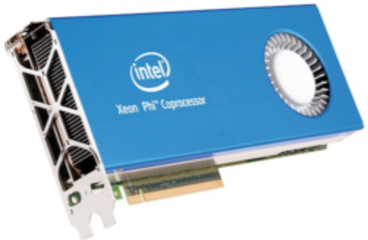


DTU

Block algebraic methods for CT and their performance

Hans Henrik Brandenborg Sørensen
DTU Computing Center
<hhbs@dtu.dk>

DTU DCC Computing Center

GPUlab DTU Informatics

DTU Compute
Department of Applied Mathematics and Computer Science

Contributions from students etc.

SIAM J. SCI. COMPUT.
Vol. 36, No. 5, pp. C524–C546

MULTICORE PERFORMANCE OF BLOCK ALGEBRAIC ITERATIVE RECONSTRUCTION METHODS*

HANS HENRIK B. SØRENSEN[†] AND PER CHRISTIAN HANSEN[†]

Abstract. Algebraic iterative methods are routinely used for solving the ill-posed sparse linear

Block Algebraic Methods for 3D Image Reconstructions on GPUs

Kenneth Kjær Nielsen

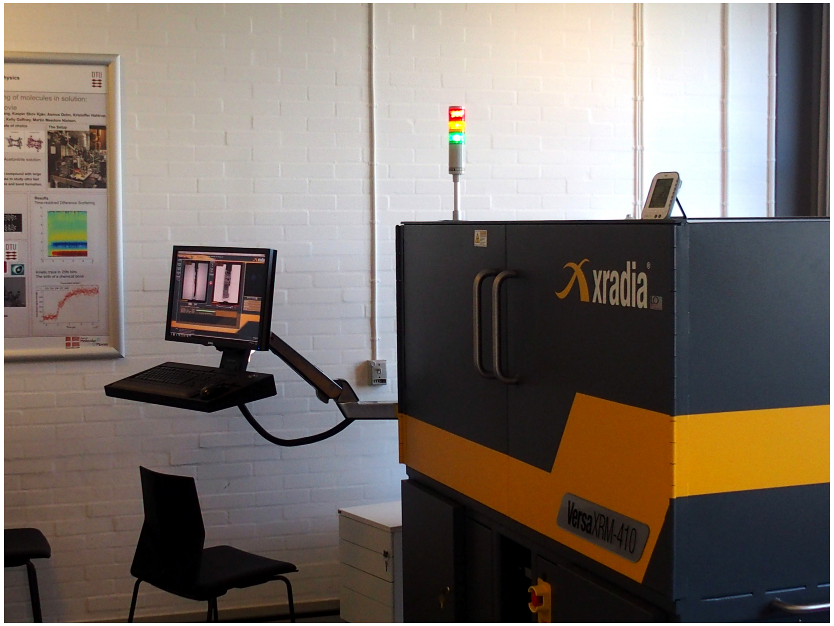
Kongens Lyngby 2014

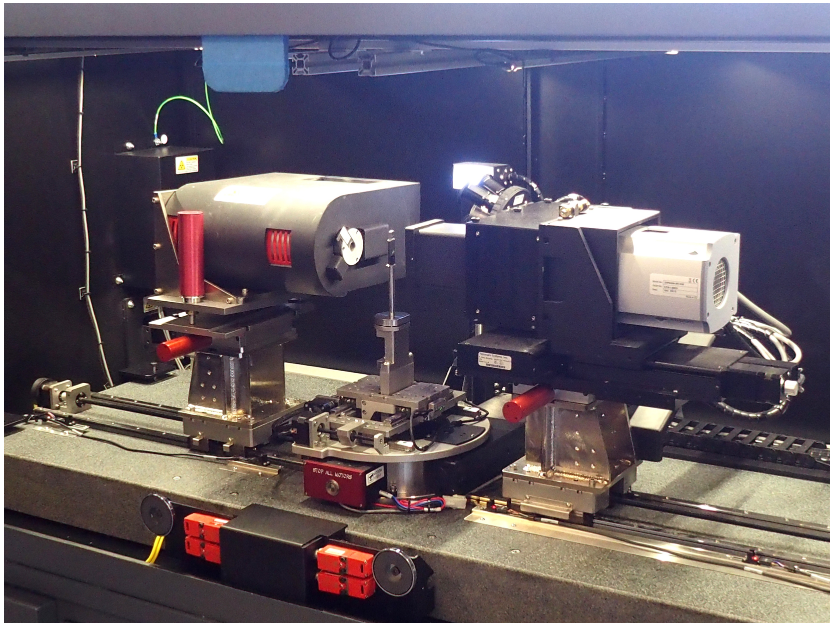
High-Performance Computing for Block-Iterative Tomography Reconstructions

Mads Friis Hansen

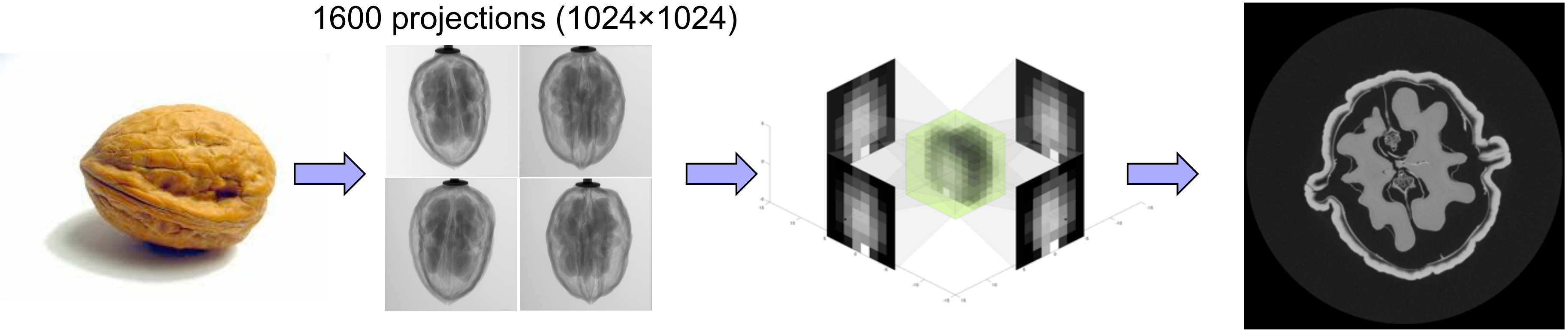
Kongens Lyngby 2016

The Walnut case





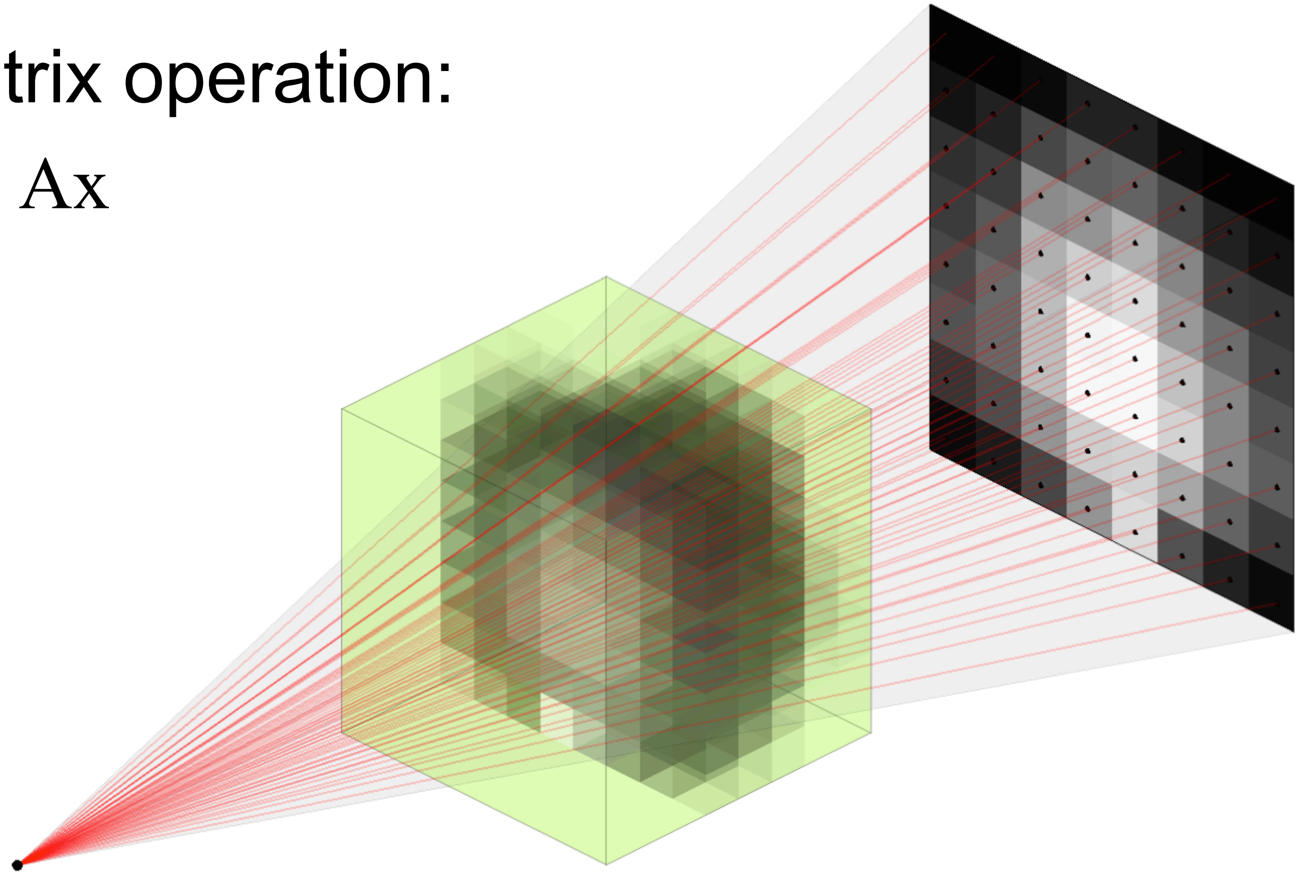
1600 projections (1024×1024)



Forward projection

- Matrix operation:

  $$y = Ax$$

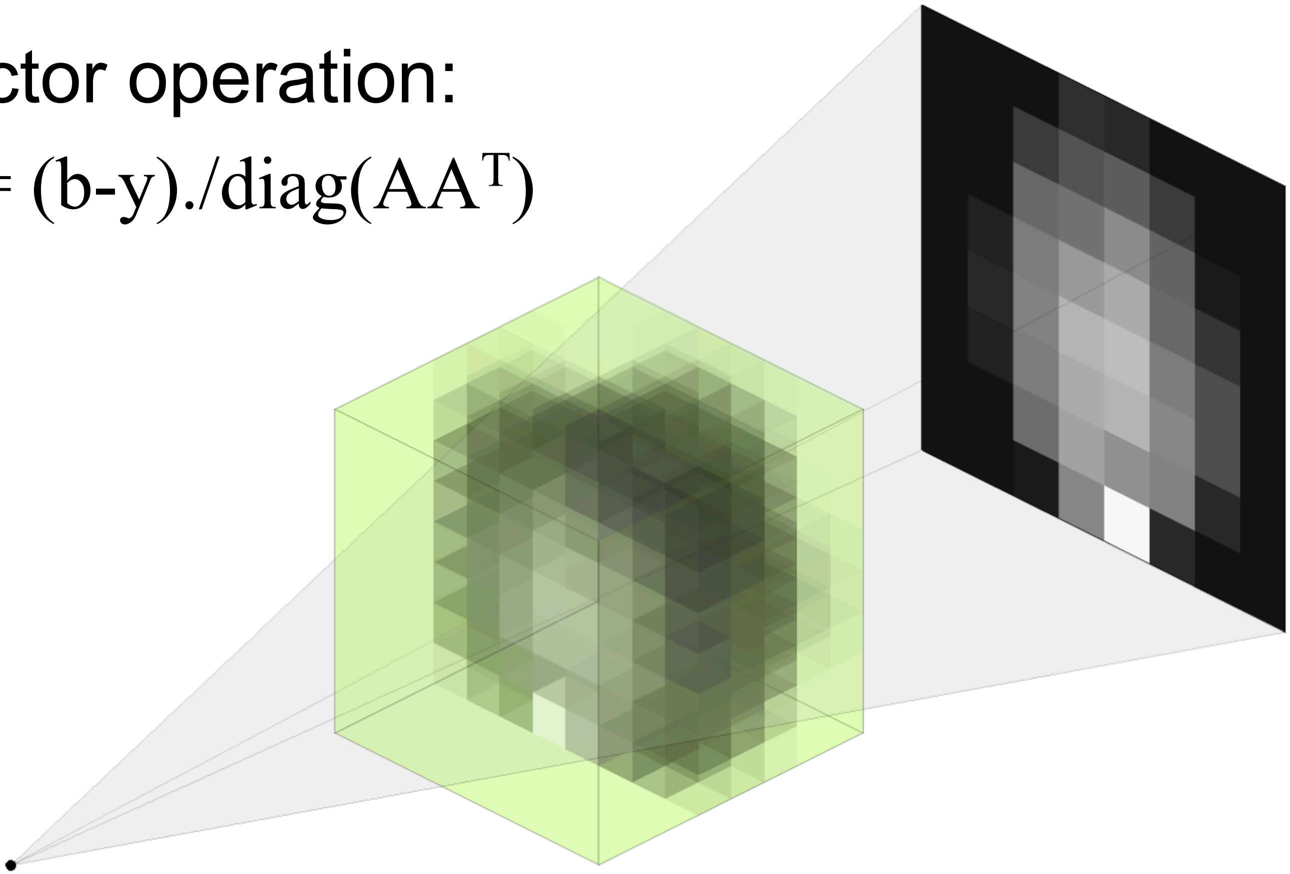


- Several different types:
  - Line, splatting, footprint, interpolation, ext. rays

Correction

- Vector operation:

