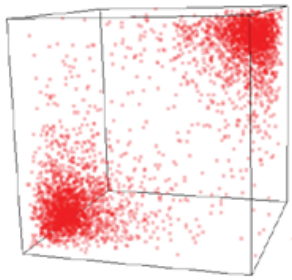
Hierarchical matrix approximations

GEORGE BIROS
padas.ices.utexas.edu

MATVEC or Kernel sum



Input

N points in $\mathbb{R}^d$: $x_1, \ldots, x_N$

N densities in $\mathbb{R}$: $w_1, \ldots, w_N$

Output

N potentials in $\mathbb{R}$: $u_1, \ldots, u_N$

$$u_i = \sum_{j=1}^{N} G(x_i, x_j) w_j$$

$$G(x_i, x_j) = \exp\left(-\frac{1}{2}\frac{\|x_i - x_j\|_2^2}{h^2}\right)$$

Gaussian	$\exp(-\|x - x_j\|^2/(2h^2))$
Laplace	$\|x - x_j\|^{2-d},\ d > 2$
Matern	$\left(\sqrt{2\nu}\|x - x_j\|\right)^{\nu} K_\nu \left(\sqrt{2\nu}\|x - x_j\|\right)$
Polynomial	$\left(x^T x_j/h + c\right)^p$
Ornstein-Uhlenbeck	$\exp(-c\|x - x_j\|)$
Multiquadratic	$\sqrt{c^2 + \|x - x_j\|_2^2}$
Inverse multiquadratic	$1/\sqrt{c^2 + \|x - x_j\|_2^2}$

KERNEL TRICK (inner products)
KERNEL DENSITY ESTIMATION
GREEN's FUNCTIONS

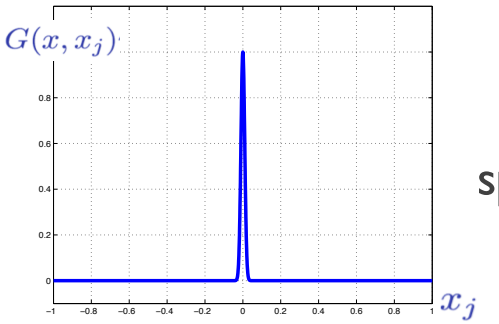
Relation to PDEs

classical N-body	potential theory
$-\Delta u(x) = \sum_{i=1}^{N} \delta(x - x_i) w_i$	$-\Delta u(x) = w(x)$
$u(x) = \sum_{i=1}^{N} \frac{1}{\|x - x_i\|} w_i$	$u(x) = \int_{x'} \frac{1}{\|x - x'\|} w(x')$

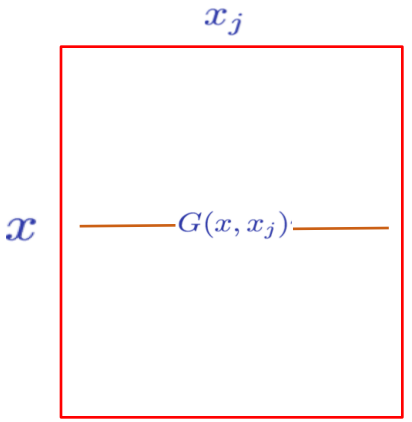
Issues with the kernel

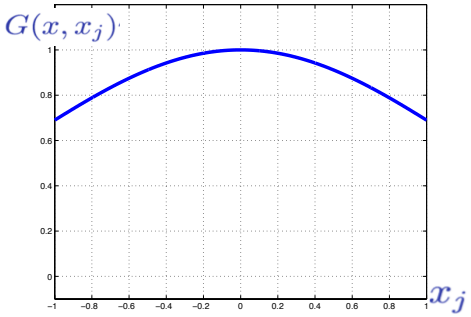
- **K** is N-by-N dense
- We need MatVecs, Solves, Eigenvalues
- Standard algorithms scale as $O(N^2)$
- Key idea: approximate K
  - But how?

Kernel $G(x, x_j)$



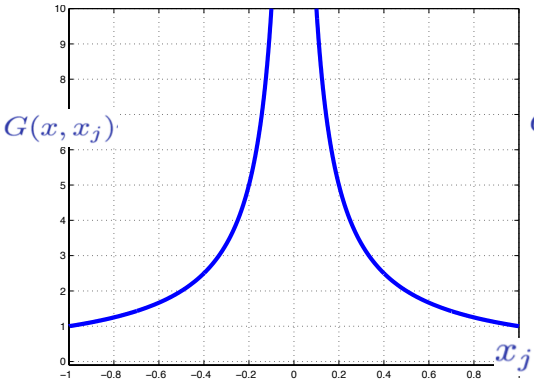
sparse - *nearest-neighbor*

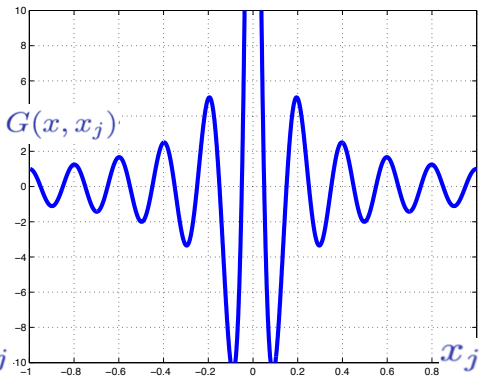




globally smooth/ low rank - *Nystrom*

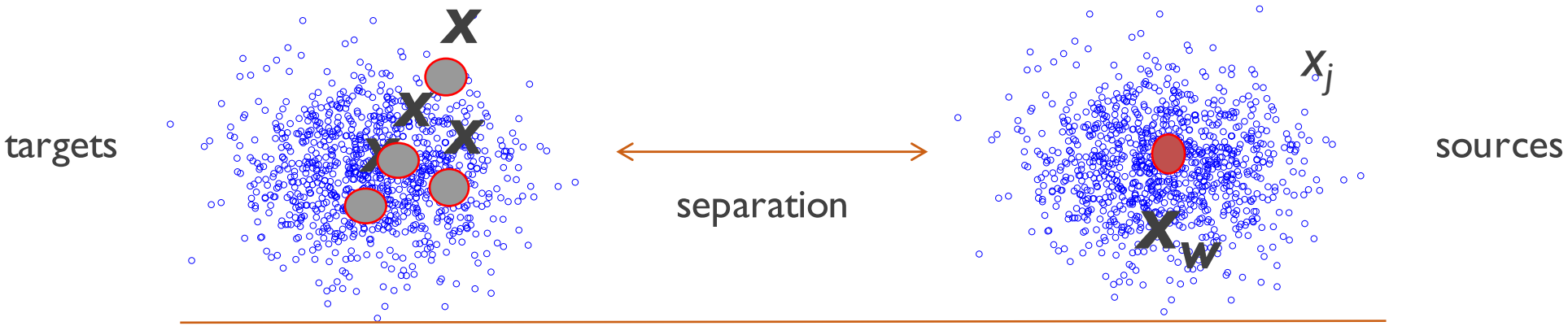
Neither sparse nor low rank - *N-body methods*





