does a factorization look like?

- For each level, we have a permutation matrix and a block diagonal matrix where each block has unit determinant. In the middle, we have all the diagonal blocks corresponding to decoupled redundant DOFs

$$K \approx \left(\prod_{\ell \in [L]} P_\ell V_\ell \right) P_0 D P_0^T \left(\prod_{\ell \in [L]} P_\ell V_\ell \right)^T \equiv F$$

- We get a factorization of the inverse operator in the same form

- This is the *recursive skeletonization factorization* [Ho & Ying, 2015], a multiplicative form of recursive skeletonization [Martinsson & Rokhlin, 2005]

Complexity note

- If the number of skeletons per box stays $O(\log N)$ as we go up the tree, then we can efficiently (for a specified tolerance)
    1. form F and F^{-1},
    2. apply F, F^{-1}, C, or C^{-1}, and
    3. compute $\log|F|$

  in time that is linear in N (up to a small polylogarithmic factor)

- If skeleton size grows for recursive skeletonization factorization, use the related *hierarchical interpolative factorization* [Ho & Ying, 2015]

$$K \approx \left(\prod_{\ell \in [L]} P_\ell V_\ell\right) P_0 D P_0^T \left(\prod_{\ell \in [L]} P_\ell V_\ell\right)^T \equiv F$$

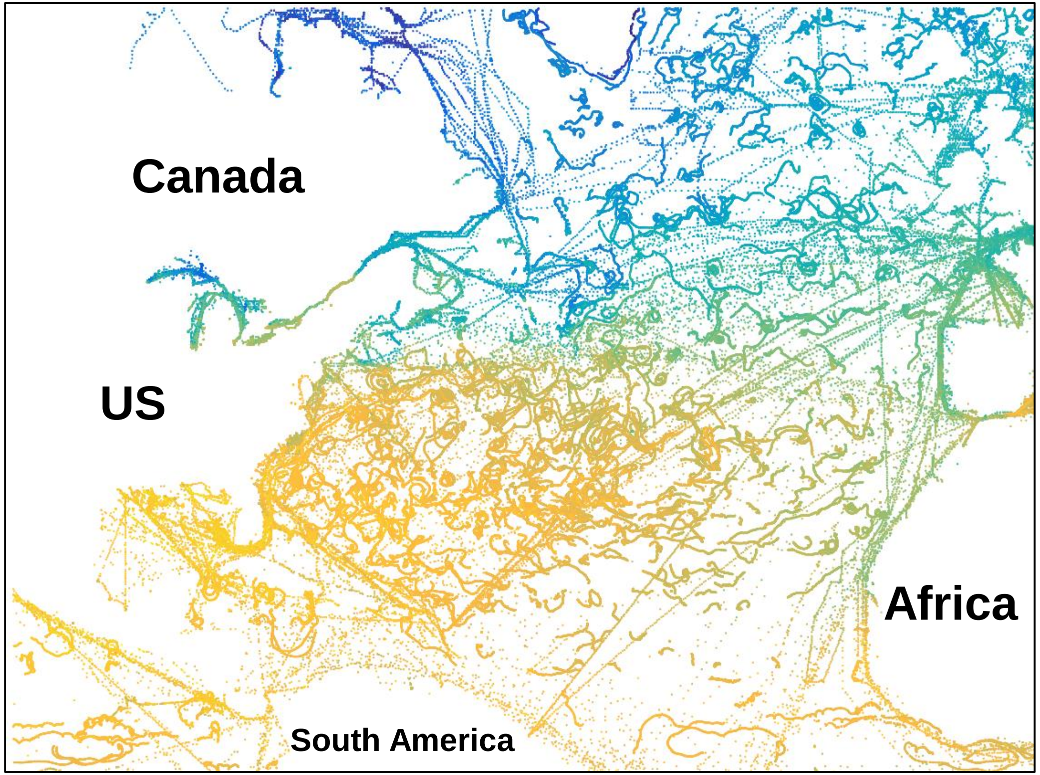
$$\log|K| \approx \log|F| = \log|D|$$

$$F = CC^T$$

Numerical results (1)