  $y := (b-y)./\mathrm{diag}(AA^T)$

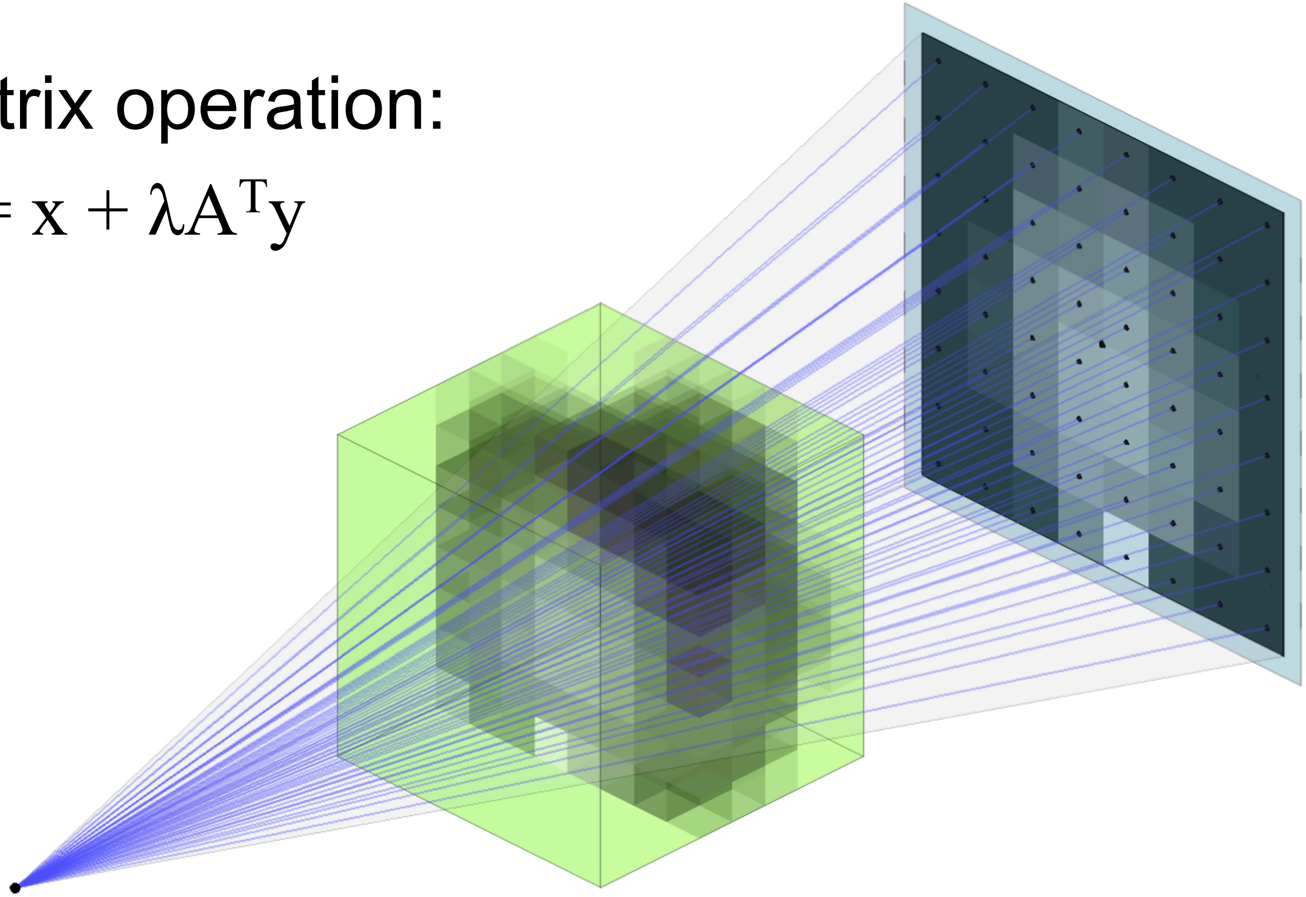


- AA^T can be pre-calculated (as forward projection).

Back projection

- Matrix operation:

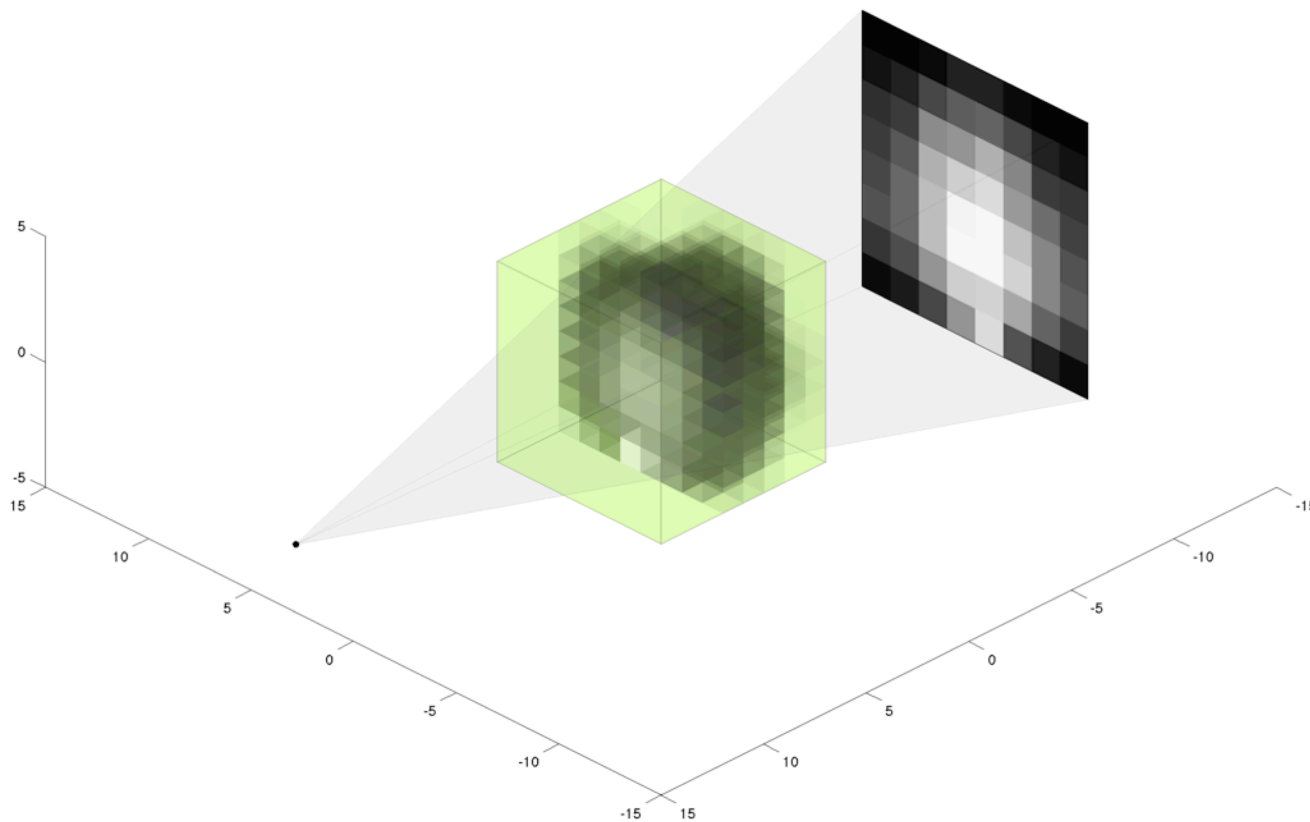
  $$x := x + \lambda A^T y$$



- Several different types:
  - Line, splatting, footprint, interpolation, ext. rays

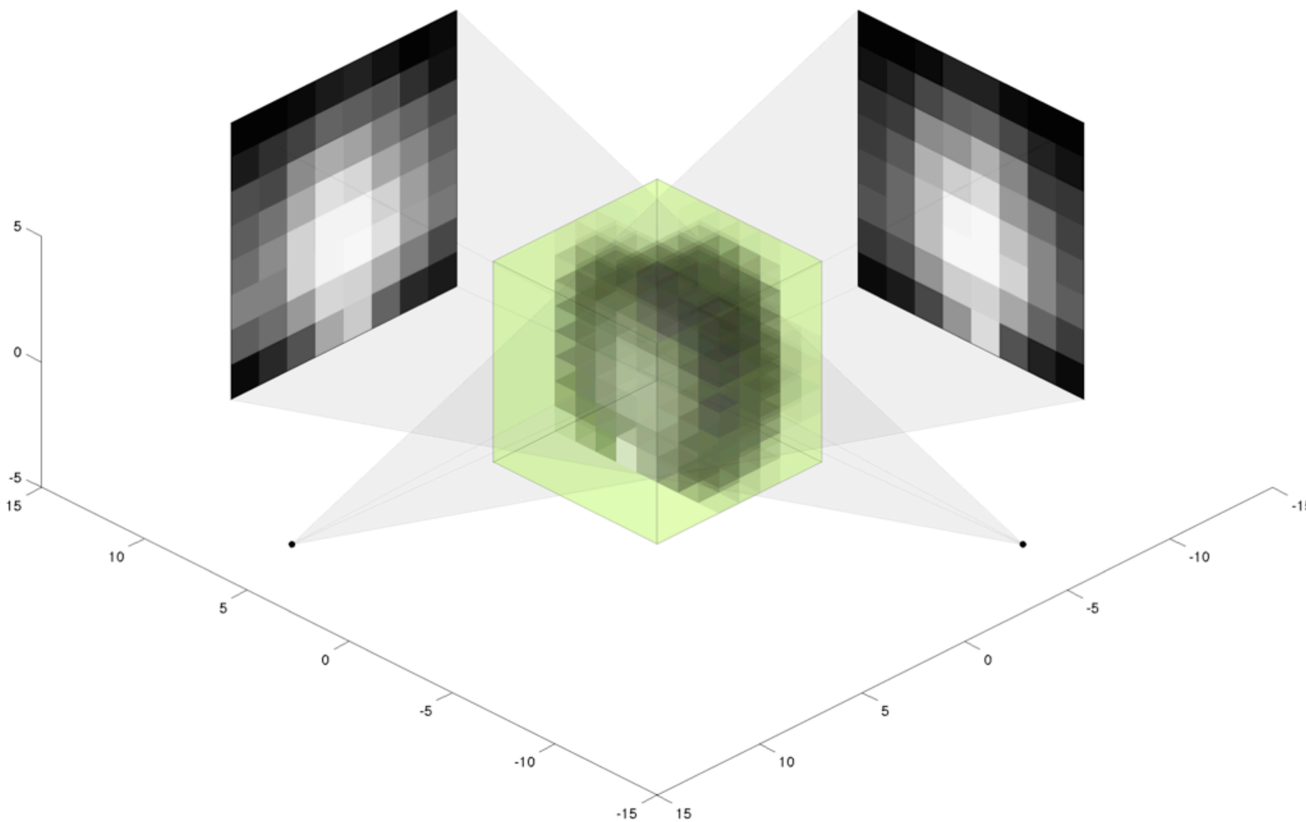
SIRT

- Forward projections ($\mathbf{A}\mathbf{x}$)



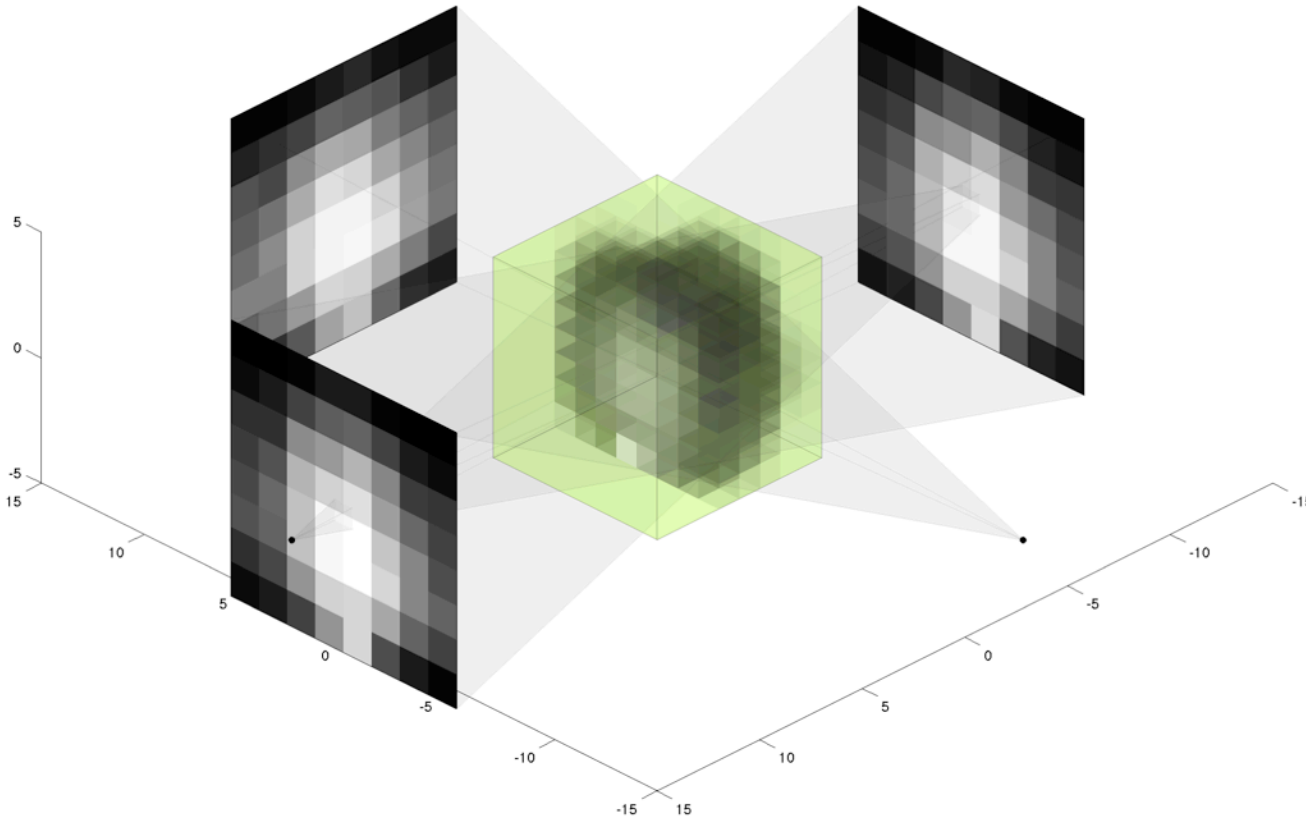
SIRT

- Forward projections ($\mathbf{A}\mathbf{x}$)