Hierarchical matrices 101

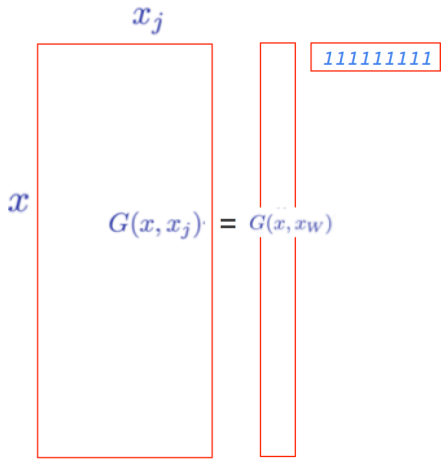
Idea I: Far-field interactions



x: Target, x_j: sources
w_j: weights

$$u(x) = \sum_j G(x, x_j) w_j$$

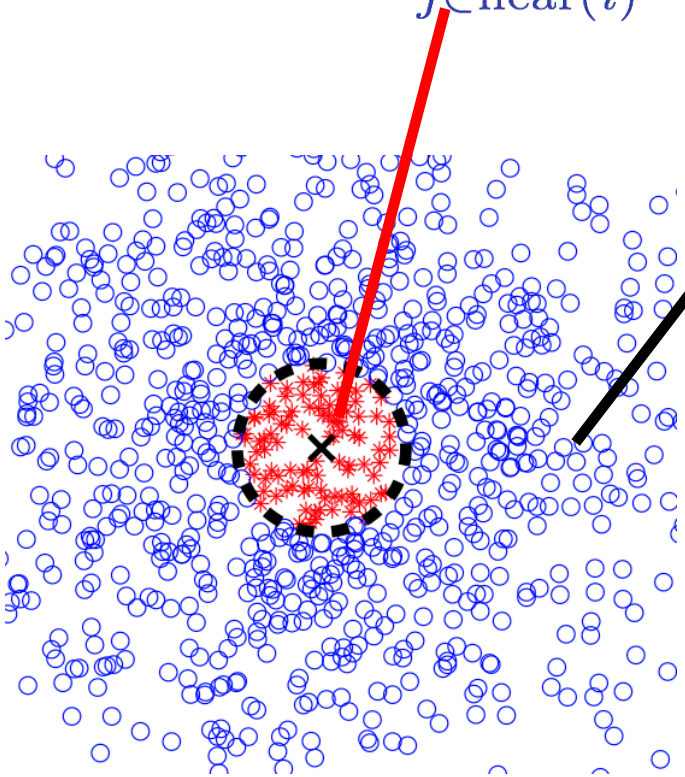
1. compute $W = \sum_i w_j$
2. choose x_W
3. $u(x) \approx G(x, x_W) W$



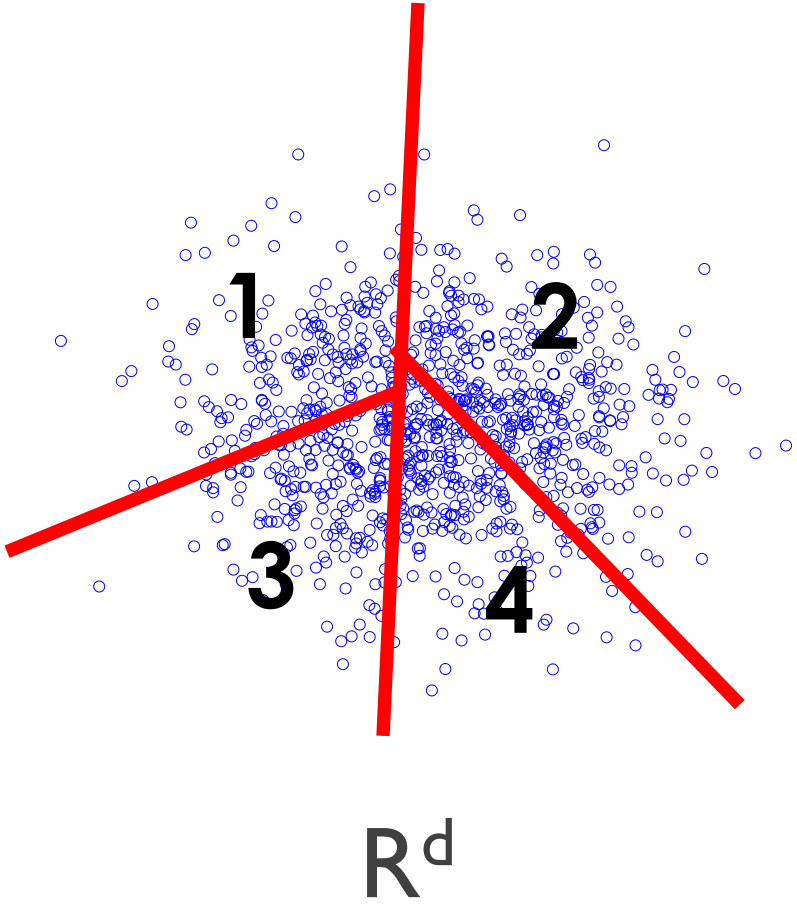
LOW RANK APPROXIMATION
$O(N^3) \rightarrow O(N)$

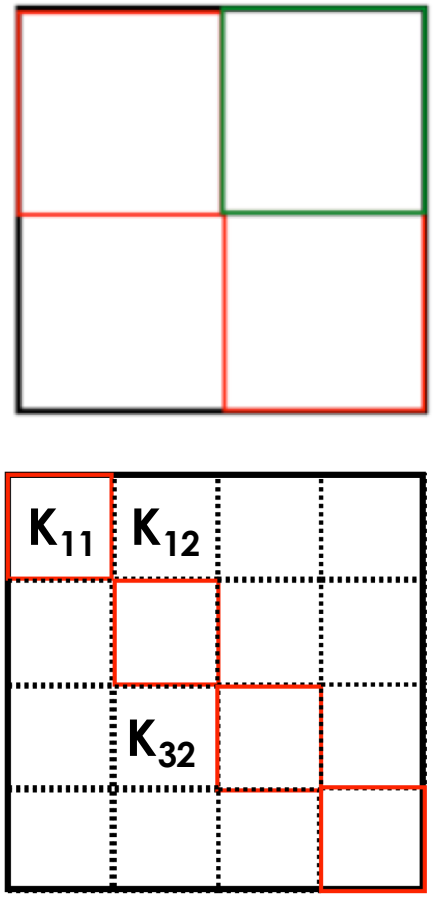
Idea II: Near-/Far-field split

$$u_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} G(x_i, x_j) w_j = \sum_{j \in \text{near}(i)} G_{ij} w_j + \sum_{j \in \text{far}(i)} G_{ij} w_j$$