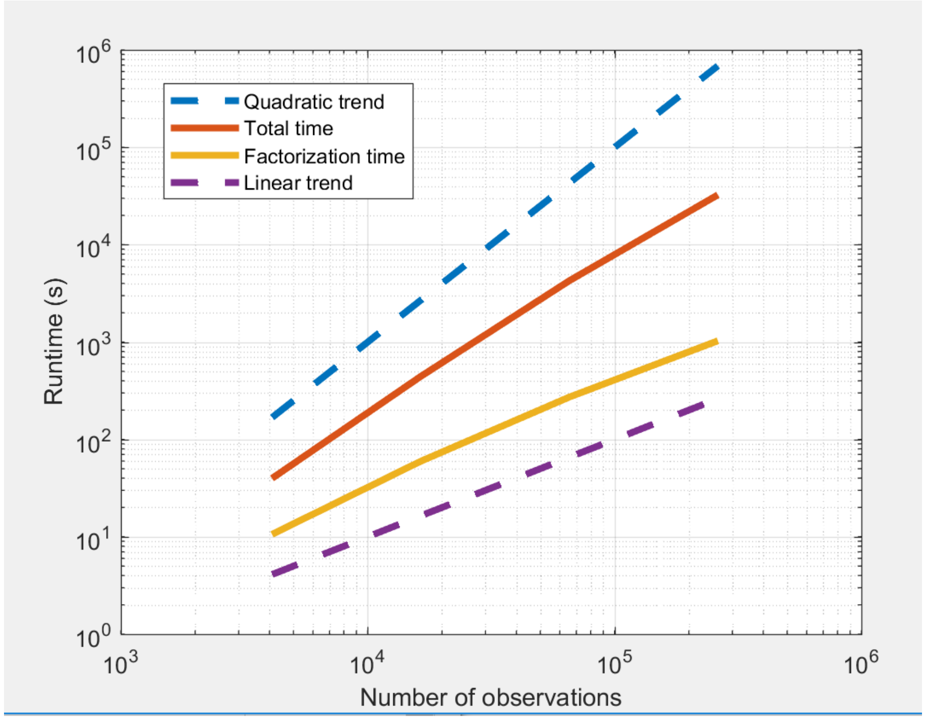
- Using the recursive skeletonization factorization, how long does it take to compute the GP log-likelihood to a given tolerance?

- Data locations: scattered ICOADS data mapped inside [0,100] x [0,100]



Numerical results (1)

- $k(x,y;\theta) = \left(1+\sqrt{3}|x-y|_\theta\right)\exp\left(-\sqrt{3}|x-y|_\theta\right) + 0.0001\delta(x-y)$



- Evaluated at length scale θ=[7,10] with tolerance $\epsilon = 10^{-11}$
- Factorization is efficient, but total time to compute log-likelihood and gradient is dominated by $O(N^2)$ product-trace

(Part 2) Computing the product-trace

$$\frac{\partial \ell_p(\theta)}{\partial \theta_i} = -\mathrm{Tr}\left(K^{-1}\frac{\partial K}{\partial \theta_i}\right)$$

$$+N\left(\frac{z^T\left(K+11^T\right)^{-1}\frac{\partial K}{\partial \theta_i}\left(K+11^T\right)^{-1}z}{z^T(K+11^T)^{-1}z}\right)$$

Selected sparse algebra using locality

$$F \equiv \left(\prod_{\ell \in [L]} P_\ell V_\ell\right) P_0 D P_0^T \left(\prod_{\ell \in [L]} P_\ell V_\ell\right)^T$$

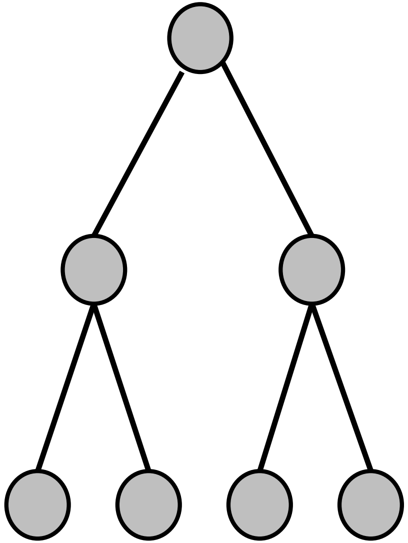
- Consider simpler case to start: drop product and look at trace of $F^{-1} \approx K^{-1}$

- Application of factorization *F* or its inverse to a vector is two-stage:

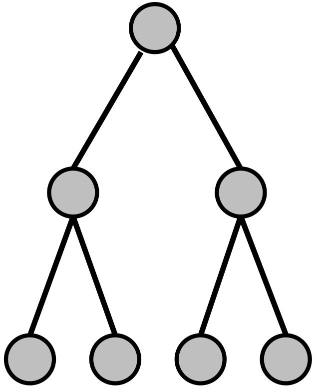
  1. Apply a diagonal block for each node from bottom-to-top

     (then apply a block diagonal operator)

  2. Apply a diagonal block for each node from top-to-bottom



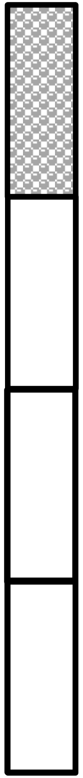
With sparse input, skip some factors on the way up