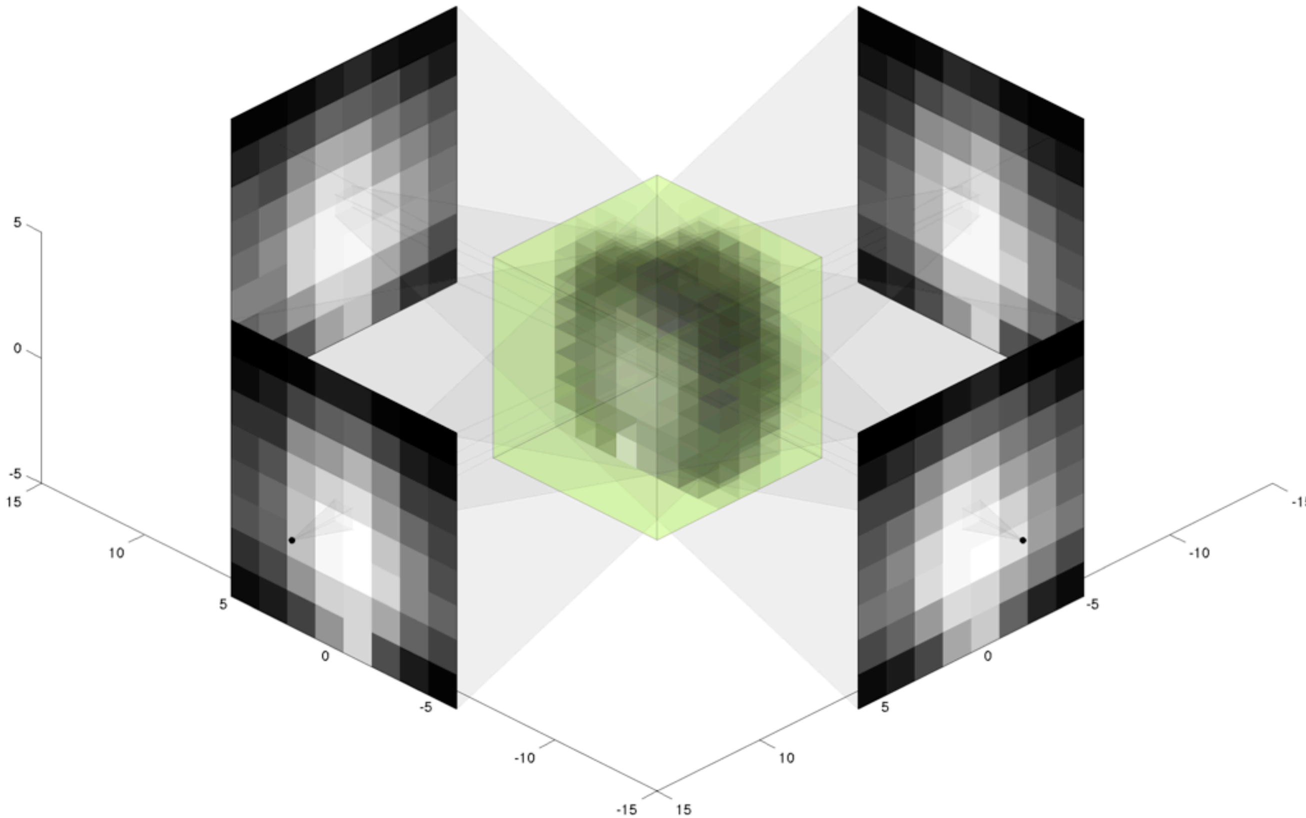
SIRT

- Forward projections ($\mathbf{A}x$)



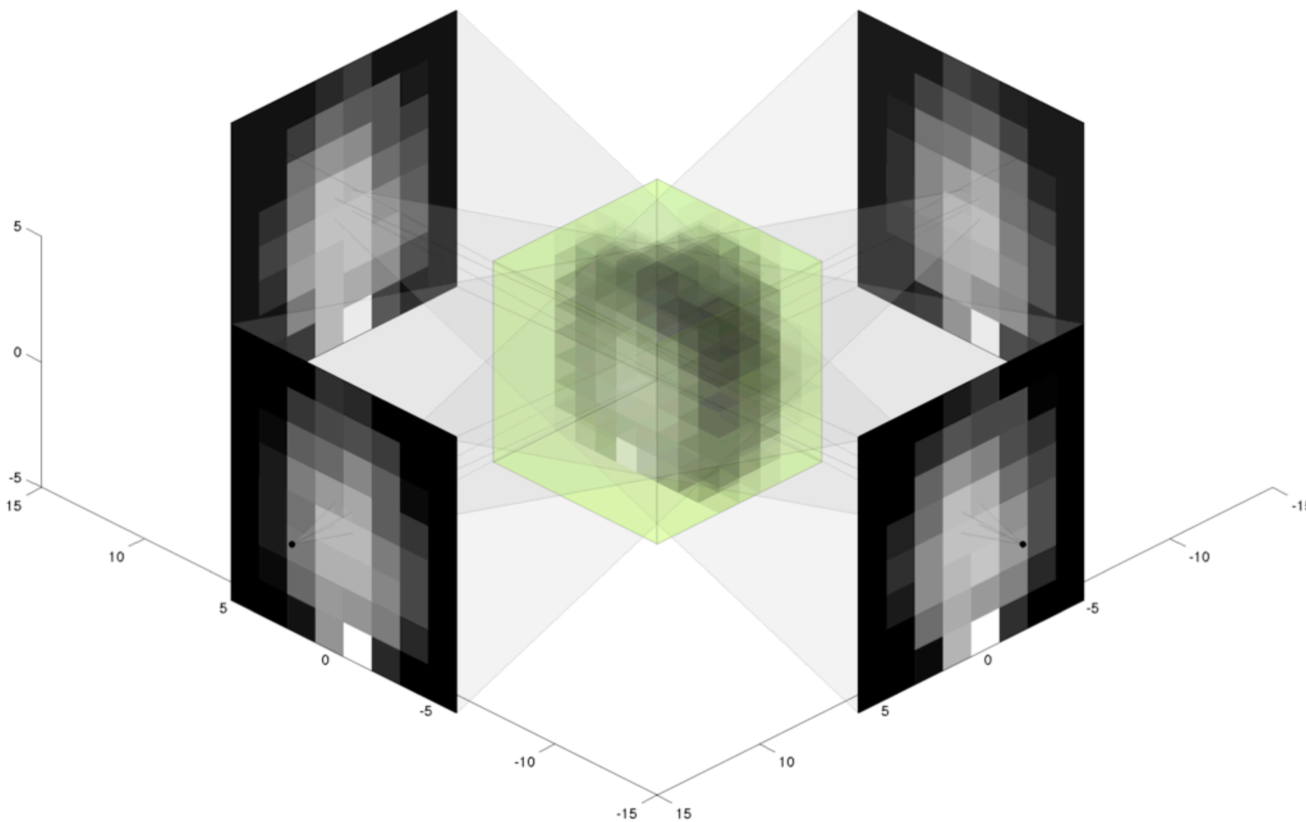
SIRT

- Forward projections ($\mathbf{A}x$)



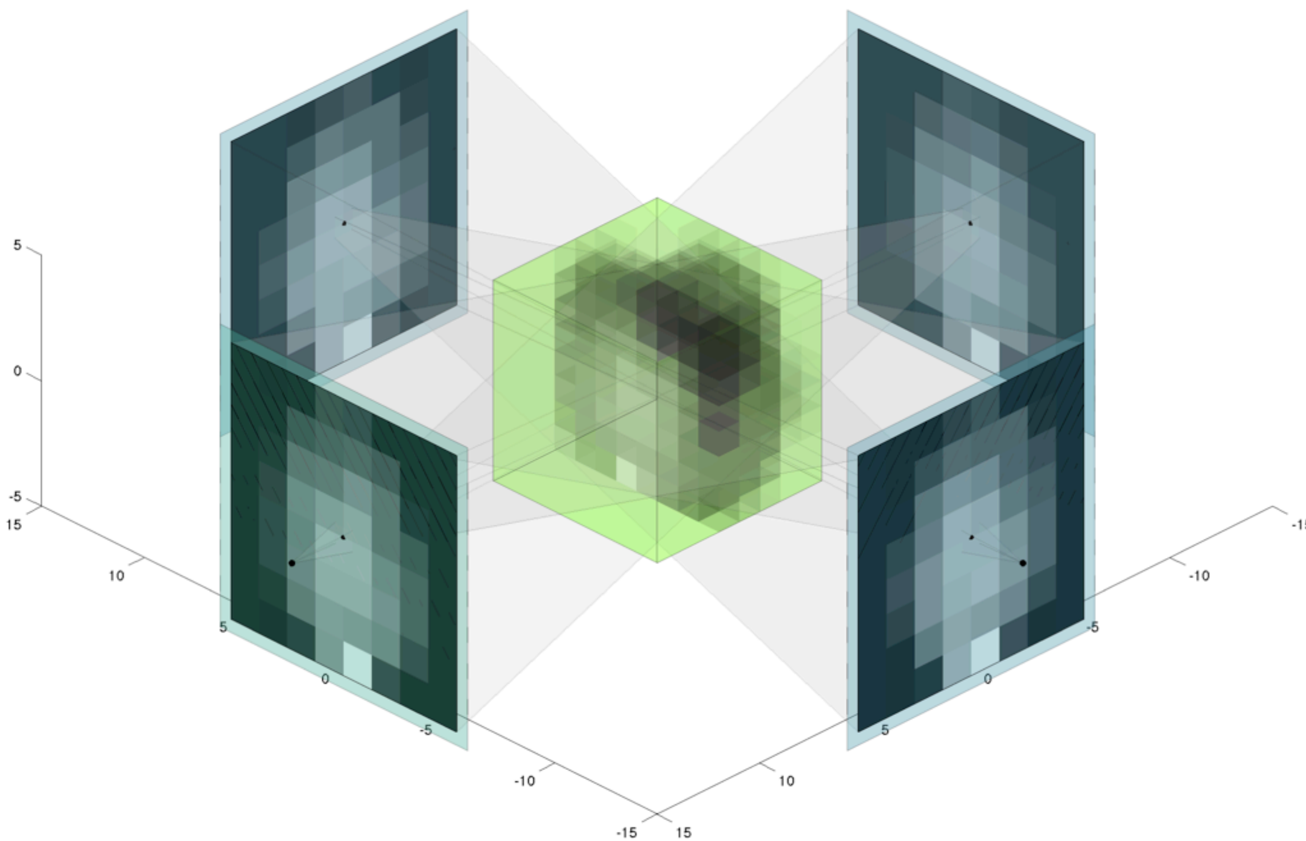
SIRT

- Correction ($y = (b\text{-}Ax)\ ./\mathrm{diag}(AA^T)$)



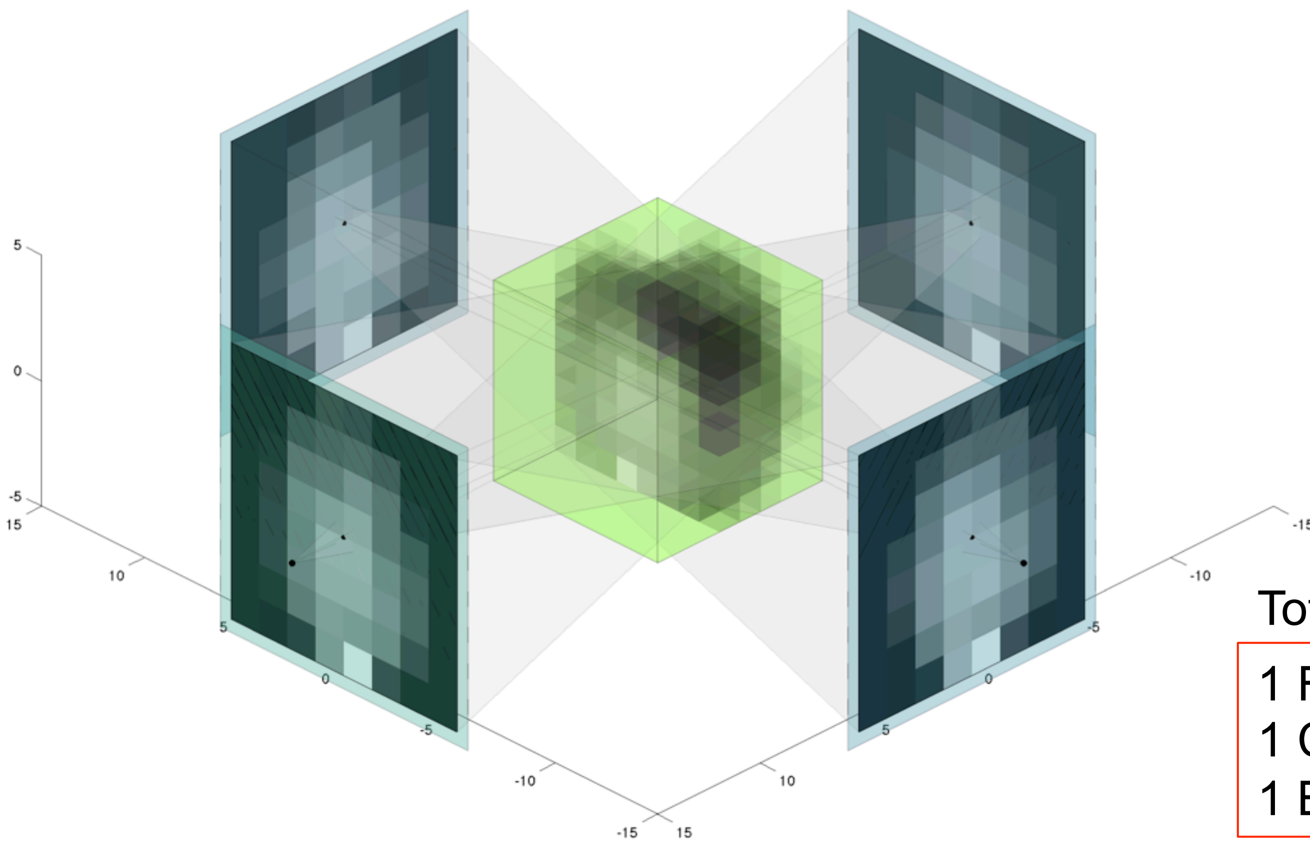
SIRT

- Back projection ($x := x + \lambda A^T y$)



SIRT

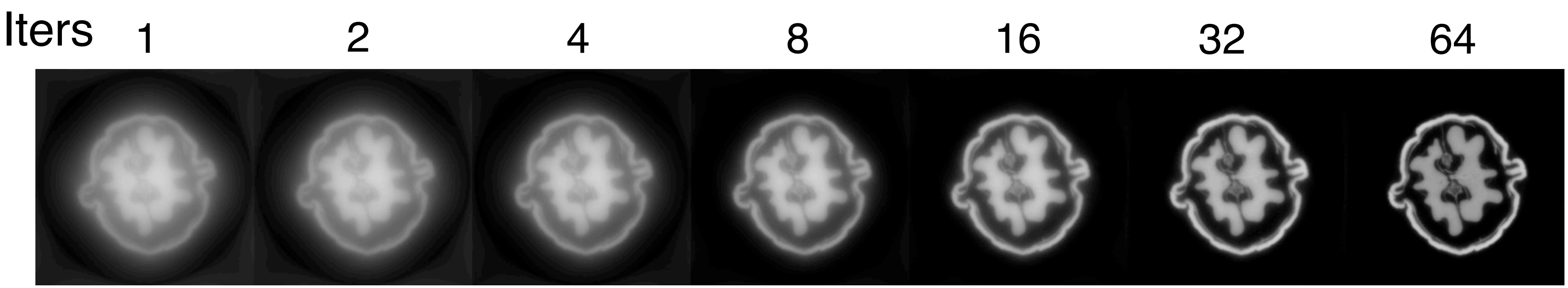
- Back projection $(x := x + \lambda A^T y)$