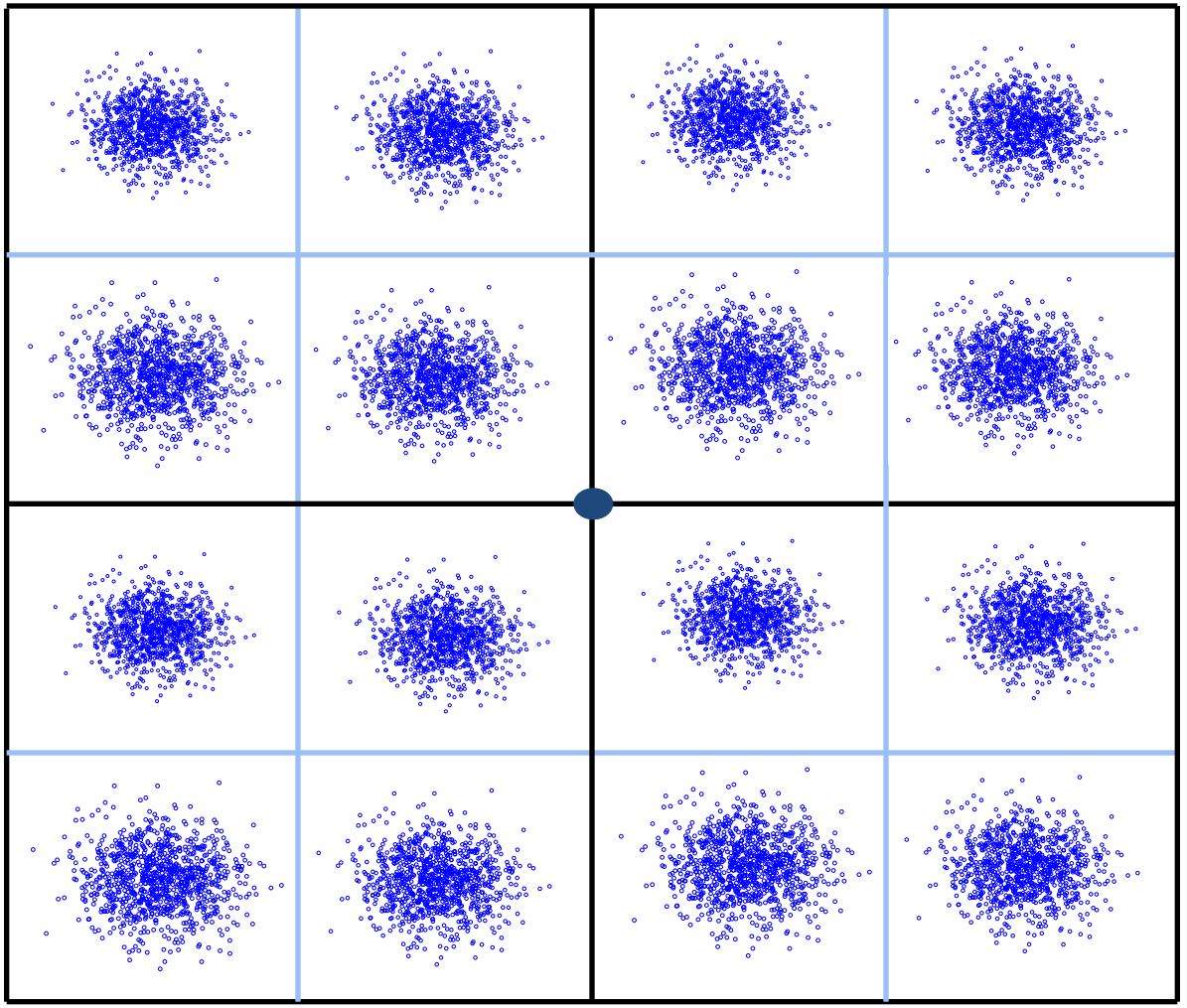
Idea III: recursion





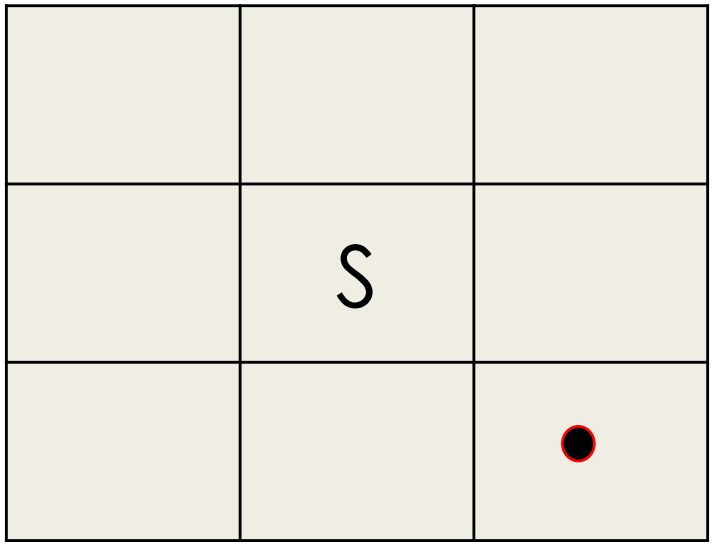
Matrix partitioning

Single level



Well separated boxes (Near- / Far-field decomp)

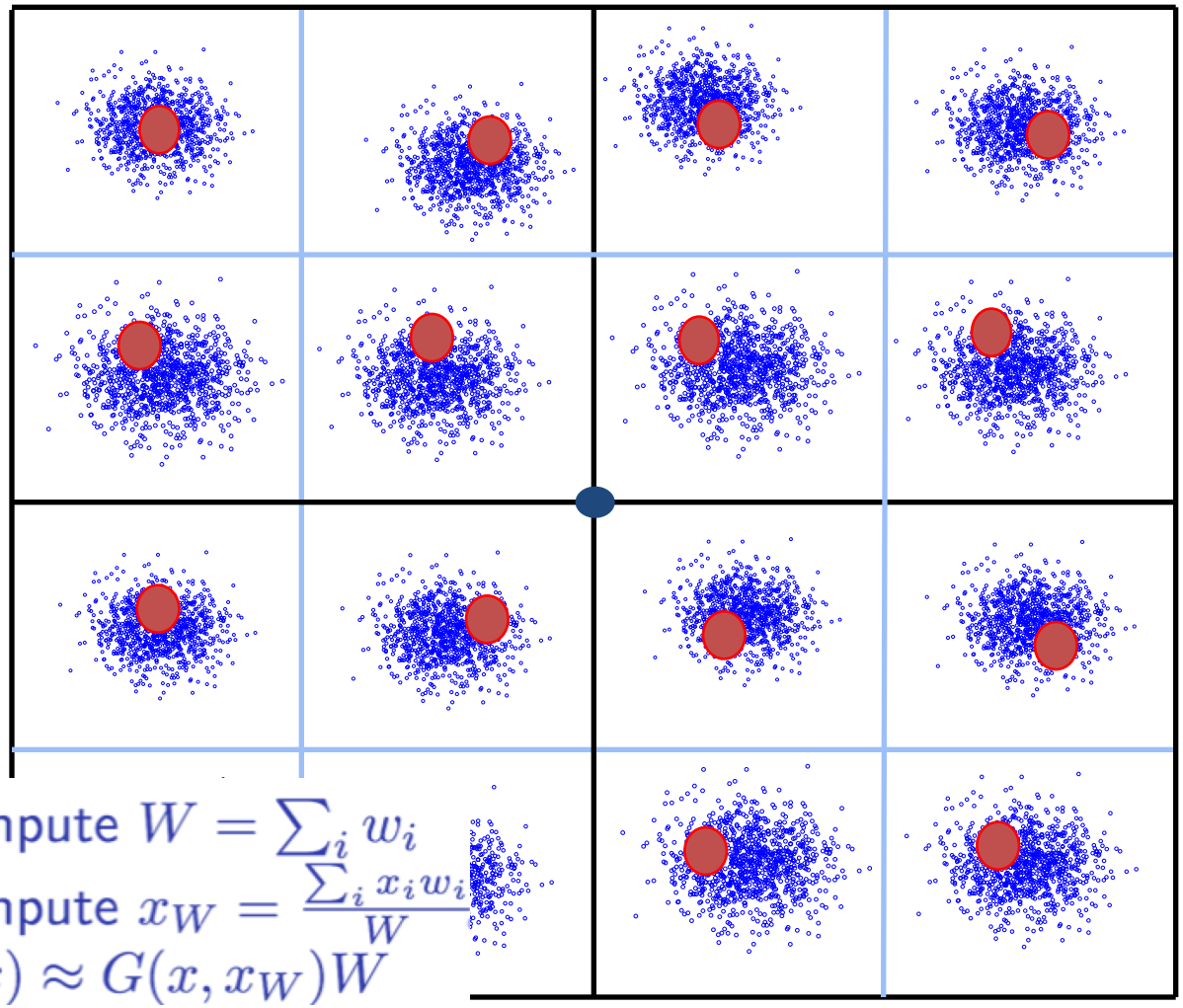
- Source (S) box and target point
- **The effect of S on any T *not* in the gray area can be approximated**