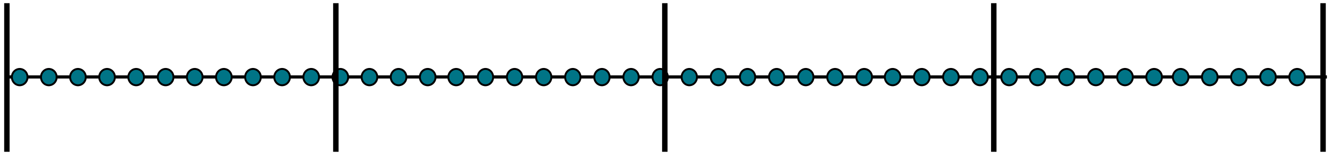


Left: tree

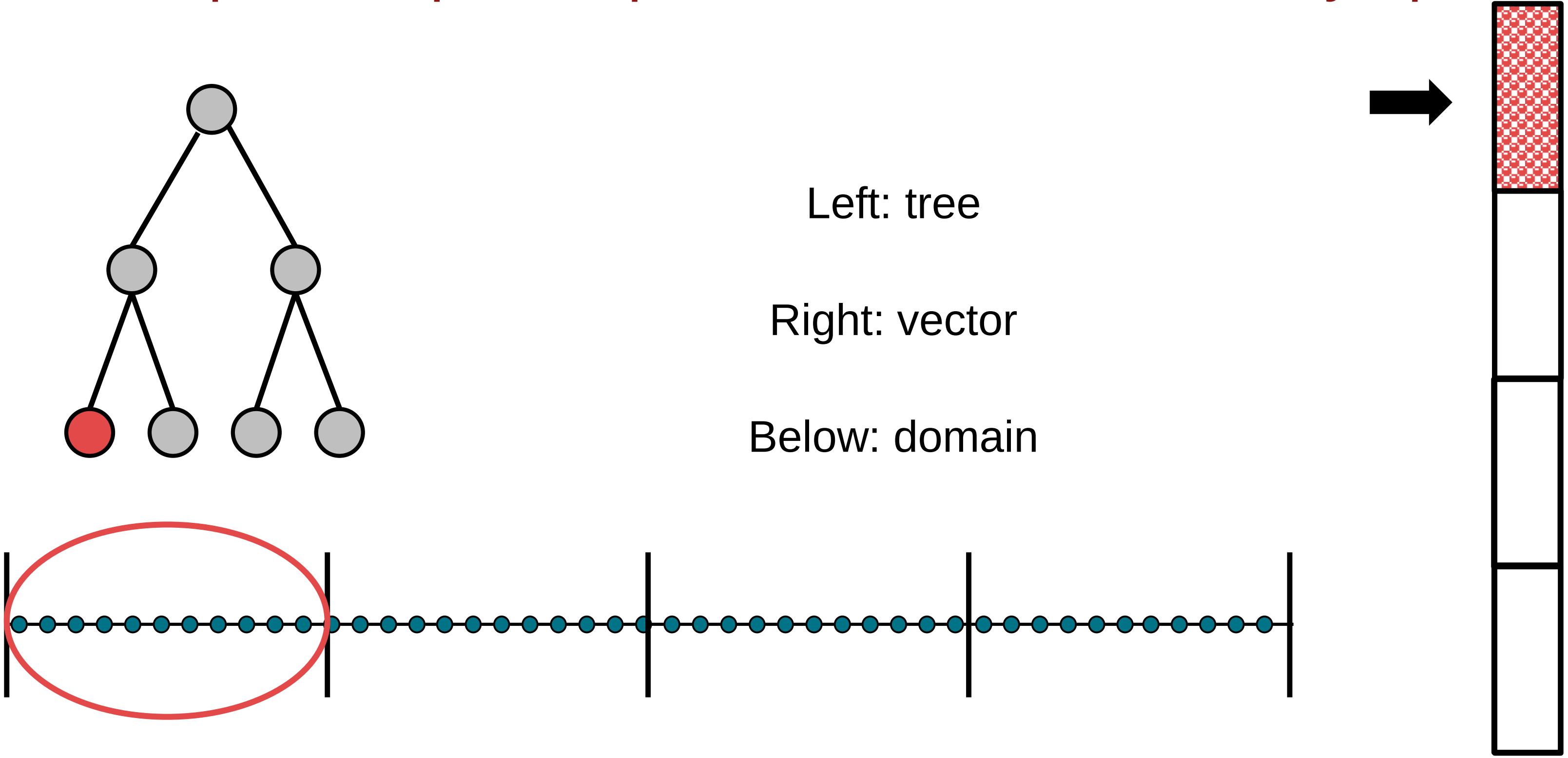
Right: vector

Below: domain





With sparse input, skip some factors on the way up

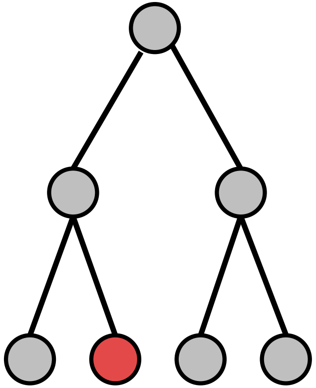


Left: tree

Right: vector

Below: domain

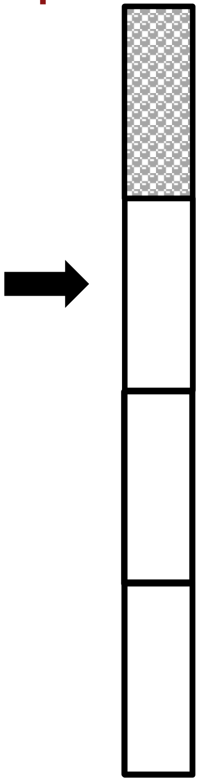
With sparse input, skip some factors on the way up