Total:

1 FP x 4
1 Correction x 4
1 BP x 4

SIRT reconstruction in ASTRA

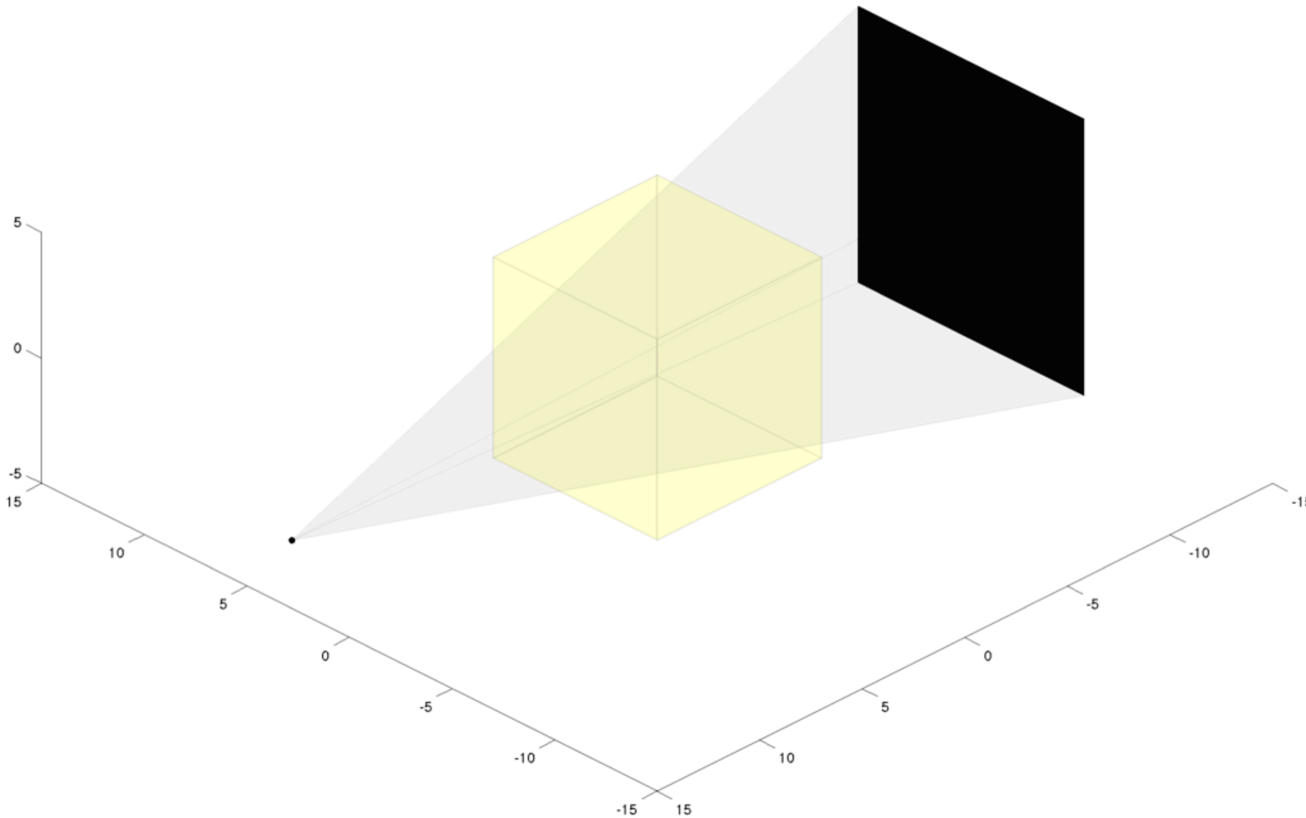



```
% Set up a GPU reconstruction
cfg = astra_struct('SIRT3D_CUDA');
cfg.ReconstructionDataId = rec_id;
cfg.ProjectionDataId = proj_id;
cfg.option.GPUindex = 0;
% Create the algorithm object
alg_id = astra_mex_algorithm('create', cfg);
% Run 64 SIRT iterations
astra_mex_algorithm('iterate', alg_id, 64);
% Get the result
rec = astra_mex_data3d('get', rec_id);
```

Block Methods

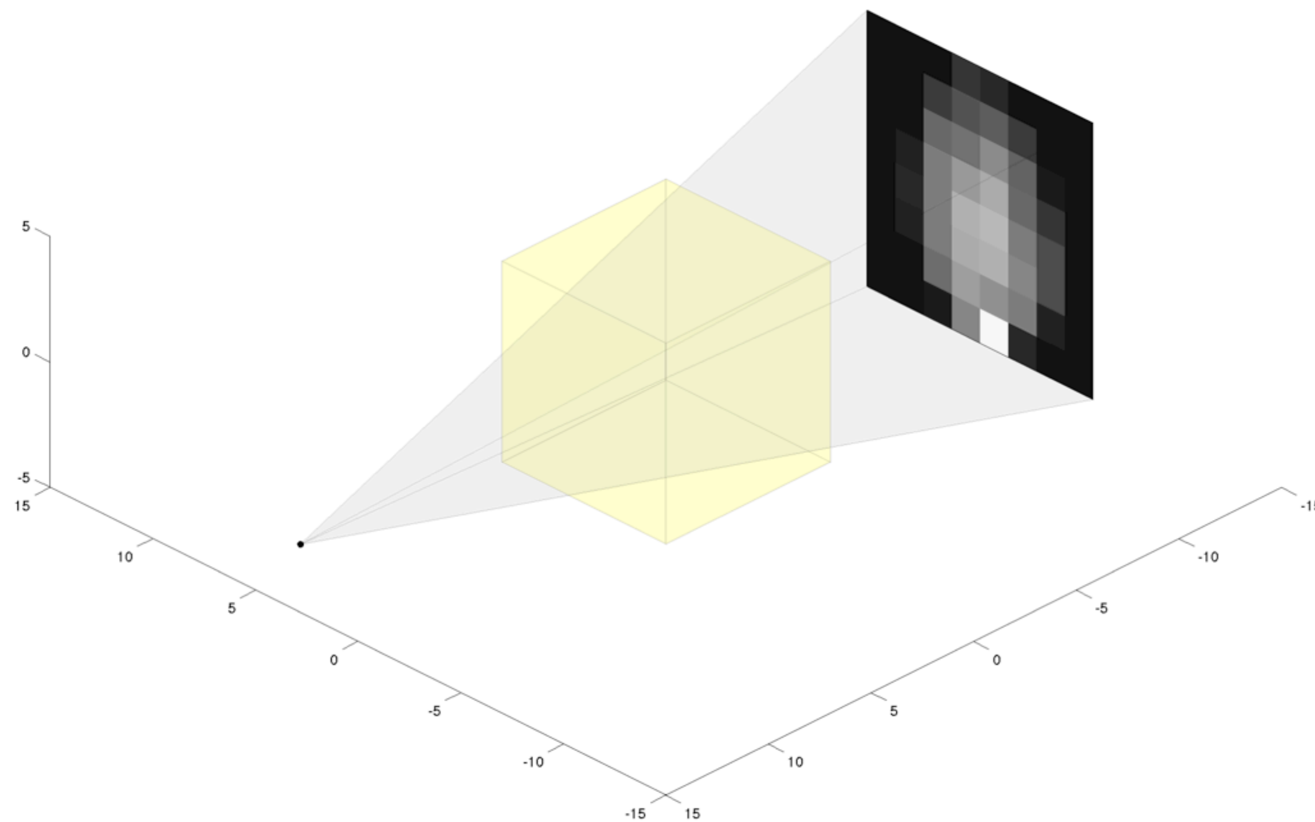
Block method (one per projection)

- Forward projection (A_0x)



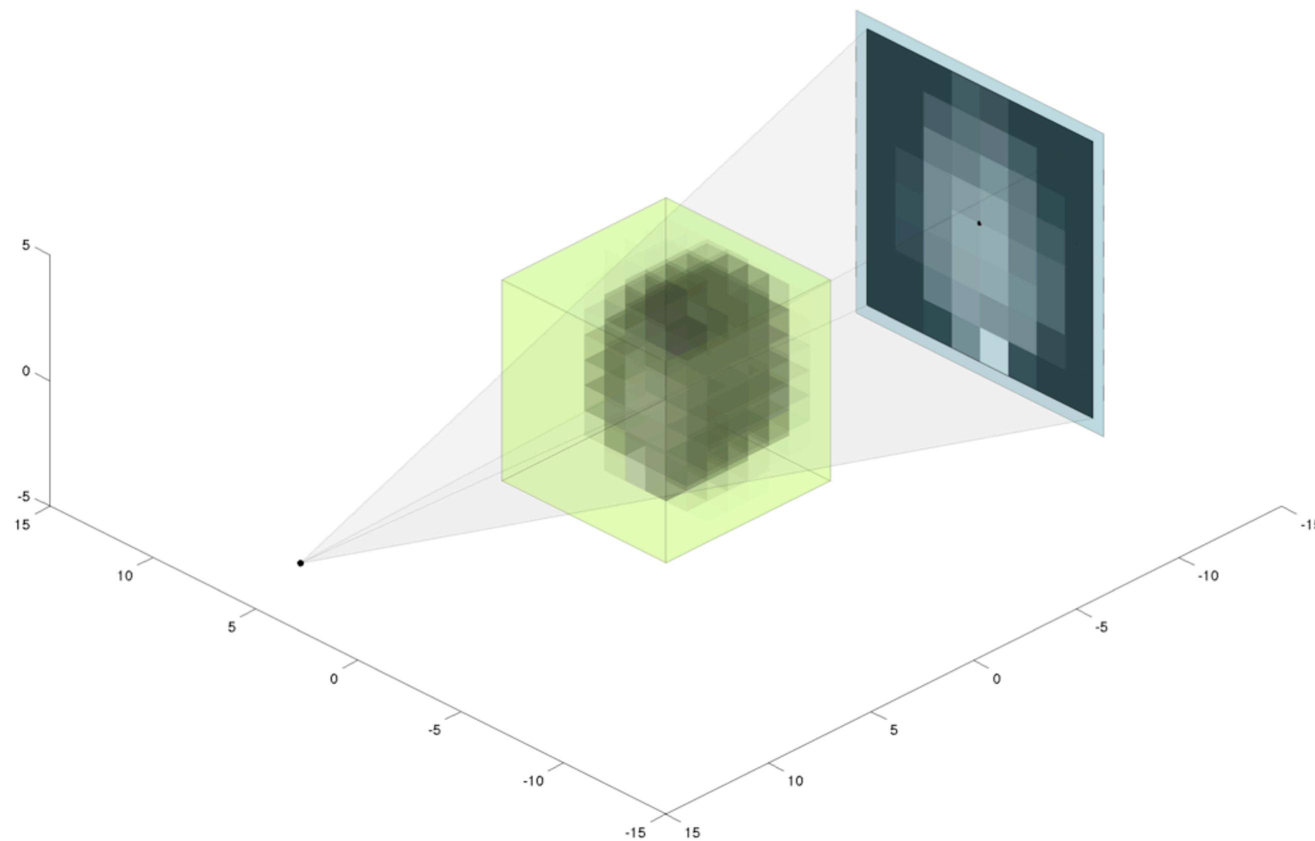
Block method (one per projection)

- Correction ($y = (b-A_0x)$ ./diag($A_0A_0^T$))



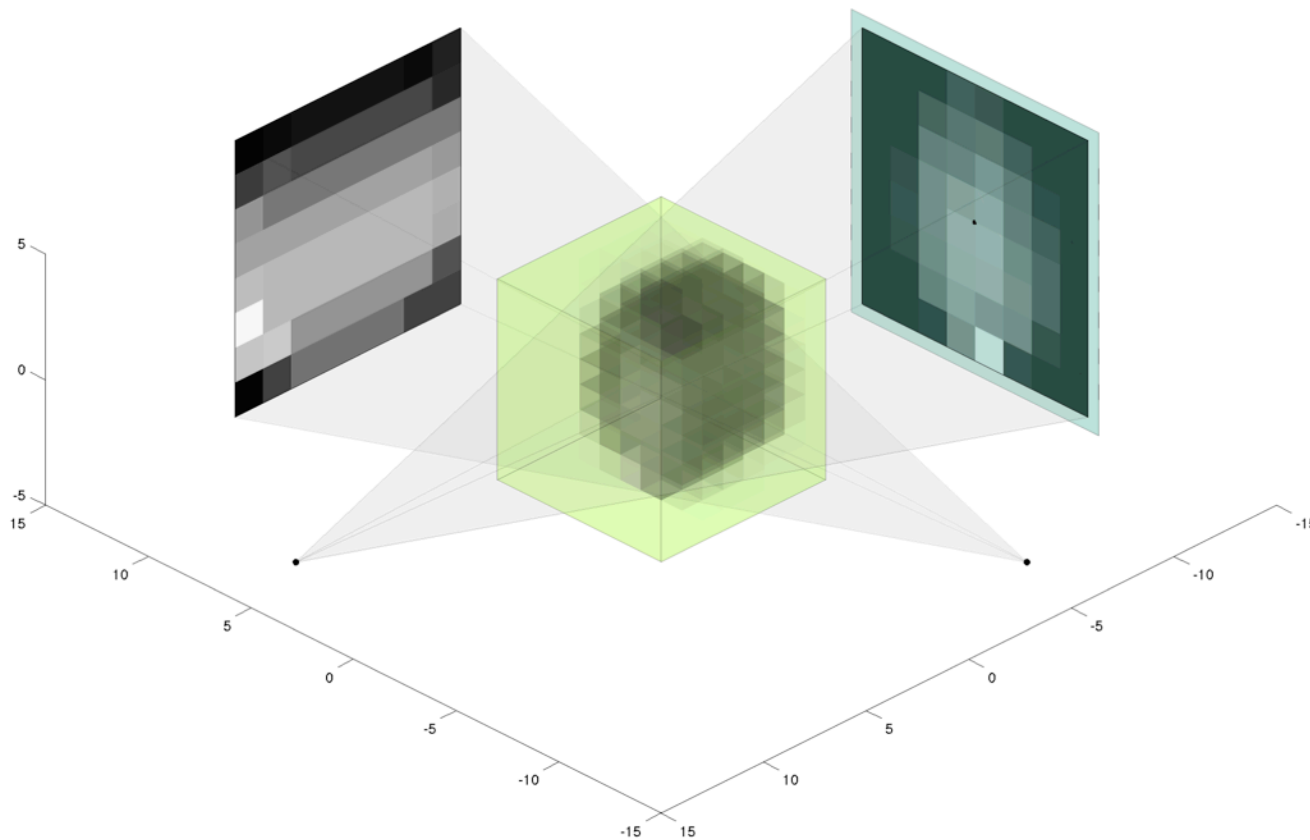
Block method (one per projection)