well separated

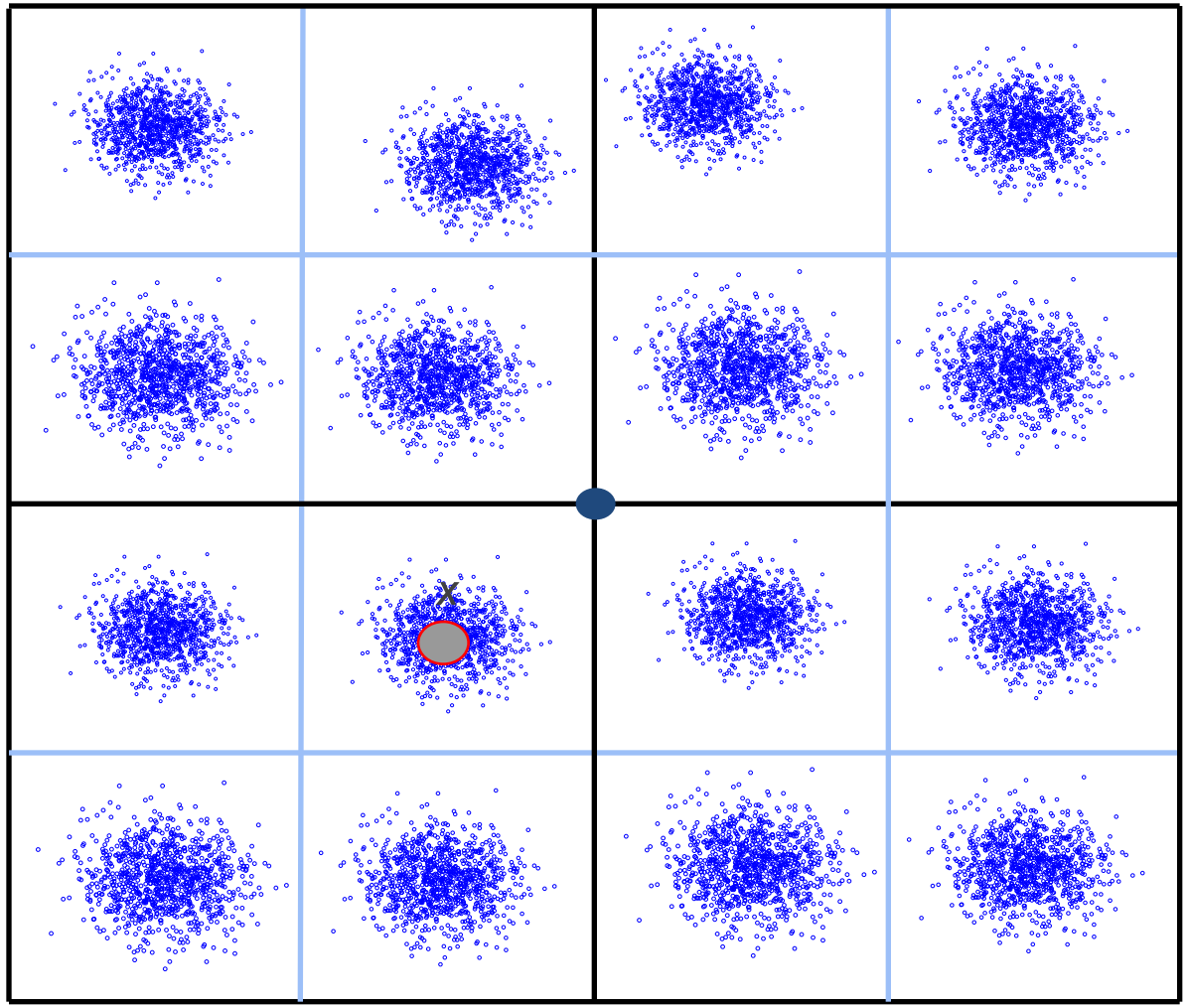
not well separated

Single level – averaging

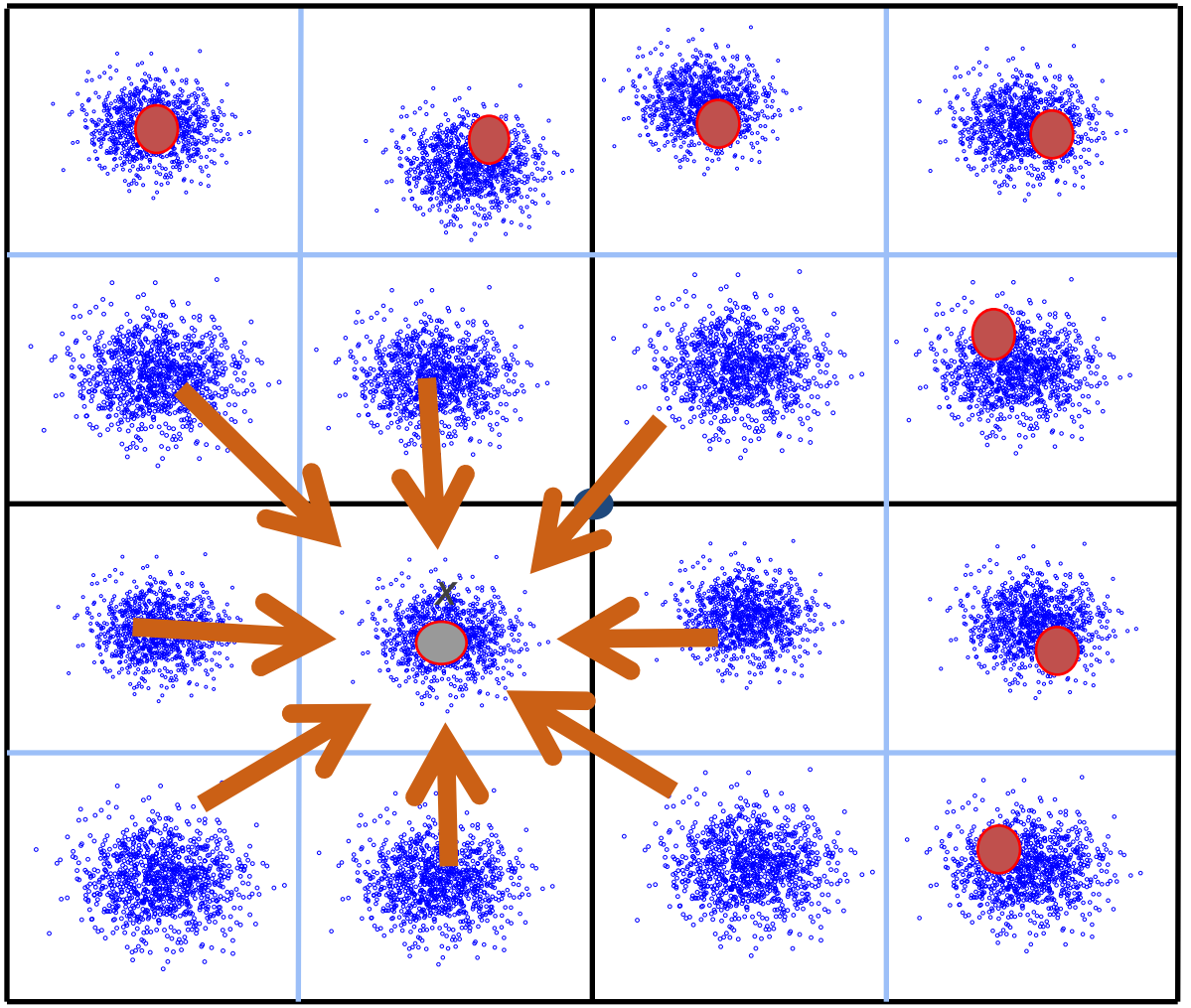


1. compute $W = \sum_i w_i$
2. compute $x_W = \frac{\sum_i x_i w_i}{W}$
3. $u(x) \approx G(x, x_W)W$