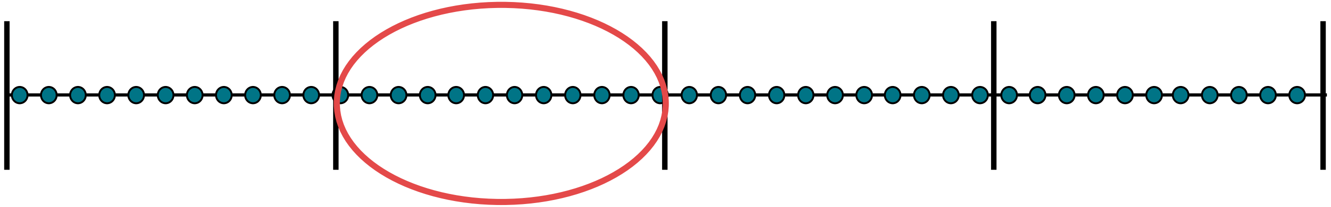


Left: tree

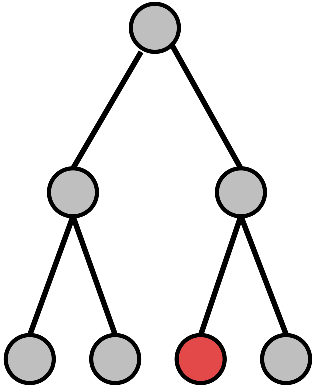
Right: vector

Below: domain





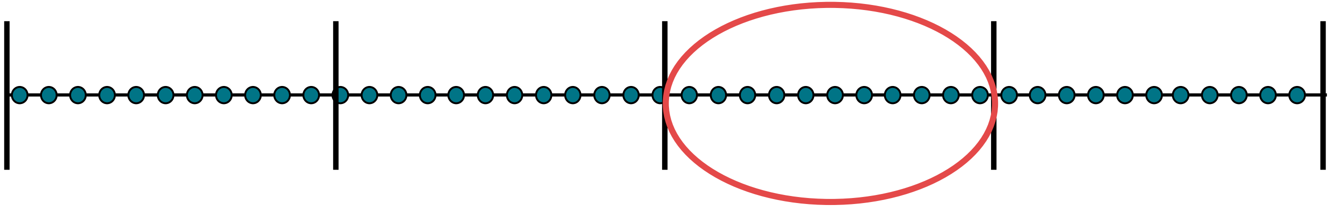
With sparse input, skip some factors on the way up

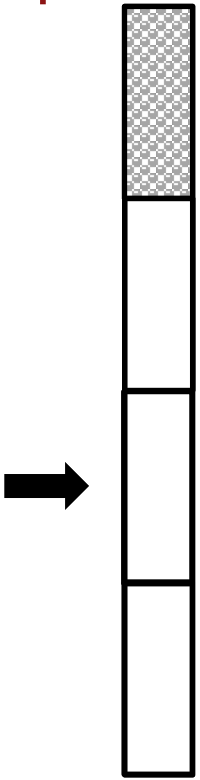


Left: tree

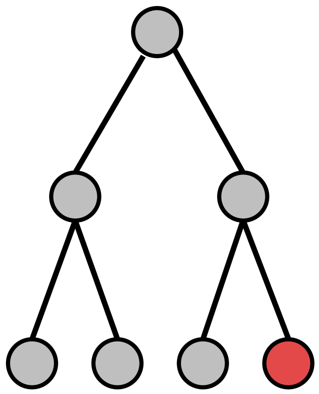
Right: vector

Below: domain





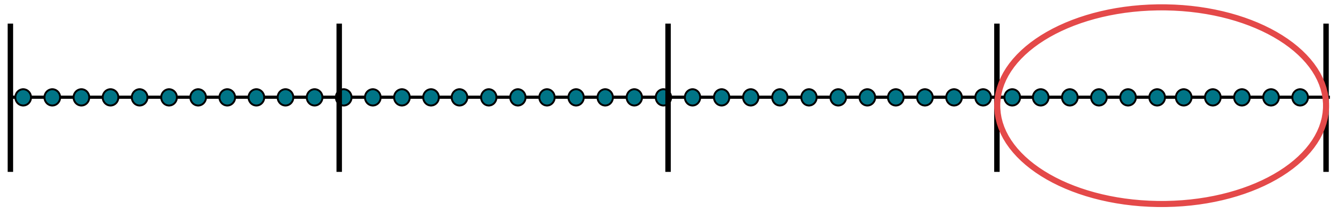
With sparse input, skip some factors on the way up