- Back projection ($x := x + \lambda A_0^T y$)



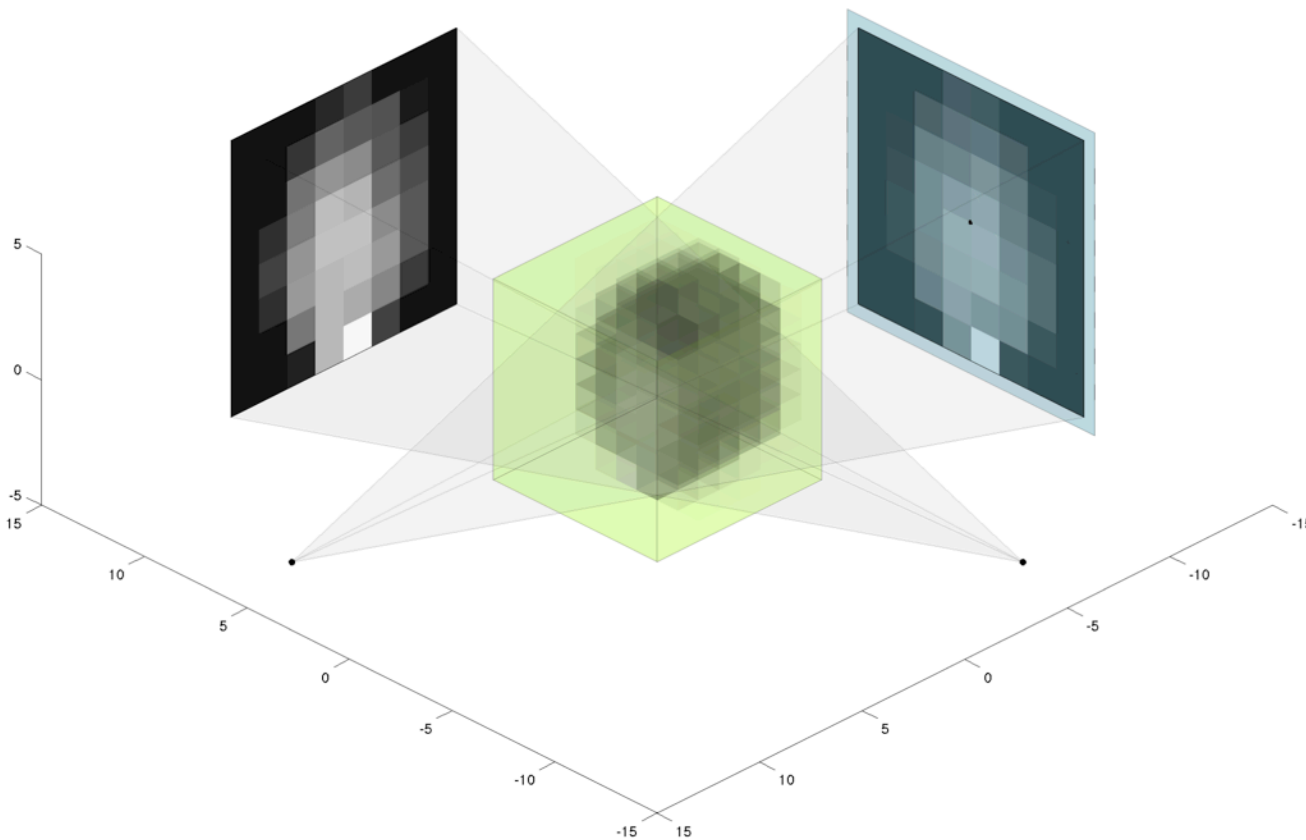
Block method (one per projection)

- Forward projection ($A_1 x$)



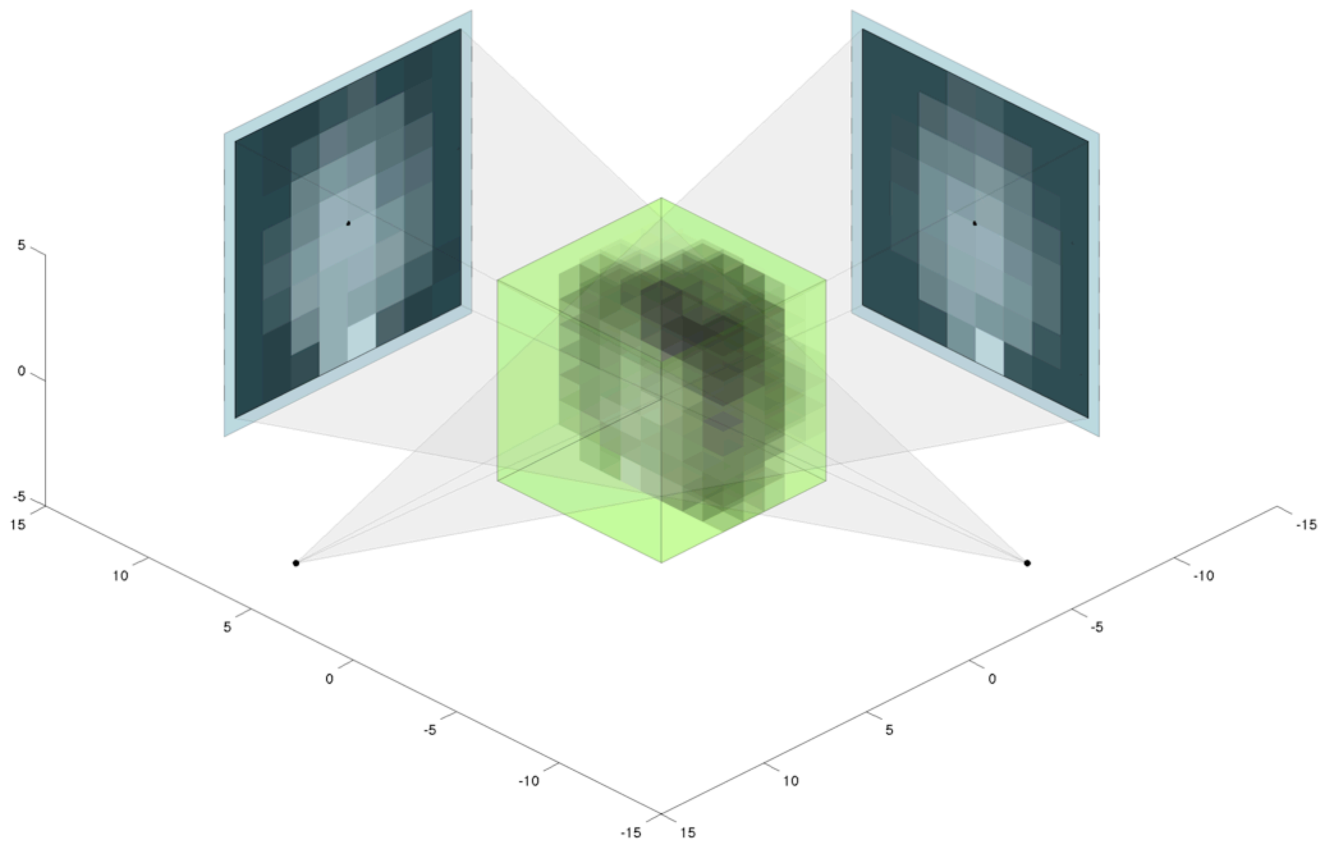
Block method (one per projection)

- Correction ($y = (b - A_1 x) ./\mathrm{diag}(A_1 A_1^T)$)



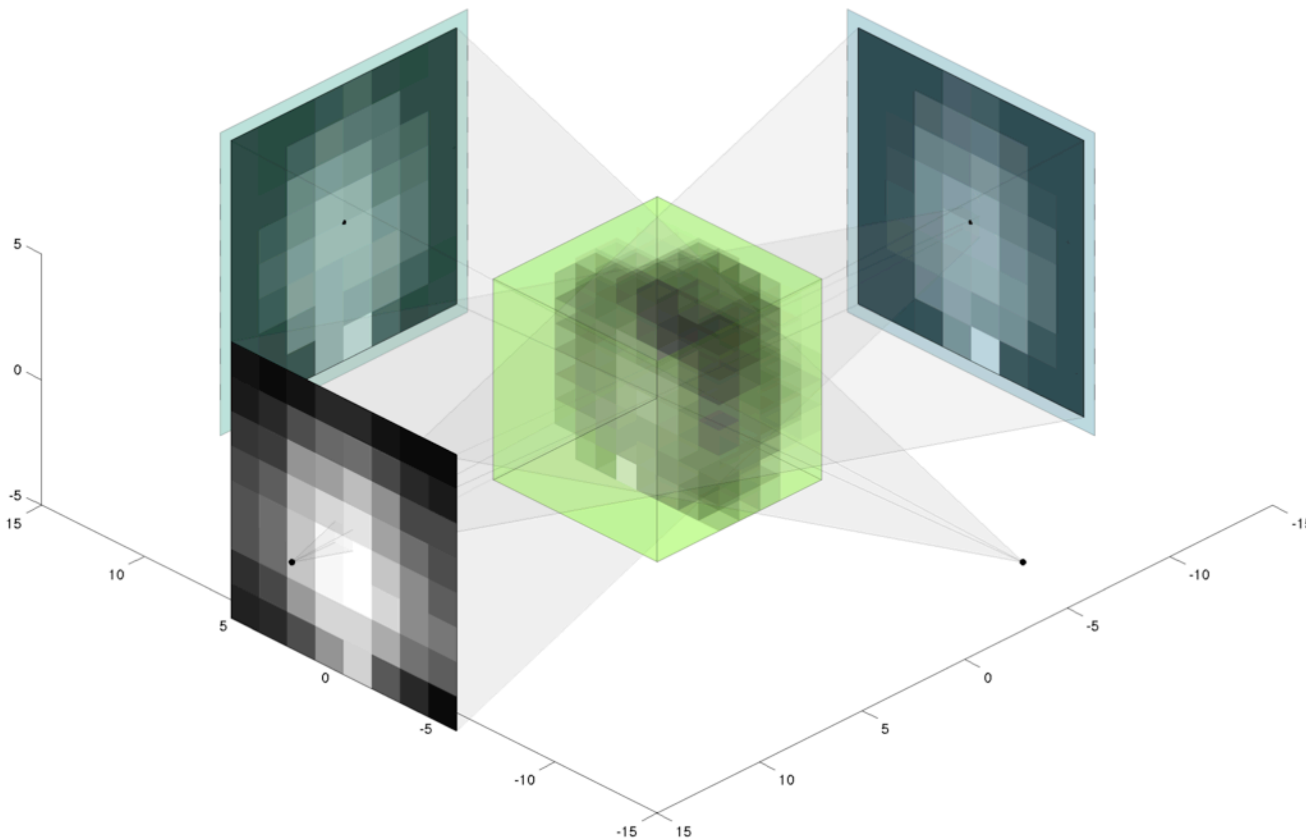
Block method (one per projection)

- Back projection ($x := x + \lambda A_1^T y$)



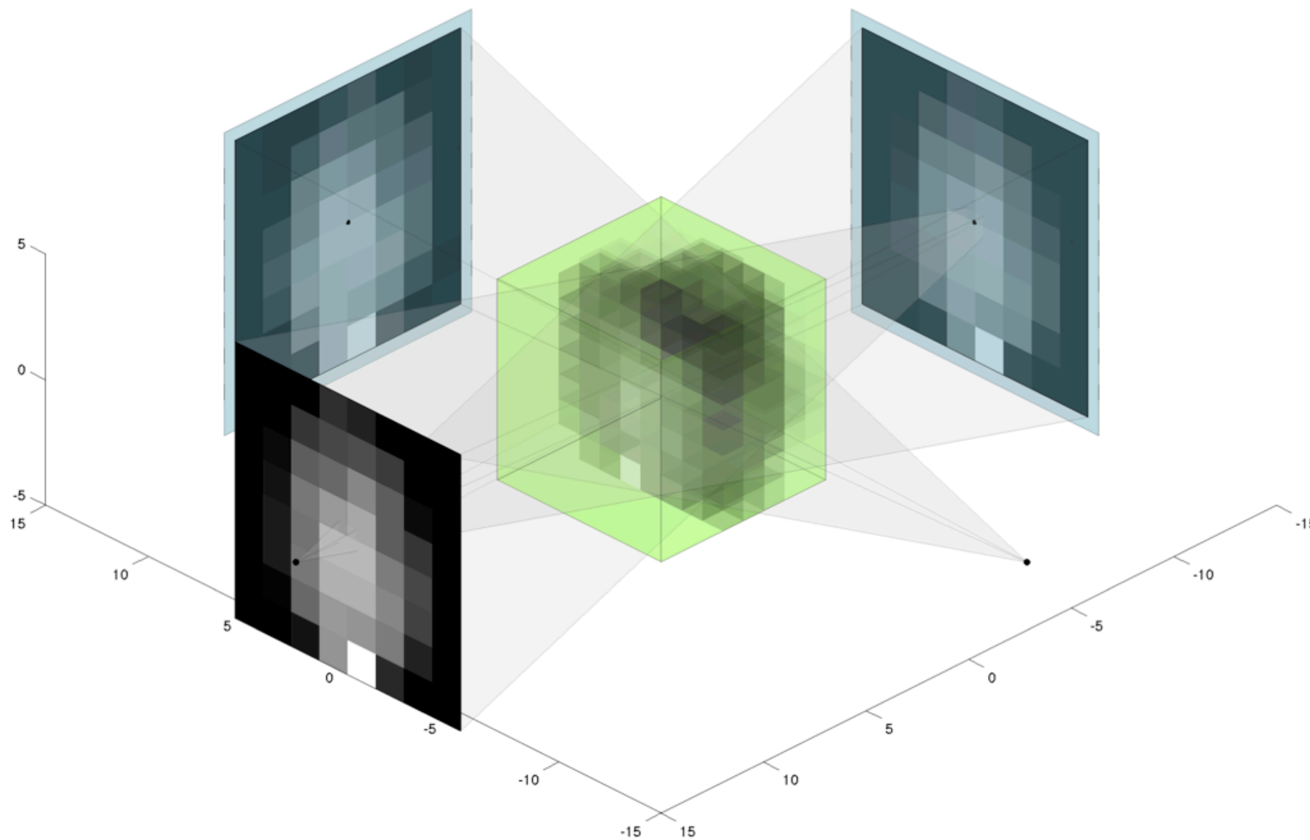
Block method (one per projection)