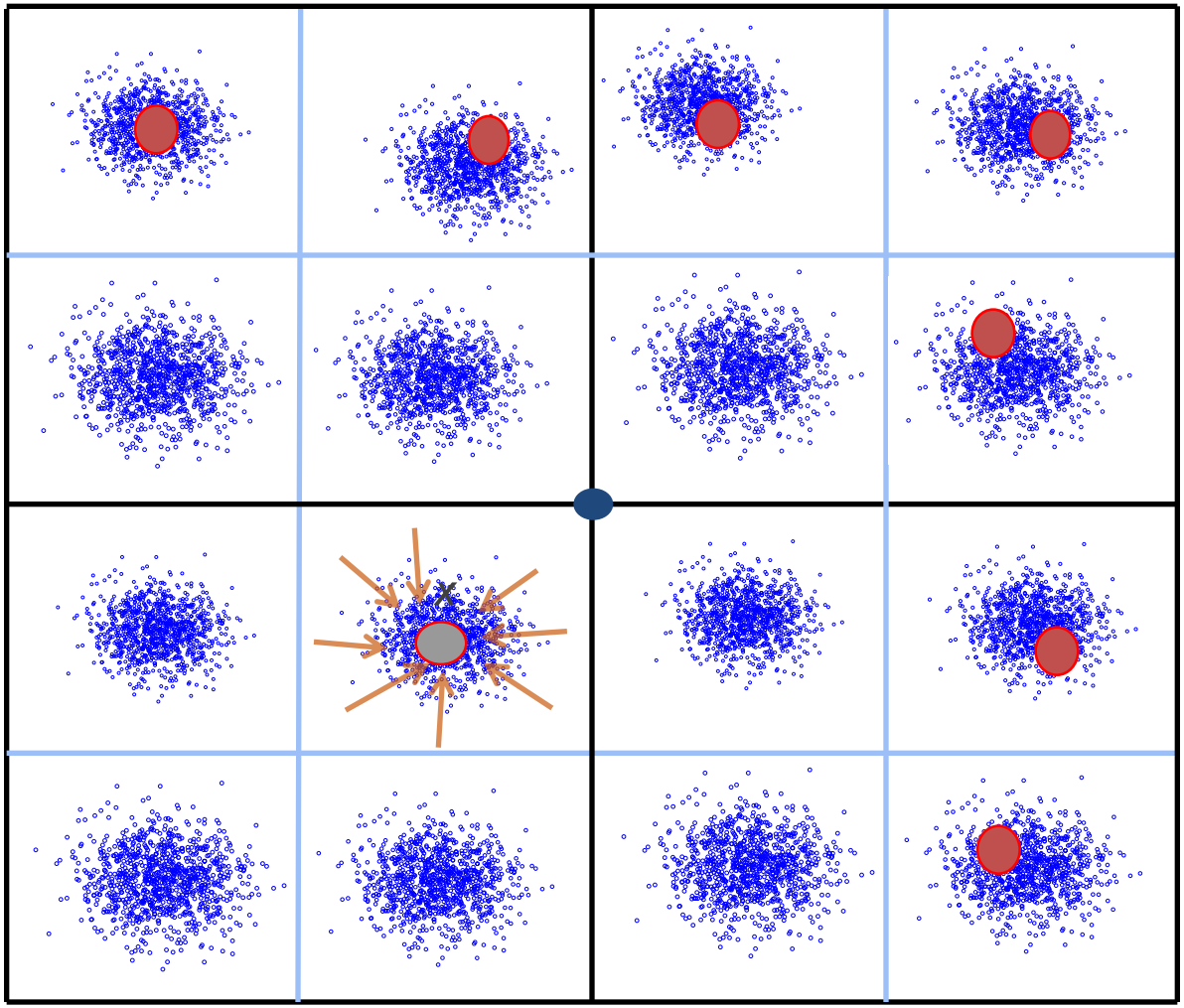
Single-level Evaluation



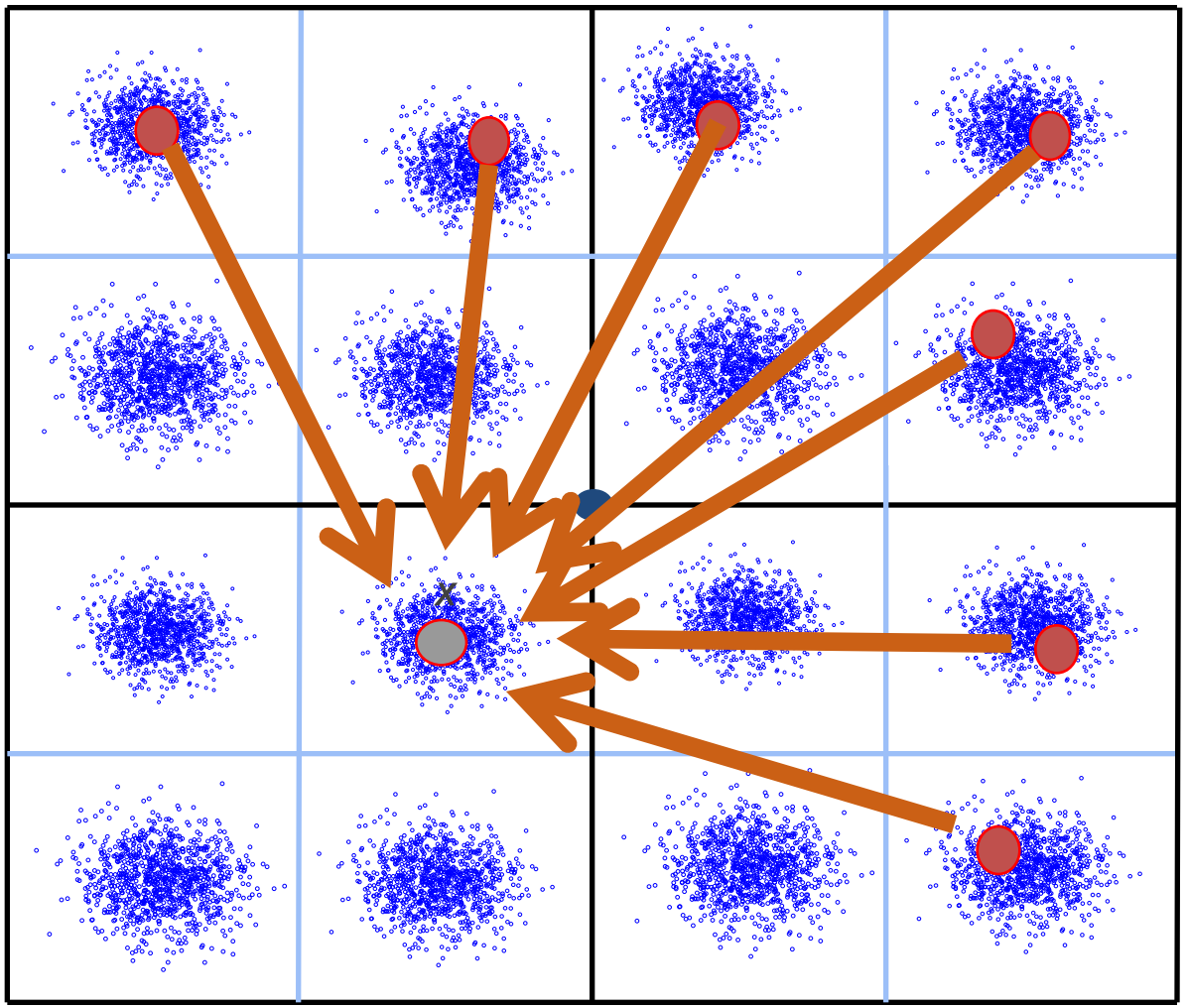
Single level – evaluation



Direct eval

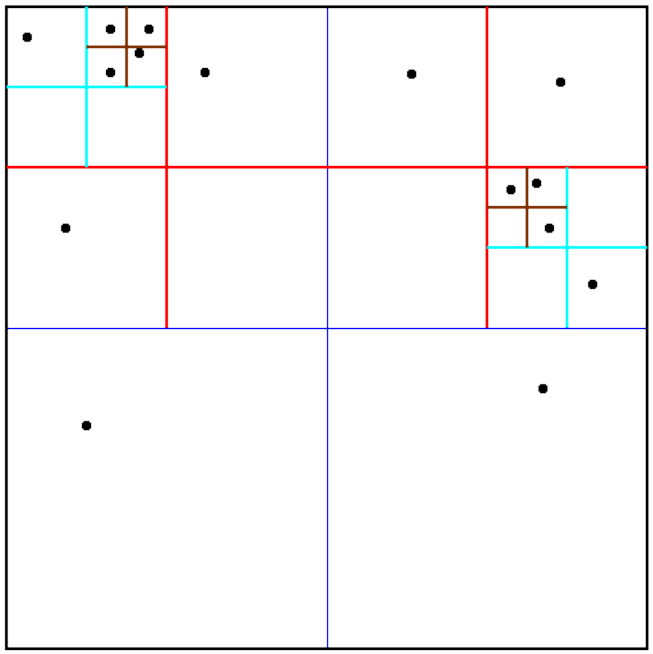


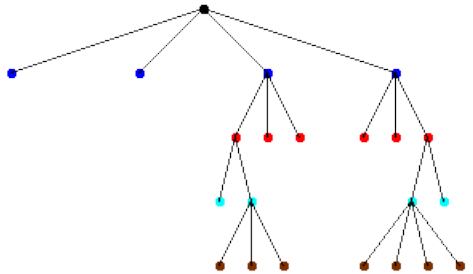
evaluation, far field



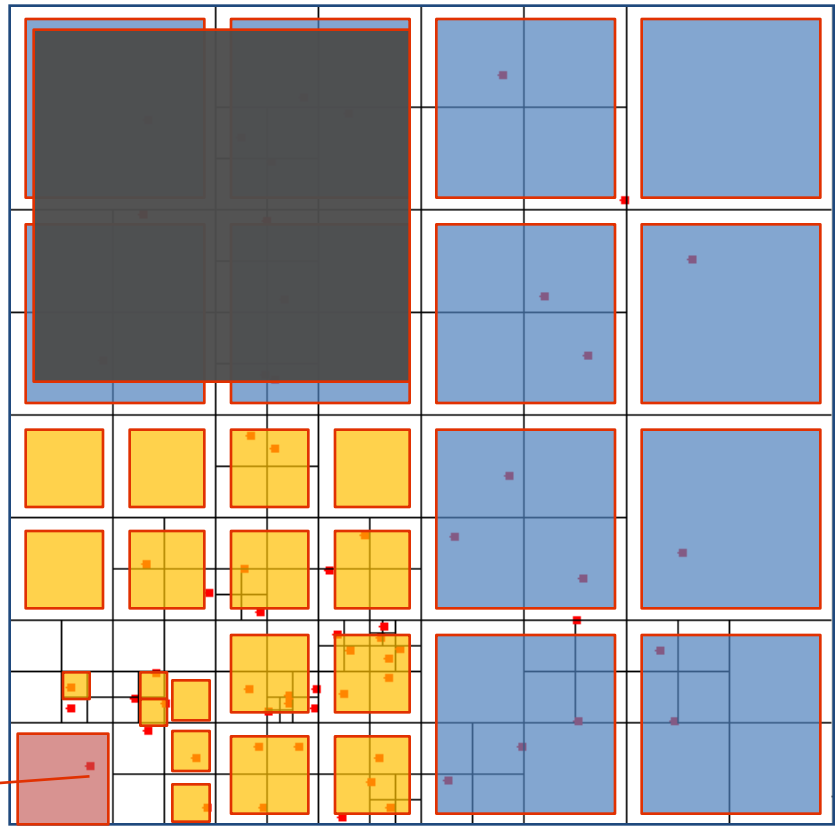
Adaptive tree

Adaptive quadtree where no square contains more than 1 particle





Treecodes - evaluation



Tree construction (quadtree in 2D)

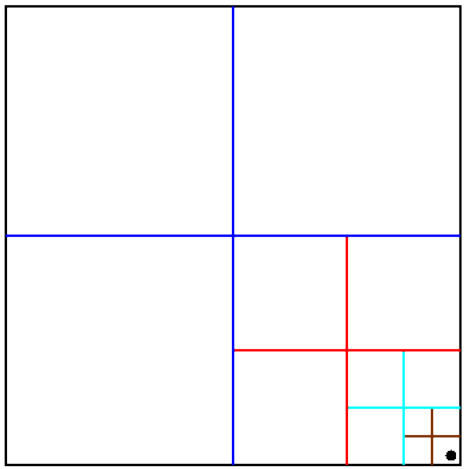
```
function quadtree_insert(j,node)
if node has children
   determine child c to which j belongs
   quadtree_insert(j,c)
elseif node contains a particle p
   add four children of n
   move p to the appropriate child
   determine child c to which j belongs
   quadtree_insert(j,c)
else
   store particle j at node
end
```

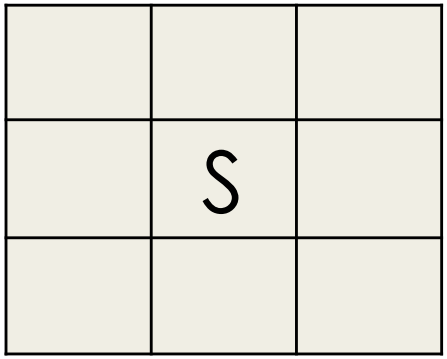
```
classdef Node;
    properties
      parent
      children  % Node array
      particles % points
      mass % total mass
      center % center of mass
    end
  end
```

Average quadtree

```
function quadtree_average(node)
if node has children