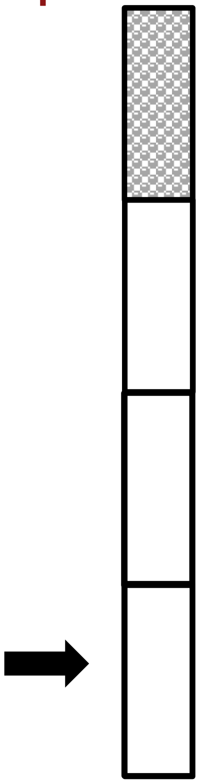


Left: tree

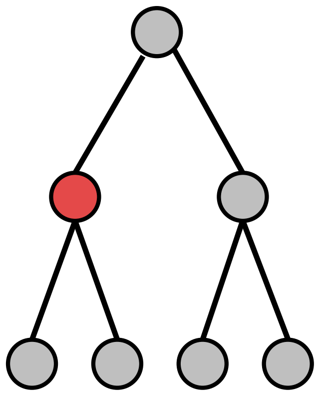
Right: vector

Below: domain





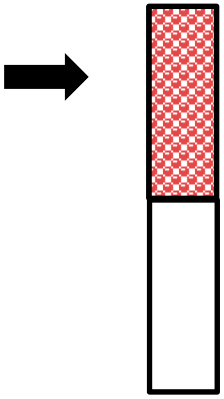
With sparse input, skip some factors on the way up

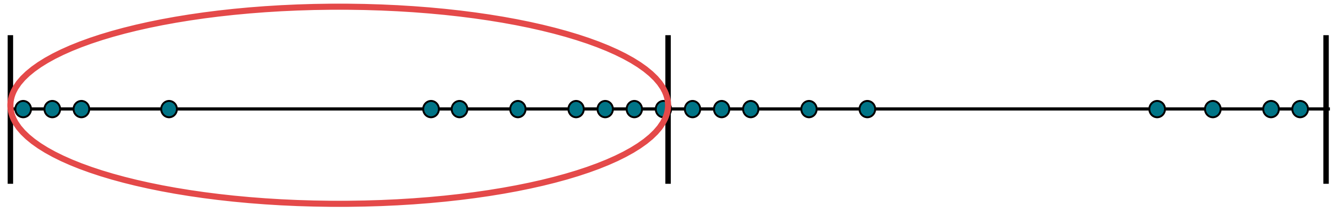


Left: tree

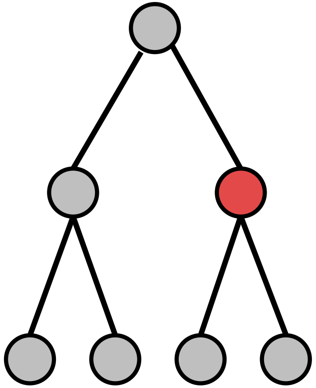
Right: vector

Below: domain





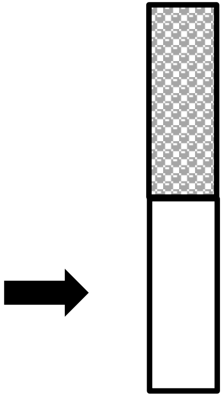
With sparse input, skip some factors on the way up