- Forward projection (A_2x)



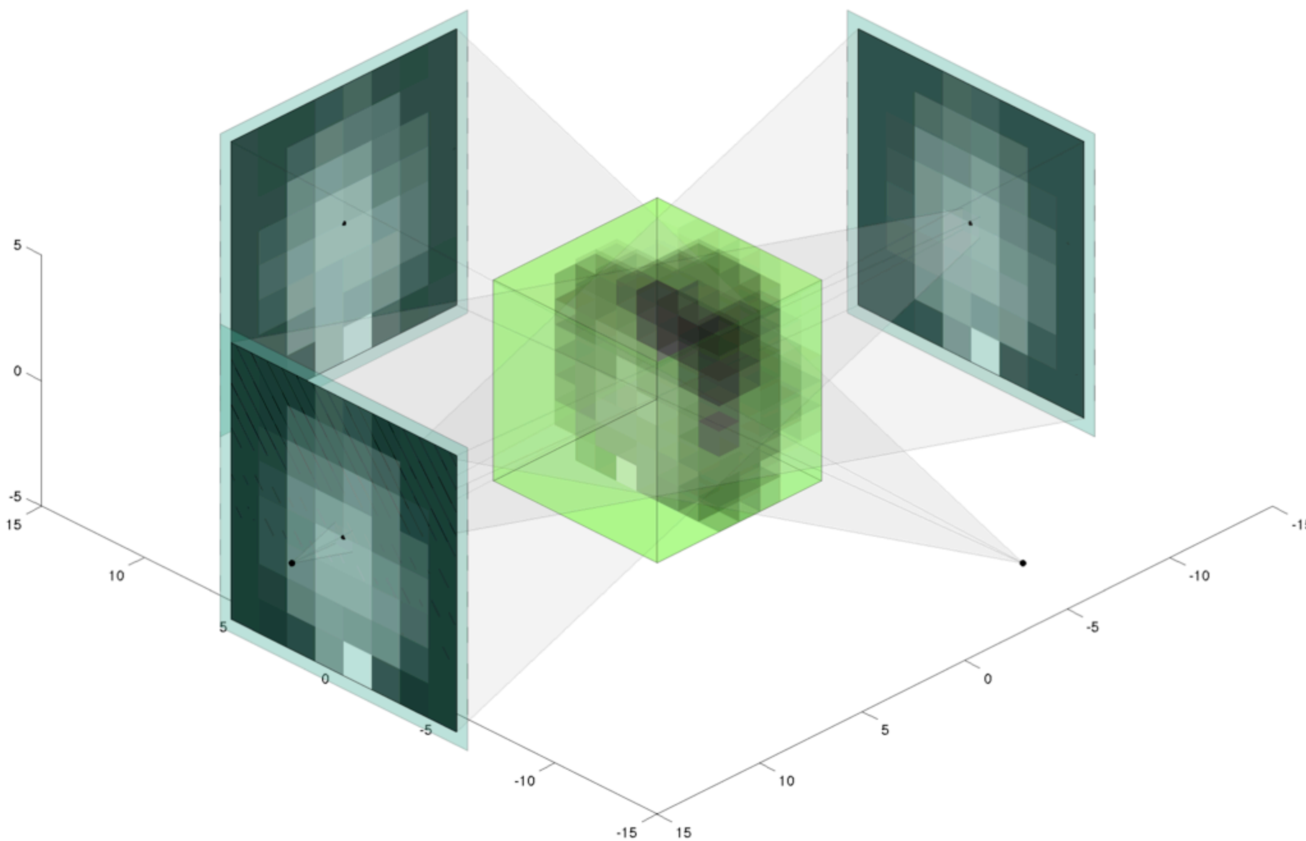
Block method (one per projection)

- Correction ($y = (b\text{-}A_2x)\ ./\mathrm{diag}(A_2A_2^T)$)



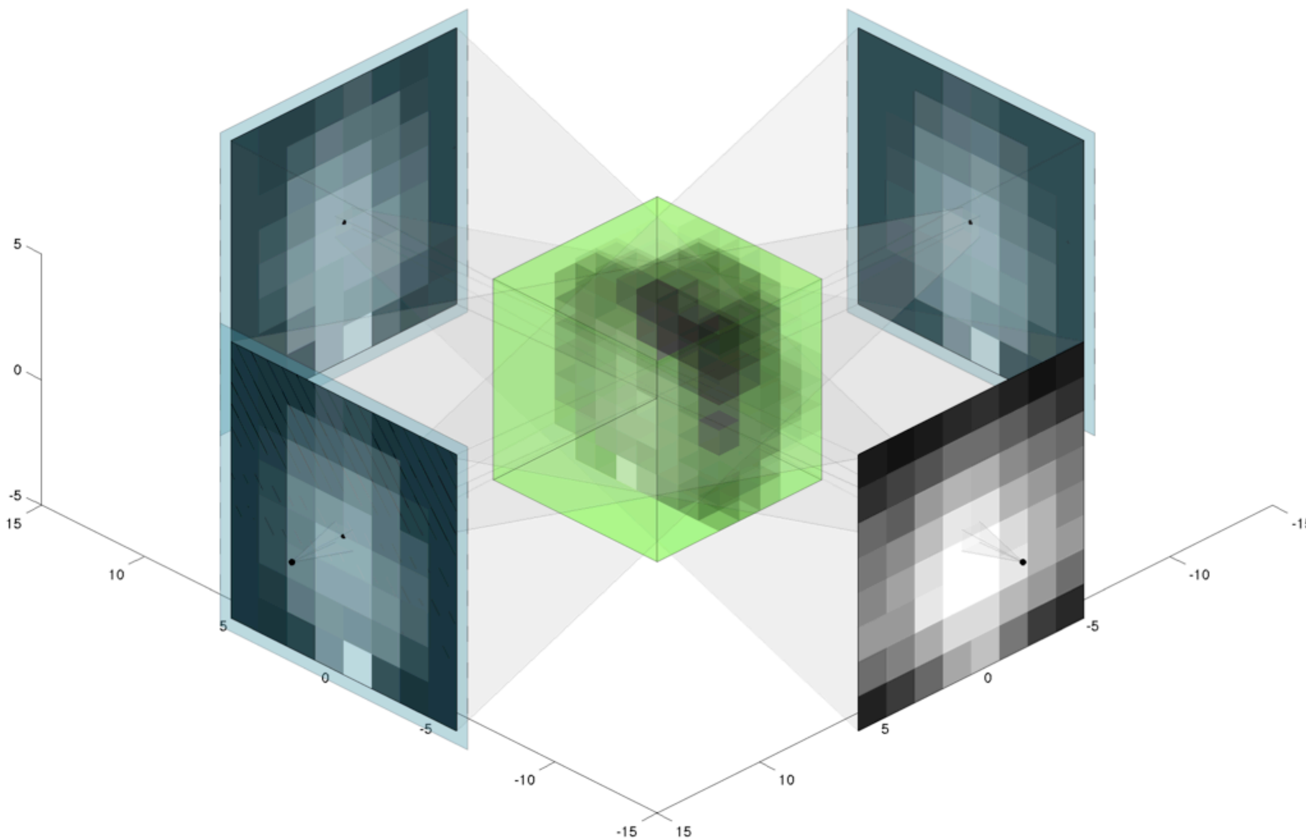
Block method (one per projection)

- Back projection ($x := x + \lambda A_2{}^T y$)



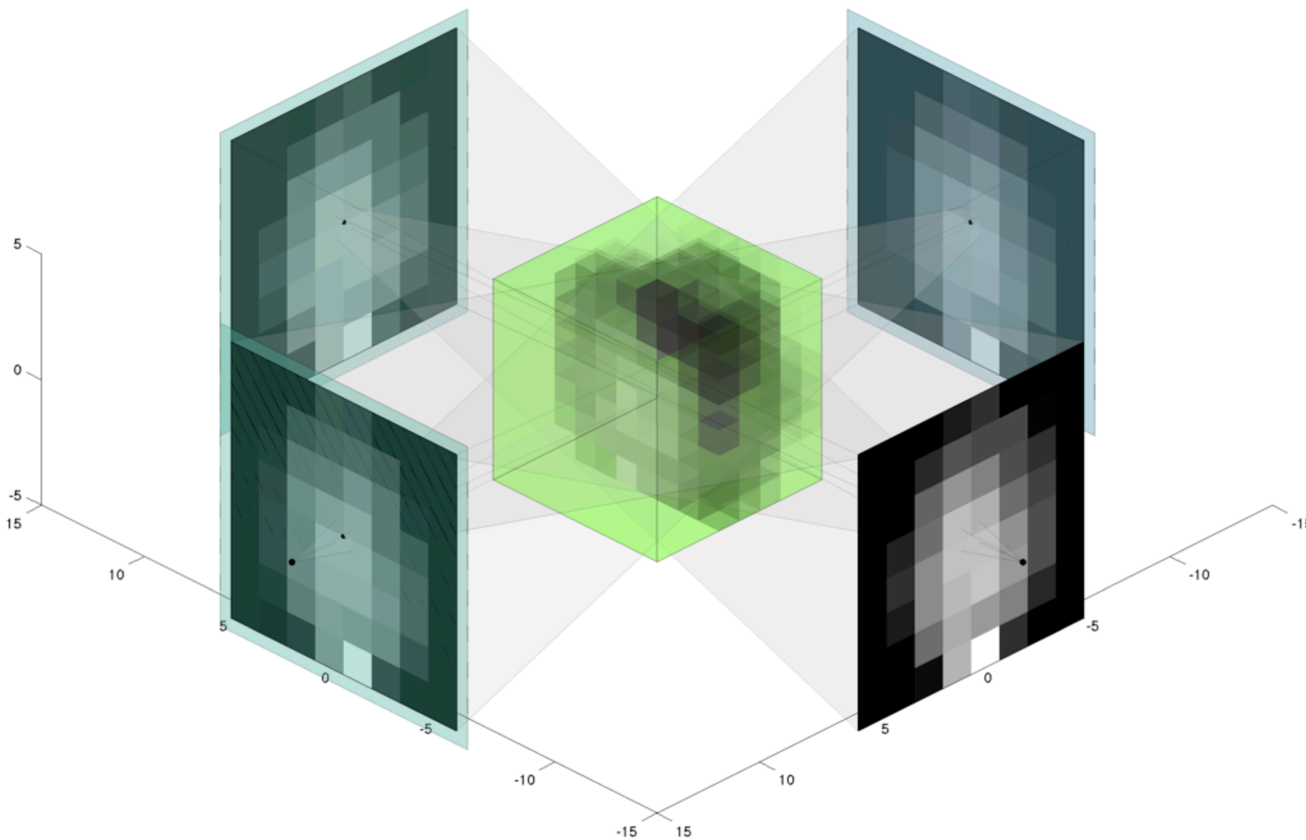
Block method (one per projection)

- Forward projection (A_3x)



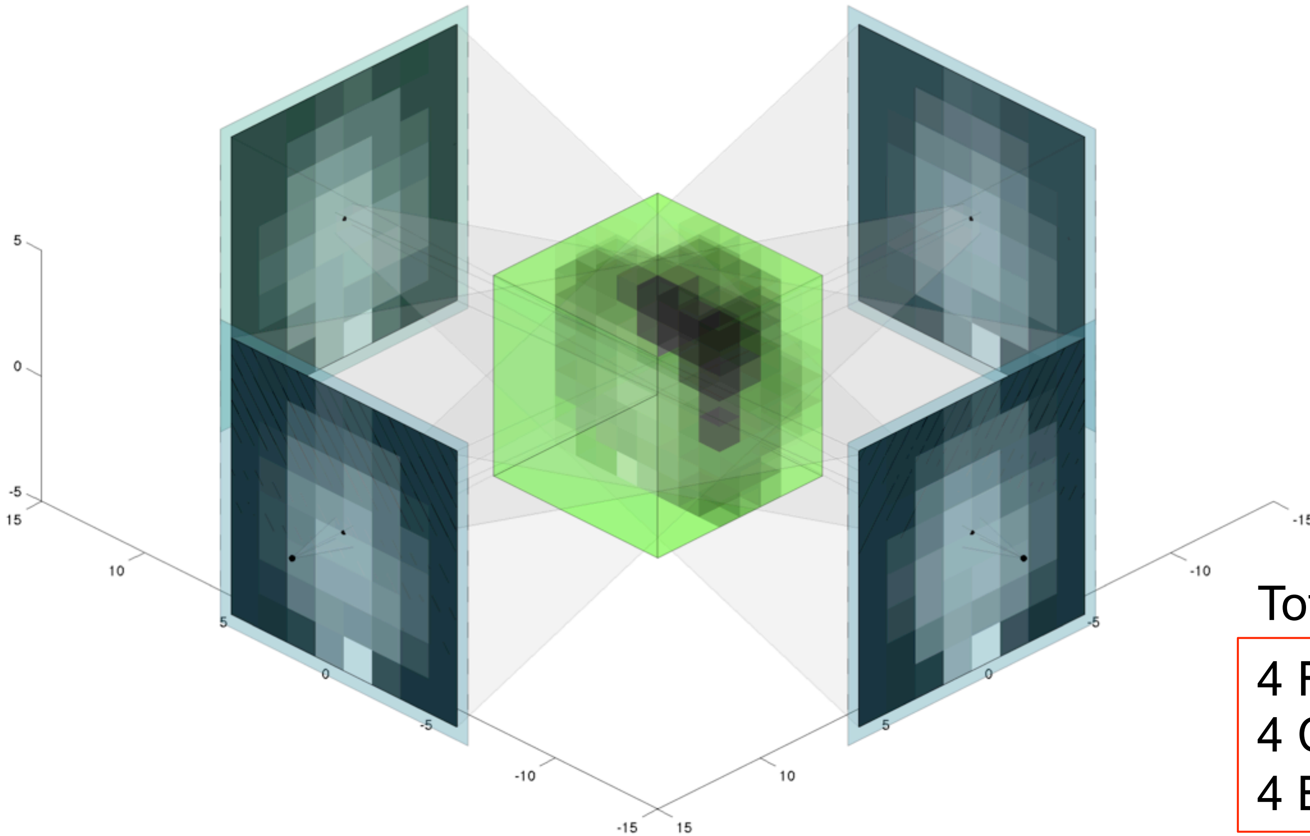
Block method (one per projection)