    for each child c
      quadtree_average(c)
      average;  % depends on the scheme
    end
else % node is leaf
    average_points;
end
```

Evaluate






```
% evalute force in all points within a node.
function quadtree_evaluate(targetNode,sourceNode)

if well_separated(targetNode,sourceNode) == true
  approximate_evaluate(targetNode,sourceNode)
  recurse=false
else
  if root has children
    recurse=yes;
  else % neighbor interactions
    direct_evaluation(targetNode,sourceNode)
  end
%

if recurse==yes
  for each child c of sourceNode
    quadtree_evaluate(targetNode,c)
  end
end
```

Complexity

- Building the tree: O(N log N)
- Averaging: O(N)
- Evaluation: O(N logN)

tree depth : logN
for every point log(N) calculation

Averaging – far field approximations

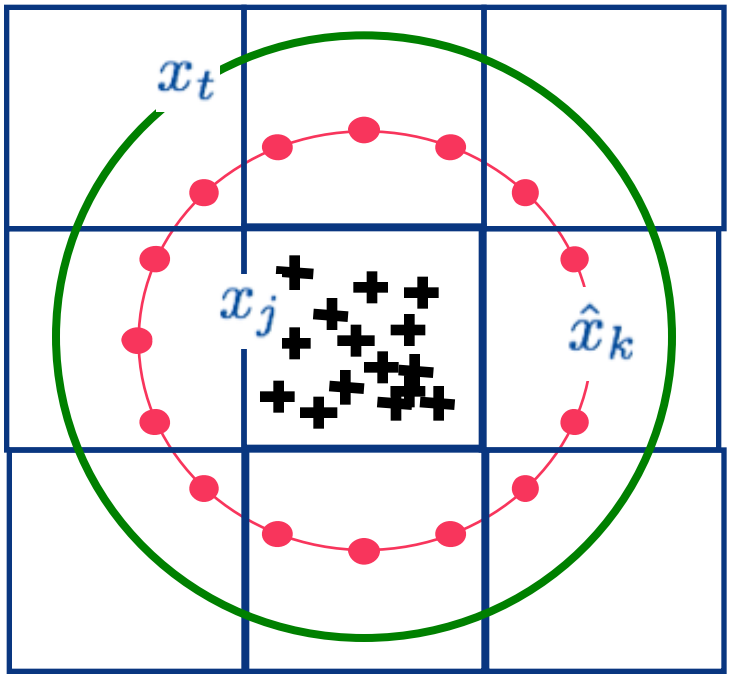
Instead of using just average density and center of mass, use *several* "equivalent" sources.

The original sources are the '+' and '-' in the figure

The equivalent sources placed at the red locations, equispaced on an *enclosing* circle.
The densities are computed by