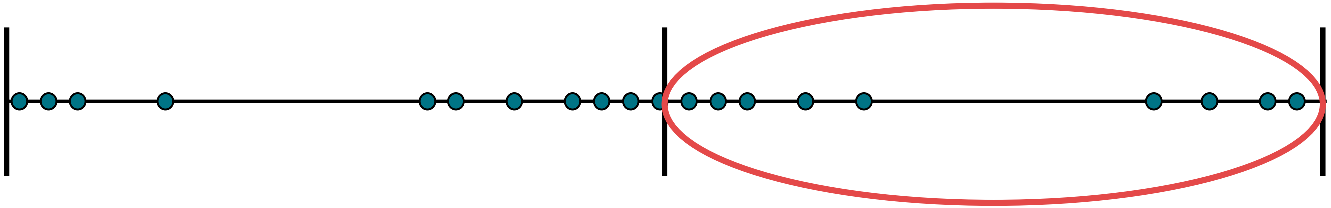


Left: tree

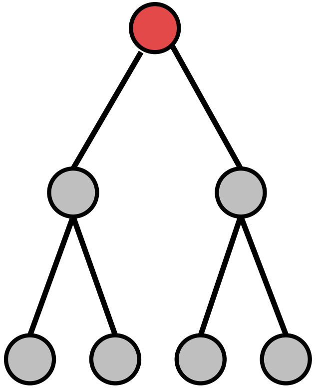
Right: vector

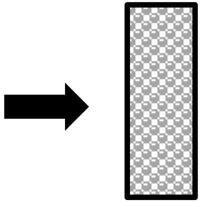
Below: domain





With sparse input, skip some factors on the way up

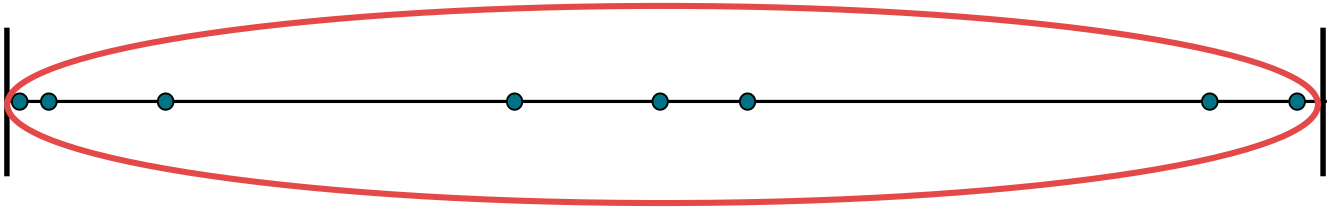




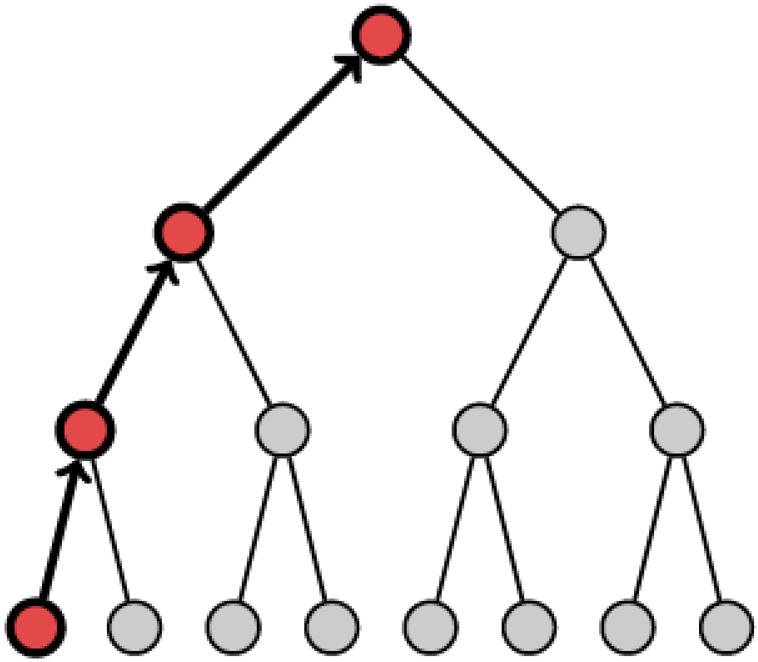
Left: tree

Right: vector

Below: domain



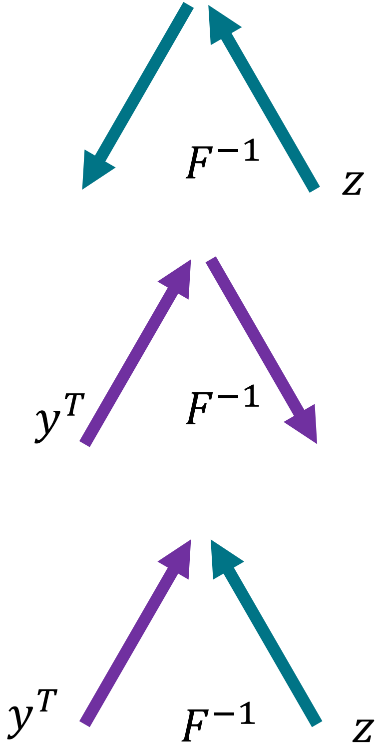
With sparse input, skip some factors on the way up



Result: easy to apply half of factorization to sparse vector

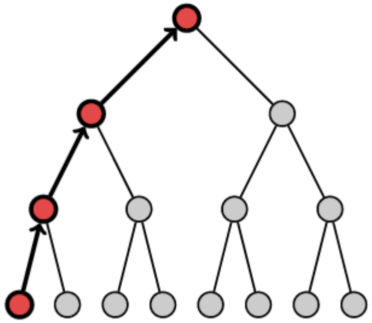
- Computing $F^{-1}z$ for a vector z is two-stage:
  1. Walk up tree
  2. Walk down tree
- Computing $y^T F^{-1}$ for a vector y is two-stage:
  1. Walk up tree
  2. Walk down tree
- Computing $y^T F^{-1} z$ for vectors y and z is two stage:
  1. Walk up tree (from right)
  2. Walk up tree (from left)

If sparse input and subselected output, only apply few factors!