- Correction $(y = (b - A_3 x)\ ./\mathrm{diag}(A_3 A_3^T))$



Block method (one per projection)

- Back projection ($x := x + \lambda A_3{}^T y$)



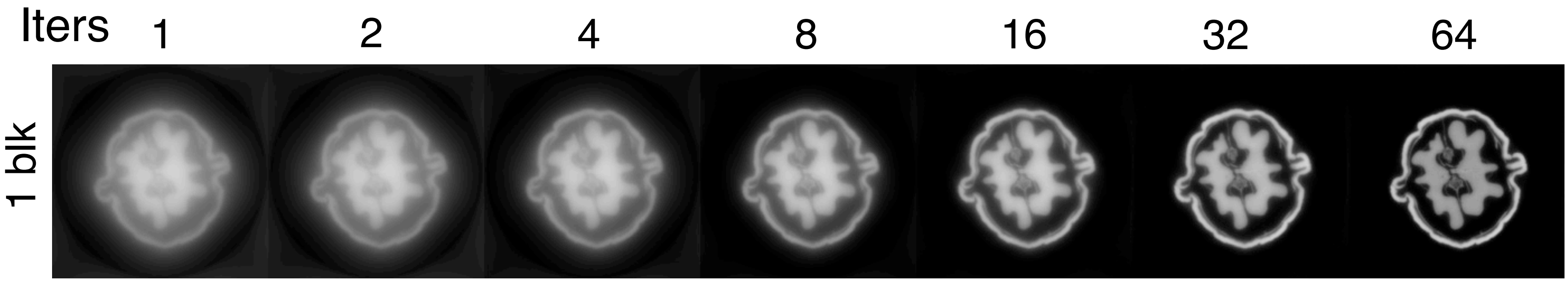
Total:

4 FP
4 Corrections
4 BP

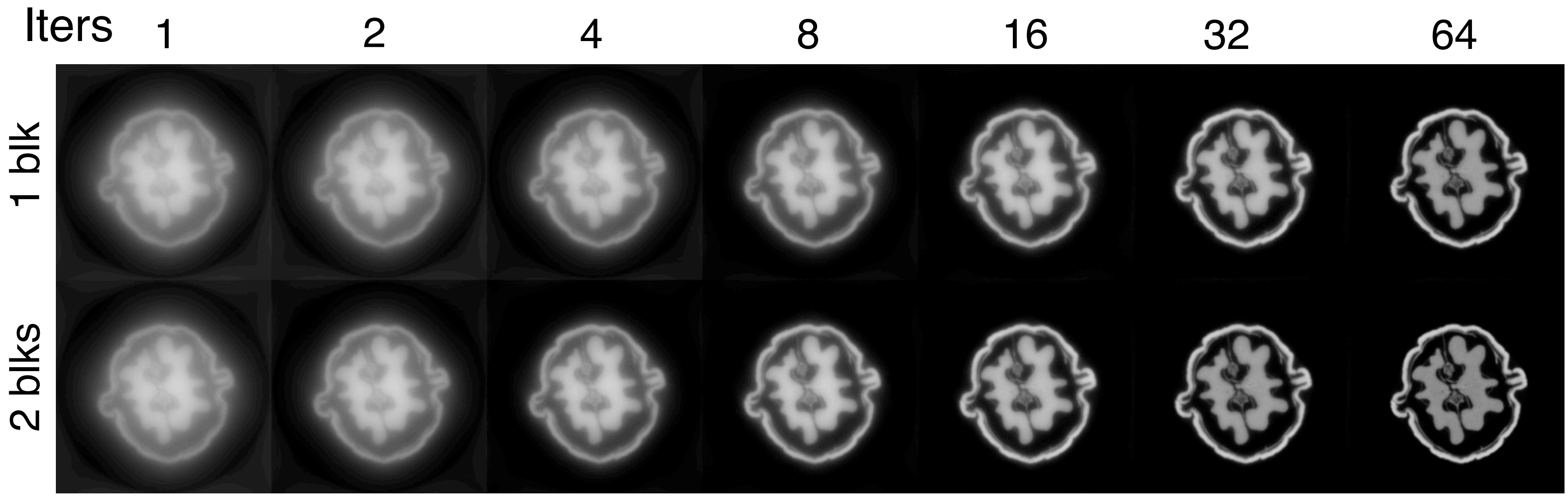
Block method using ASTRA

```
...
for iter = 1:num_iter
    for blk = 1:num_blks
        % Create the algorithm object
        alg_id = astra_mex_algorithm('create', cfg{blk});
        % Run 1 SIRT iteration with this block
        astra_mex_algorithm('iterate', alg_id, 1);
        % Delete algorithm object to conserve memory
        astra_mex_algorithm('delete', alg_id);
    end
end
% Get the result
rec = astra_mex_data3d('get', rec_id);
```