1. $$u_t = \sum_{j=1}^{N} G(x_t, x_j) w_j$$

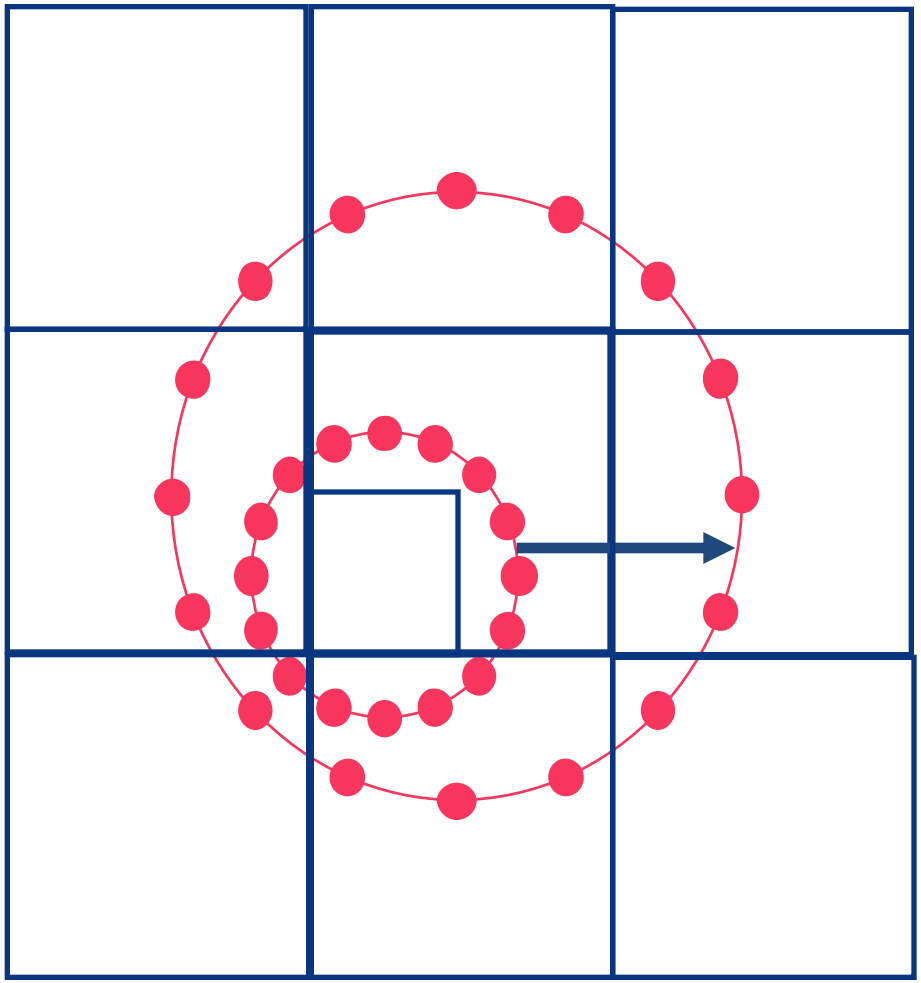
2. $$\sum_{k=1}^{M} G(x_t, \hat{x}_k) \hat{w}_k = u_t$$



Least squares problem

Node averaging

The same idea can be used for averaging the equivalent densities of the children of a node.



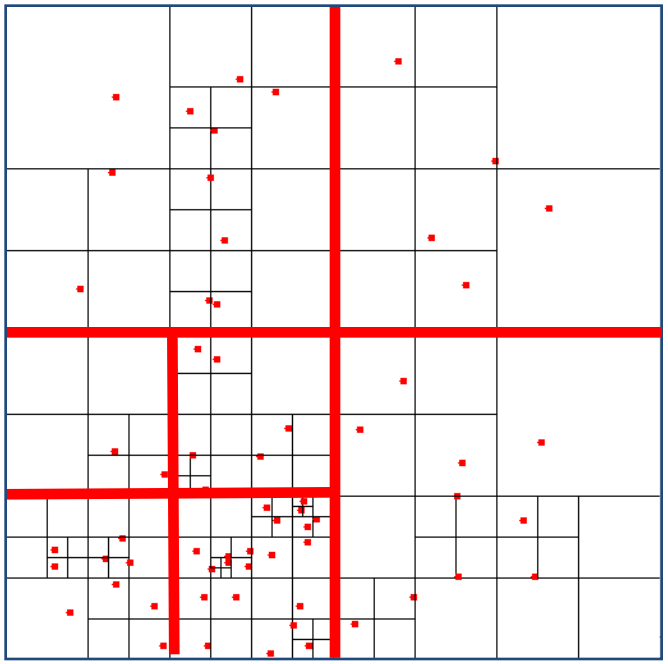
Issues with high D

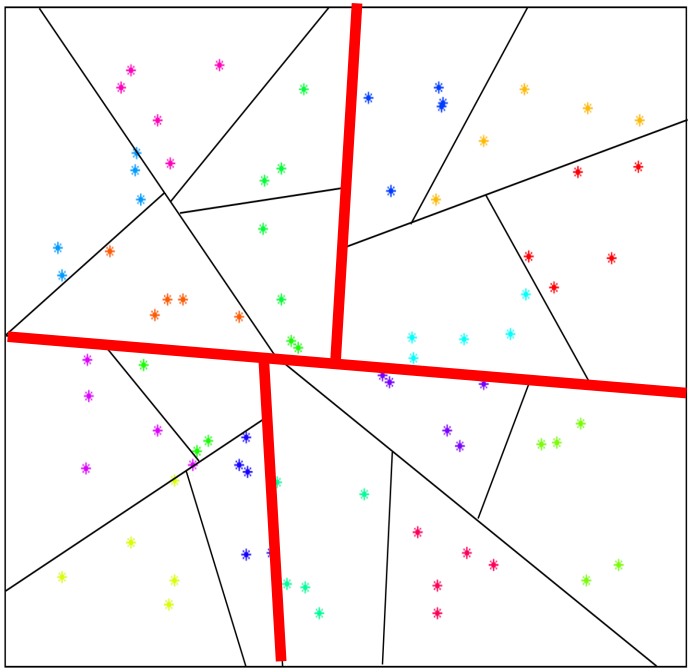
- Approximation of far-field
  - (polynomial in ambient-D)
- Pruning (near-far field)
  - (polynomial in ambient-D)
- Practically, no scalable algorithms
  - no parallelism, no control of complexity/accuracy

- Nystrom method assumes global low-rank
  - doesn't scale with problem size

Regular decompositions

- Low dimensions
  High dimensions





LOW-RANK approximation

- Given K
  - SVD is out of question
  - How can we create a low rank approximation

HMT - paper

PROTO-ALGORITHM: SOLVING THE FIXED-RANK PROBLEM

Given an $m \times n$ matrix $\boldsymbol{A}$, a target rank k, and an oversampling parameter p, this procedure computes an $m \times (k+p)$ matrix $\boldsymbol{Q}$ whose columns are orthonormal and whose range approximates the range of $\boldsymbol{A}$.

1 Draw a random $n \times (k+p)$ test matrix $\boldsymbol{\Omega}$.
2 Form the matrix product $\boldsymbol{Y} = \boldsymbol{A\Omega}$.
3 Construct a matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$.

HMT - paper

THEOREM 1.1. *Suppose that* $\boldsymbol{A}$ *is a real* $m \times n$ *matrix. Select a target rank* $k \geq 2$ *and an oversampling parameter* $p \geq 2$, *where* $k + p \leq \min\{m, n\}$. *Execute the proto-algorithm with a standard Gaussian test matrix to obtain an* $m \times (k+p)$ *matrix* $\boldsymbol{Q}$ *with orthonormal columns. Then*

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \leq \left[1 + 9\sqrt{k+p} \cdot \sqrt{\min\{m, n\}}\right] \sigma_{k+1}$$

HMT - SVD

Prototype for Randomized SVD

Given an $m \times n$ matrix $\boldsymbol{A}$, a target number k of singular vectors, and an exponent q (say, $q = 1$ or $q = 2$), this procedure computes an approximate rank-$2k$ factorization $\boldsymbol{U\Sigma V}^*$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal, and $\boldsymbol{\Sigma}$ is nonnegative and diagonal.

Stage A:

1 Generate an $n \times 2k$ Gaussian test matrix $\boldsymbol{\Omega}$.

2 Form $\boldsymbol{Y} = (\boldsymbol{AA}^*)^q \boldsymbol{A\Omega}$ by multiplying alternately with $\boldsymbol{A}$ and $\boldsymbol{A}^*$.

3 Construct a matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$.

Stage B:

4 Form $\boldsymbol{B} = \boldsymbol{Q}^* \boldsymbol{A}$.

5 Compute an SVD of the small matrix: $\boldsymbol{B} = \widetilde{\boldsymbol{U}} \boldsymbol{\Sigma V}^*$.

6 Set $\boldsymbol{U} = \boldsymbol{Q}\widetilde{\boldsymbol{U}}$.

Note: The computation of $\boldsymbol{Y}$ in step 2 is vulnerable to round-off errors. When high accuracy is required, we must incorporate an orthonormalization step between each application of $\boldsymbol{A}$ and $\boldsymbol{A}^*$; see Algorithm 4.4.