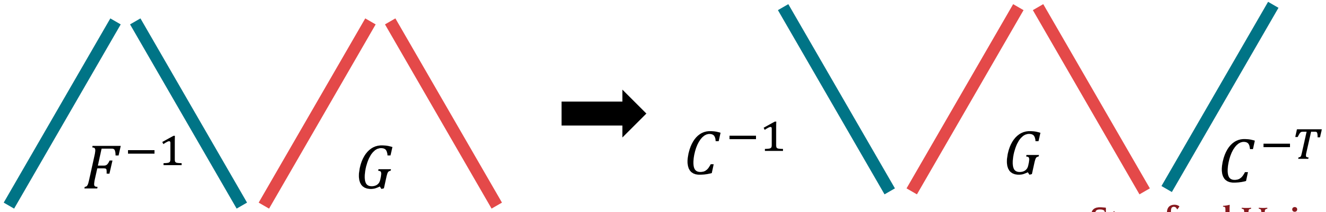


Selected sparse algebra (SSA) complexity

Operation	Complexity
Compute F_{ij}	$O(1)$
Compute F_{ij}^{-1}	$O(\log^3 N)$
Compute diag (F^{-1})	$O(N \log^3 N)$
Compute $\mathrm{Tr}(F^{-1}G)$	$O(N \log^3 N)$

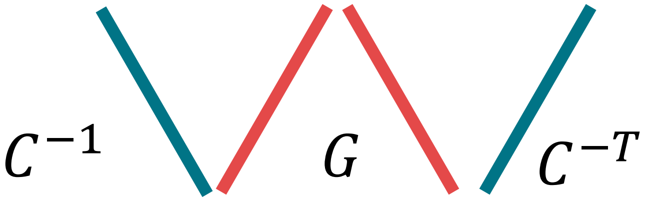


- Similar in spirit to SelInv [Lin et al., 2009] and FIND algorithm [Li et al., 2008] for sparse matrices.
- Does not give product trace directly, but similar idea using two trees

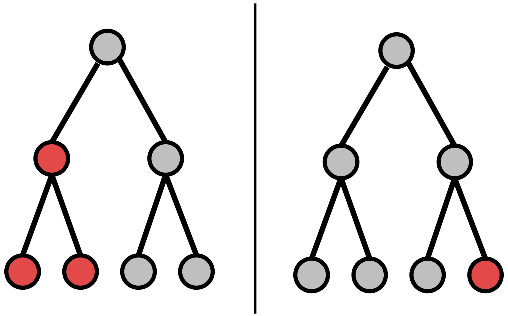


Efficient product-trace computation

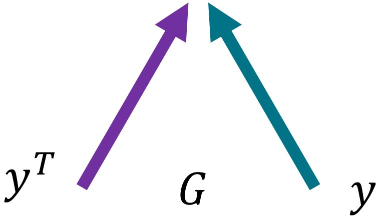
- To compute $\mathrm{Tr}(F^{-1}G) = \mathrm{Tr}(C^{-1}GC^{-T})$:



1. For many i (not all), $C^{-T}e_i$ is sparse (walk down tree)



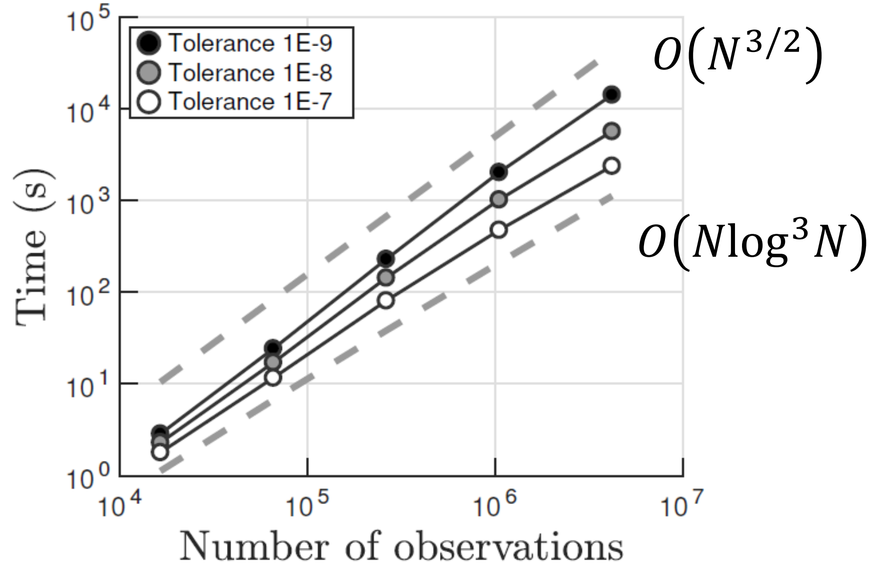
2. If y is sparse, then $y^T G y$ is fast (SSA from before)

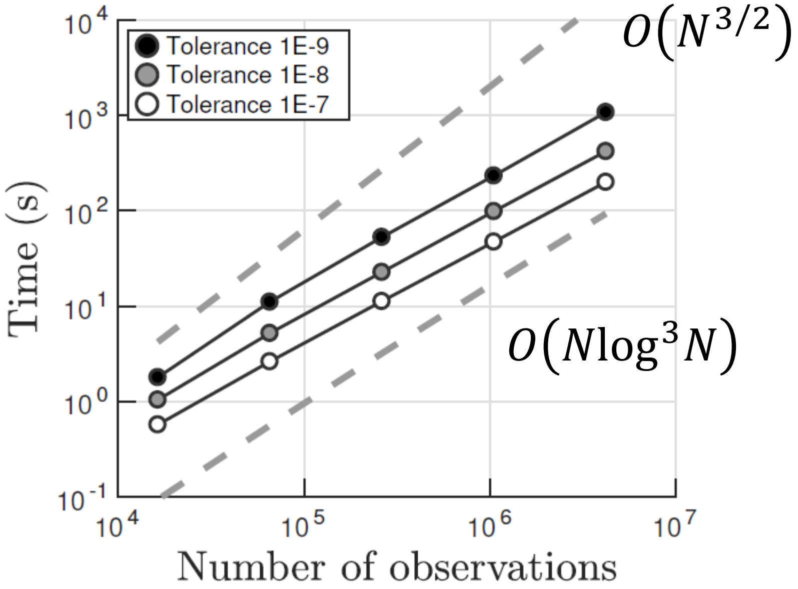


Results for SSA

$$k(x,y;\theta) = \left(1+\sqrt{3}|x-y|_\theta\right)\exp\left(-\sqrt{3}|x-y|_\theta\right)$$

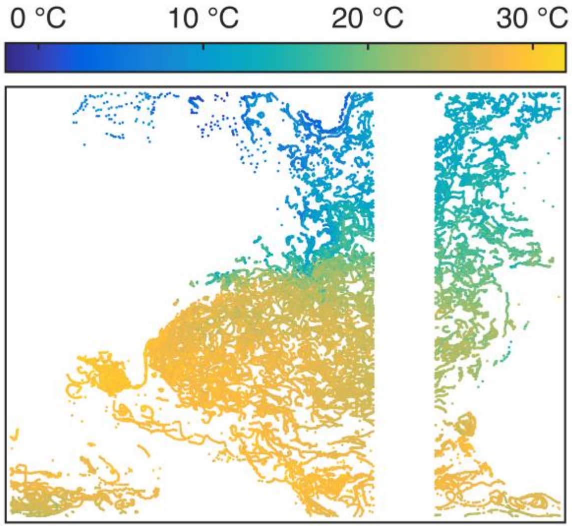
- Computing the product-trace using SSA
- Gridded observations in 2D with Matérn family kernel with length parameters [7,10] and [70,100].

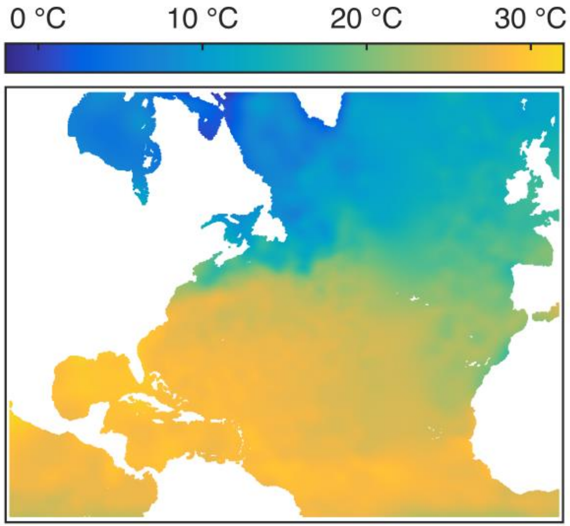




Gaussian process MLE with skeletonization

- Cost per iteration of black-box optimization: $O(pN\log^3 N)$
- Generated data: convergence in 5 to 7 quasi-Newton iterations
- Numerical differentiation stagnates: gradients are essential!





Comments on results

- Basic take-away
  - Hierarchical skeletonization-based factorizations are a natural choice for low-dimensional (spatial) Gaussian processes
  - Hardest (slowest) part is computing product trace for gradient

For $N = 512^2$

Time for SSA is 230 seconds = 7.1 minutes

For $N = 2048^2 = 16 * 512^2$

Time for SSA is 5900 seconds = 1.64 hours

Linear scaling

Selected References

Full references refer to my thesis “Data-sparse Algorithms for Structured Matrices”
https://searchworks.stanford.edu/view/12069374

- K. Ho and L. Ying. Hierarchical interpolative factorization for elliptic operators: integral equations. Communications in Pure and Applied Mathematics 69-7 (2016).
- V. Minden, A. Damle, K. Ho, and L. Ying. Fast spatial Gaussian process maximum likelihood estimation via skeletonization factorizations. arXiv:1603.08057
- P.-G. Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions, J. Computational Physics, 205 (2005).
- S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D.W. Hogg, and M. O’Neil. Fast direct methods for Gaussian processes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38 (2016).

Thanks!