Best known results.

- But cannot be applied
- It requires MATVEC
  - which we are trying to approximate.

Nystrom method - motivation

- Assume K is rank r
- $\exists$ $r \times r$ block of K that is full rank.
  Assume K_{11} is full rank

Let $K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}$

Then notice that we can write K as

$$\begin{bmatrix} K_{11} \\ K_{21} \end{bmatrix} K_{11}^{-1} \begin{bmatrix} K_{11} & K_{12} \end{bmatrix} = \begin{bmatrix} K_{11} \\ K_{21} \end{bmatrix} \begin{bmatrix} I & K_{11}^{-1} K_{12} \end{bmatrix}$$

$$= \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{21} K_{11}^{-1} K_{12} \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}$$

Then columns of $\begin{bmatrix} K_{11} \\ K_{21} \end{bmatrix}$ form a basis so that $\exists\, S \in \mathbb{R}^{r \times N-r}$

$$\begin{bmatrix} K_{12} \\ K_{22} \end{bmatrix} = \begin{bmatrix} K_1 \\ K_{21} \end{bmatrix} S$$

$K_{22} = K_{21} S = K_{21} K_{11}^{-1} K_{12}$

Therefore we need to store

Notice that

$$\begin{matrix} N \\ \end{matrix} \begin{bmatrix} K_1 \\ K_{21} \end{bmatrix} \begin{matrix} r \\ \end{matrix} \overbrace{\begin{bmatrix} I & K_{11}^{-1} K_{12} \end{bmatrix}}^{N}$$

is a rank r decomposition.

In practice K is not exactly rank r

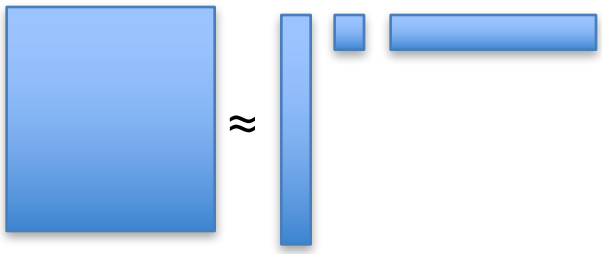
We commit an error of $K_{22} - K_{21} K_{11}^{-1} K_{12}$

Nystrom method (and its variants)

- **Low-rank** decomposition of G
- random sampling of **O(s)** points, **s** : target **rank**

- Work $G(x, x_j) = G_{x,s}(G_{\ell,s})^{\dagger} G_{s,x_j}$

$$Ns + s^3$$



- Error

$$\|G - \tilde{G}\| \leq \sqrt{1 + 6N/s}\ \sigma_{s+1}(G)$$

Talwalkar et al, 10

Interpolative decomposition

Let $K \in \mathbb{R}^{n \times m}$. Let $\mathcal{S}$ be an index set with $|\mathcal{S}| = s$ and $1 \leq \mathcal{S}_j \leq m$. Let $K_S = K(:, \mathcal{S})$ be the columns of K indexed by $\mathcal{S}$, and K_R be the remaining *unskeletonized* columns of K. Assuming $s < m < n$ and that K_S is full-rank, we can approximate the columns of K_R by $K_S P$, where $P = K_S^{\dagger} K_R$, $P \in \mathbb{R}^{s \times m}$. The ID consists of the index set $\mathcal{S}$, referred to as the *skeleton*, and matrix P. Following [21, 5], we refer to the construction of this approximation for a matrix as *skeletonization*.

In order to compute an ID such that $\|K_R - K_S P\|$ is small, we employ a pivoted QR factorization to obtain $K\Pi = QR$ for some permutation Π, an orthonormal matrix Q, and upper triangular matrix R. The skeleton $\mathcal{S}$ corresponds to the first s columns of $K\Pi$, and the matrix P can be computed from R in $\mathcal{O}(s^3 + s^2(m-s))$ time. It can be shown [16] that

$$\|K_R - K_S P\| \leq \sqrt{1 + m(m-s)}\,\sigma_{s+1}(K), \tag{2.1}$$

where σ_{s+1} is the ith singular value of K. The overall cost for $s < m < n$ is $\mathcal{O}(nm^2)$.

Random features

- A shift invariant kernel $k(\Delta)$ is the Fourier transformation of a non-negative measure

$$k(\mathbf{x}-\mathbf{y}) = \int_{\mathcal{R}^d} p(\omega) e^{j\omega'(\mathbf{x}-\mathbf{y})}\, d\omega = E_\omega[\zeta_\omega(\mathbf{x})\zeta_\omega(\mathbf{y})^*].$$

- Eg.

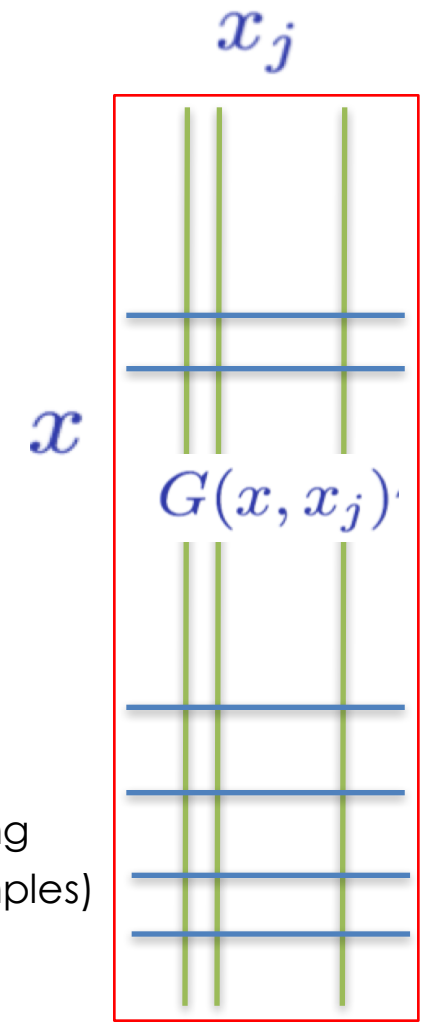
Kernel Name	$k(\Delta)$	$p(\omega)$
Gaussian	$e^{-\frac{\|\Delta\|_2^2}{2}}$	$(2\pi)^{-\frac{D}{2}} e^{-\frac{\|\omega\|_2^2}{2}}$
Laplacian	$e^{-\|\Delta\|_1}$	$\prod_d \frac{1}{\pi(1+\omega_d^2)}$
Cauchy	$\prod_d \frac{2}{1+\Delta_d^2}$	$e^{-\|\Delta\|_1}$

- Does not require inversion
- Extremely fast
- But not for all kernels

Far-field approximation

$$G(x, x_j) = G_{x,s}(G_{\ell,s})^{\dagger} G_{s,x_j}$$

- uniform sampling large errors
- norm sampling better errors
- leverage sampling close to optimal, requires SVD(G)
- range space sampling requires fast matvec(G)
- ASKIT: random sampling + **nearest neighbors**
  - approximates norm sampling, sometimes behaves like leverage sampling
  - also uses kernel properties (decreasing/ increasing require different samples)
  - adaptive rank selection, only parameter is absolute performance



Random features for $\exp\left(-\frac{x^T x}{2}\right)$

```
function [F,so] = create_random_features(x, m, s)

F=[];so=[];
if nargin<1; selftest; return; end
if nargin<3, s=rng; so=s; end;

[n,d] = size(x);
rng(s);
om = randn(m,d);
b= 2*pi*rand(1,m);
b = repmat (b,n,1);
F = sqrt(2)/sqrt(m) *cos(x*om'+b);
```

```
function selftest
n =  1000;
d =  20;
x = rand(n,d);
h = 2;
rho = distance(x',x')/h;
K = exp( -1/2  * rho.^2 );
m = 10; %it doen't make sense to get it more than

s = rng;
F = create_random_features(x/h, m, s);
fprintf('relative L2 norm error =%1.2E\n', norm(F*F' - K)/norm(K));

% compute best m-rank approximation.
S=svd(K);
fprintf('Relateve L2 norm of SVD = %1.2E\n', S(m)/S(1));
```