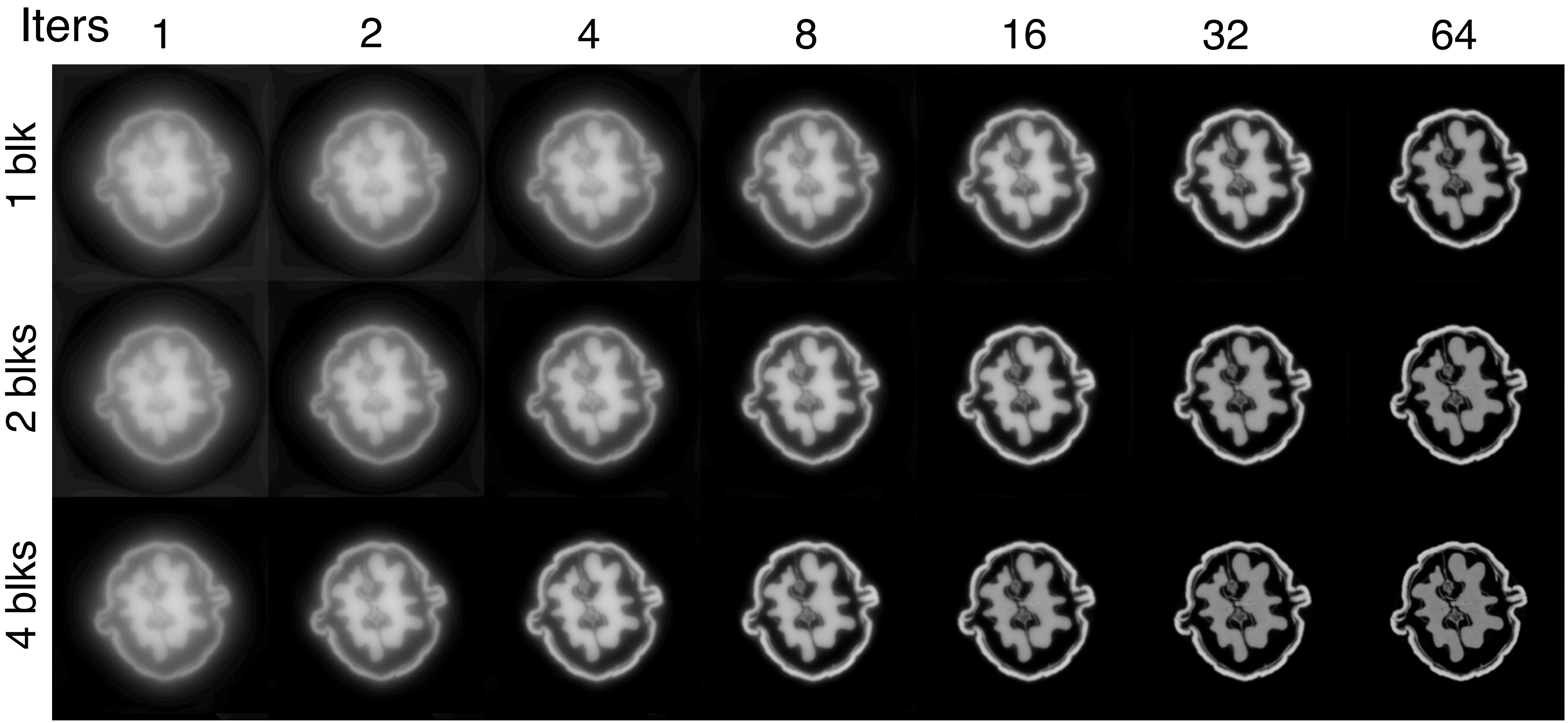
Block method reconstruction



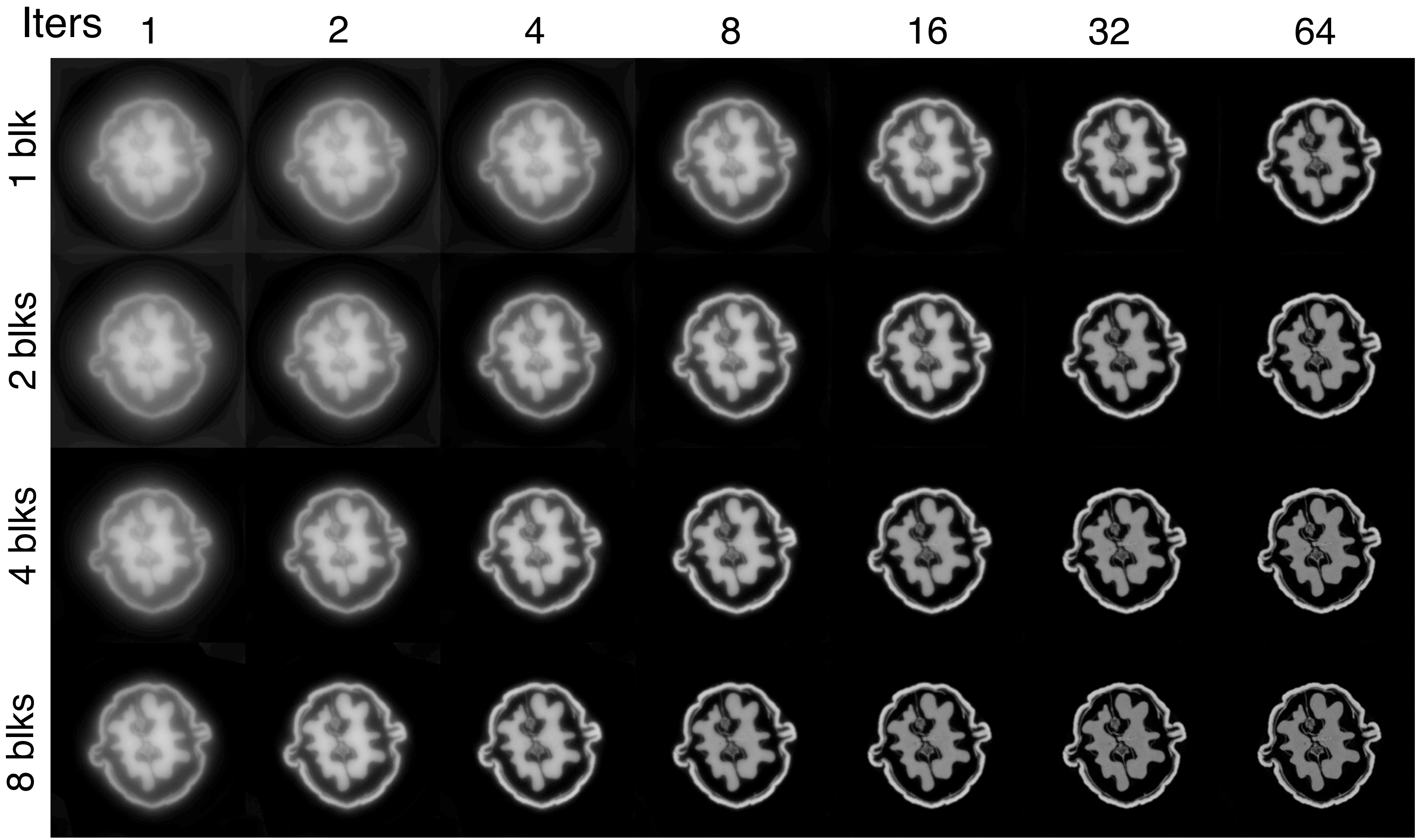
Block method reconstruction



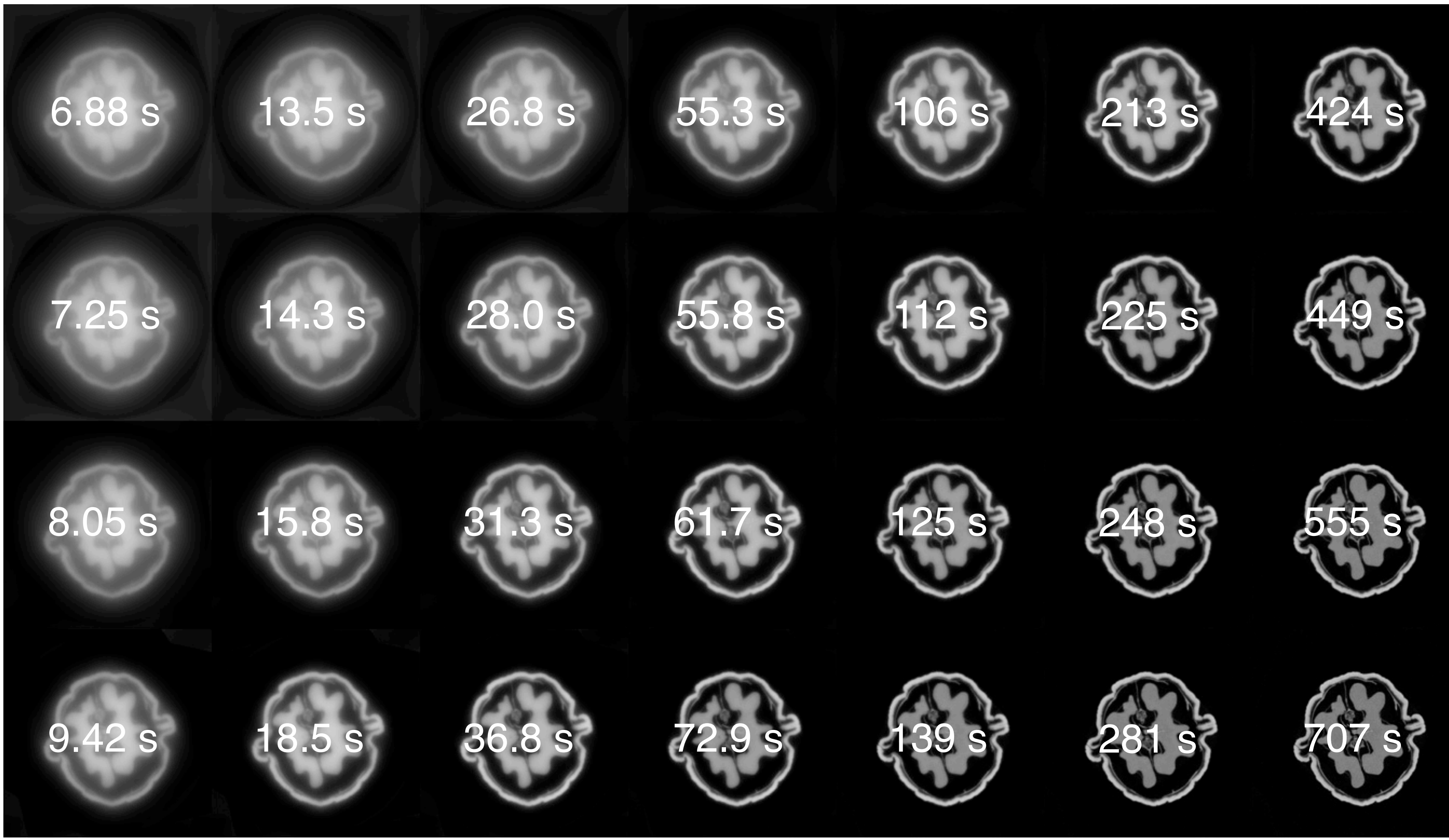
Block method reconstruction



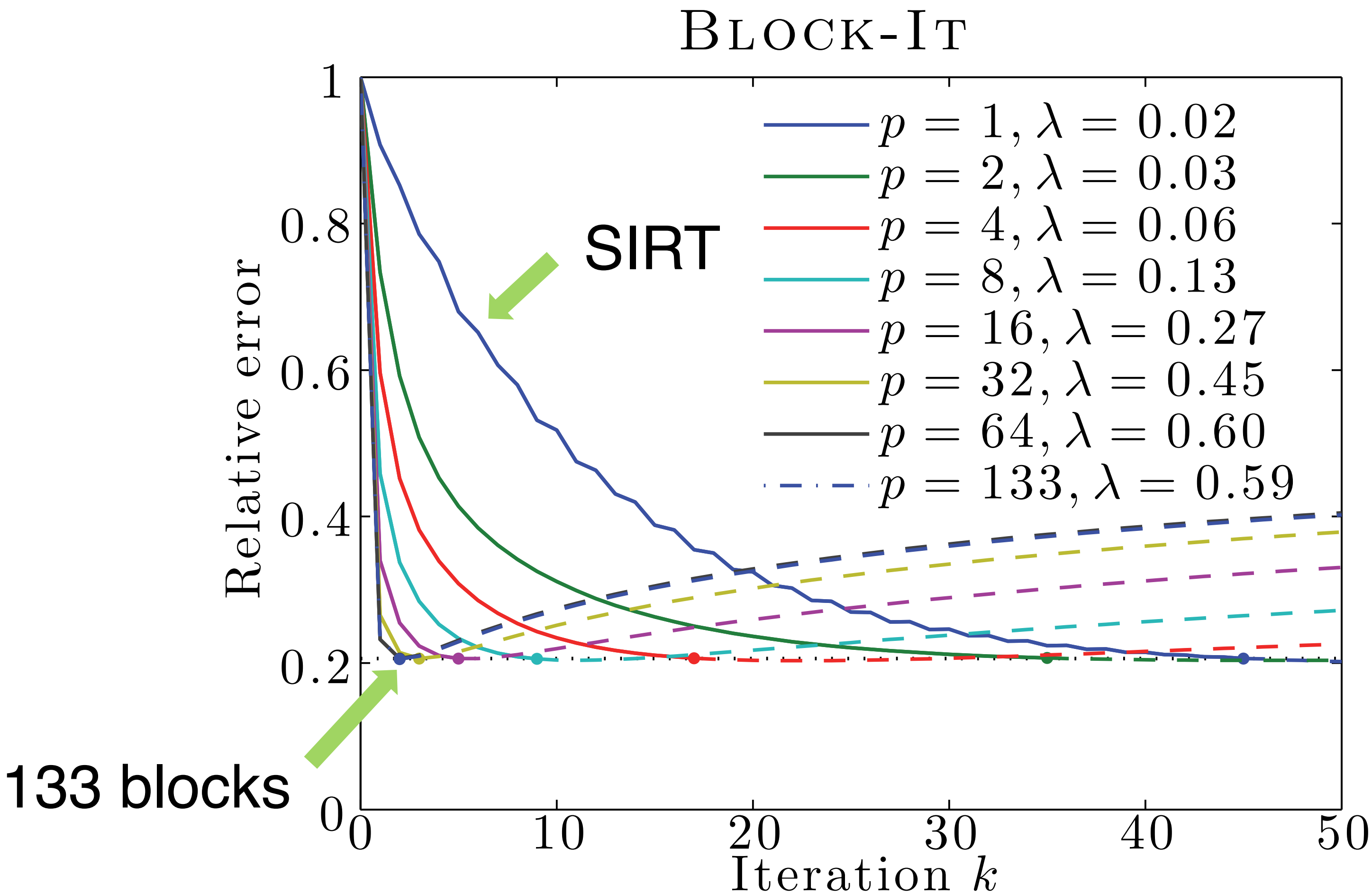
Block method reconstruction



Block method reconstruction



Block methods converge faster



Fully optimized block code

1 iter | 2 iters | 3 iters

400 blks

GPU	GTX 680	K20	K40	K80(x1)	TITAN X
Runtime (s)	47.0	25.1	20.7	22.8	10.5

- <1 min is ok! - but what about larger problems?

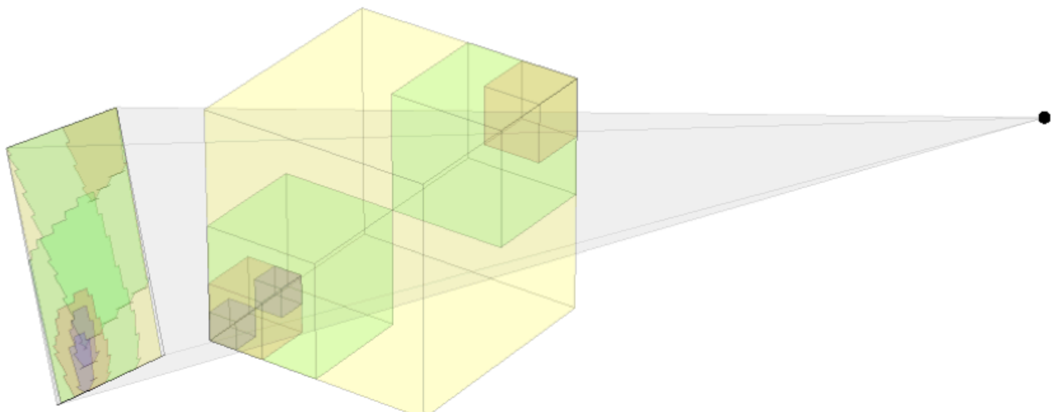
Distributed / Multi-GPU computing

Domain decomposition

```
Domains:                                             Detector:       Angles
                           512
            +-------------------------------+                    +       +-----
           /                               /|                   /|        \ 400
          /                               / |                  / |         \
         /                               /  |                 /  |          \
    512 /               3               /   +            512 /   |
       /                               /   /|               /    |
      /                               / 3 / |              /     |
     /                               /   /  |             /      |
    +-------------------------------+   /   + 512        +       | 512
    |                               |  /   /|            |       |
128 |               3               | / 2 / |            |       |
    |                               |/   /  |            |       |
    +-------------------------------+   /   +            |       |
    |                               |  /   /|            |       |
128 |               2               | / 1 / |            |       |
    |                               |/   /  |            |       |
    +-------------------------------+   /   +        512 |       +
    |                               |  /   /             |      /
128 |               1               | / 0 /              |     /
    |                               |/   /               |    /
    +-------------------------------+   / 512            |   / 512
    |                               |  /                 |  /
128 |               0               | /                  | /
    |                               |/                   |/
    +-------------------------------+                    +
                   512
```

Communication

- Setting up for distributed/multi-GPU computing
  - Top level domain decomposition of the solution x
  - One domain for each MPI thread / GPU



- We communicate the FP once per block
  - Note that only part of the FP is required from others

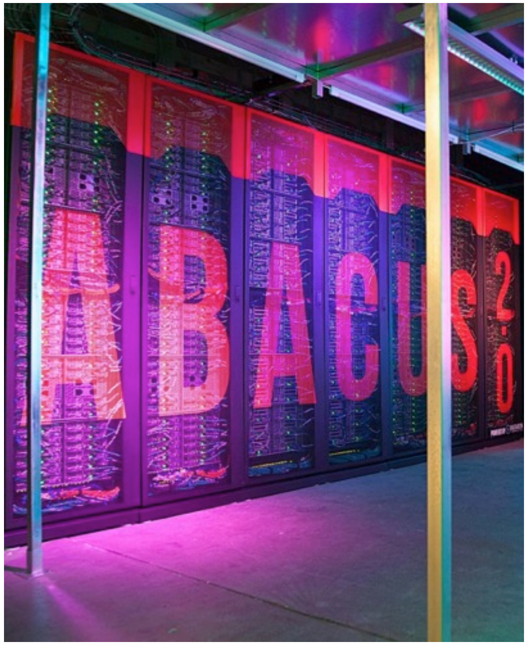
Multi-GPU results (K80)

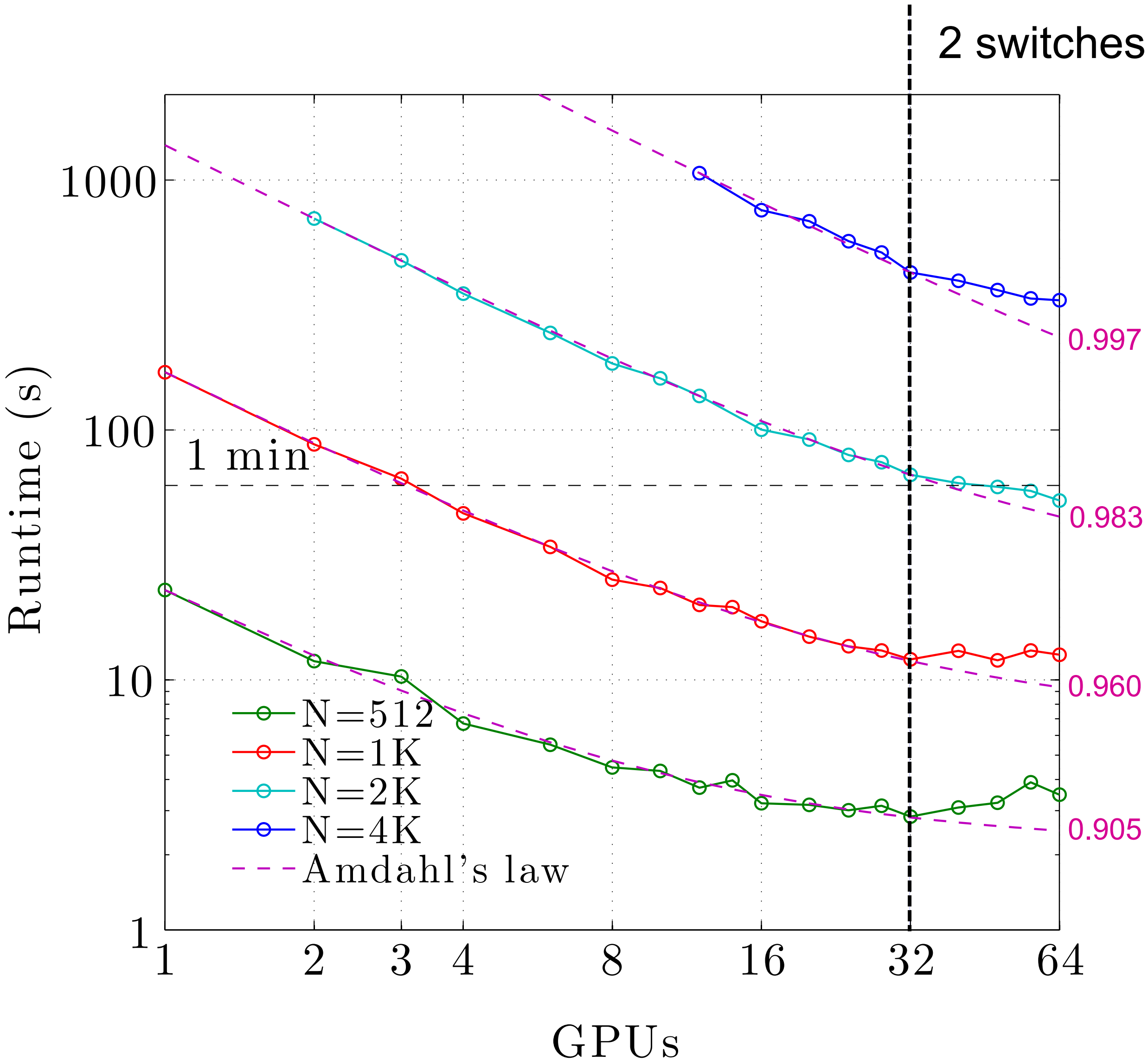
N	256	512	1024	2048
1 GPU	3.33	22.8	181	-
2 GPU	2.31	12.6	95.6	798
4 GPU	2.08	7.68	51.1	398
8 GPU	2.00	4.65	26.4	211

Walnut case - full reconstruction time (2 iters) without disk I/O in seconds.

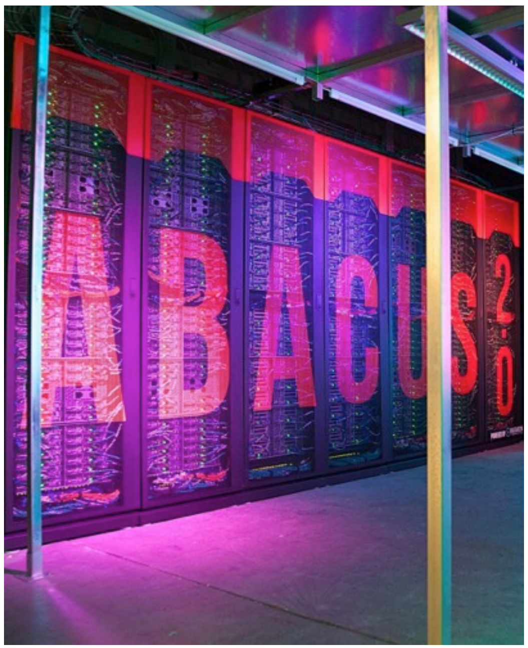
Solution size: N^3 / Projections: $400 \times N^2$

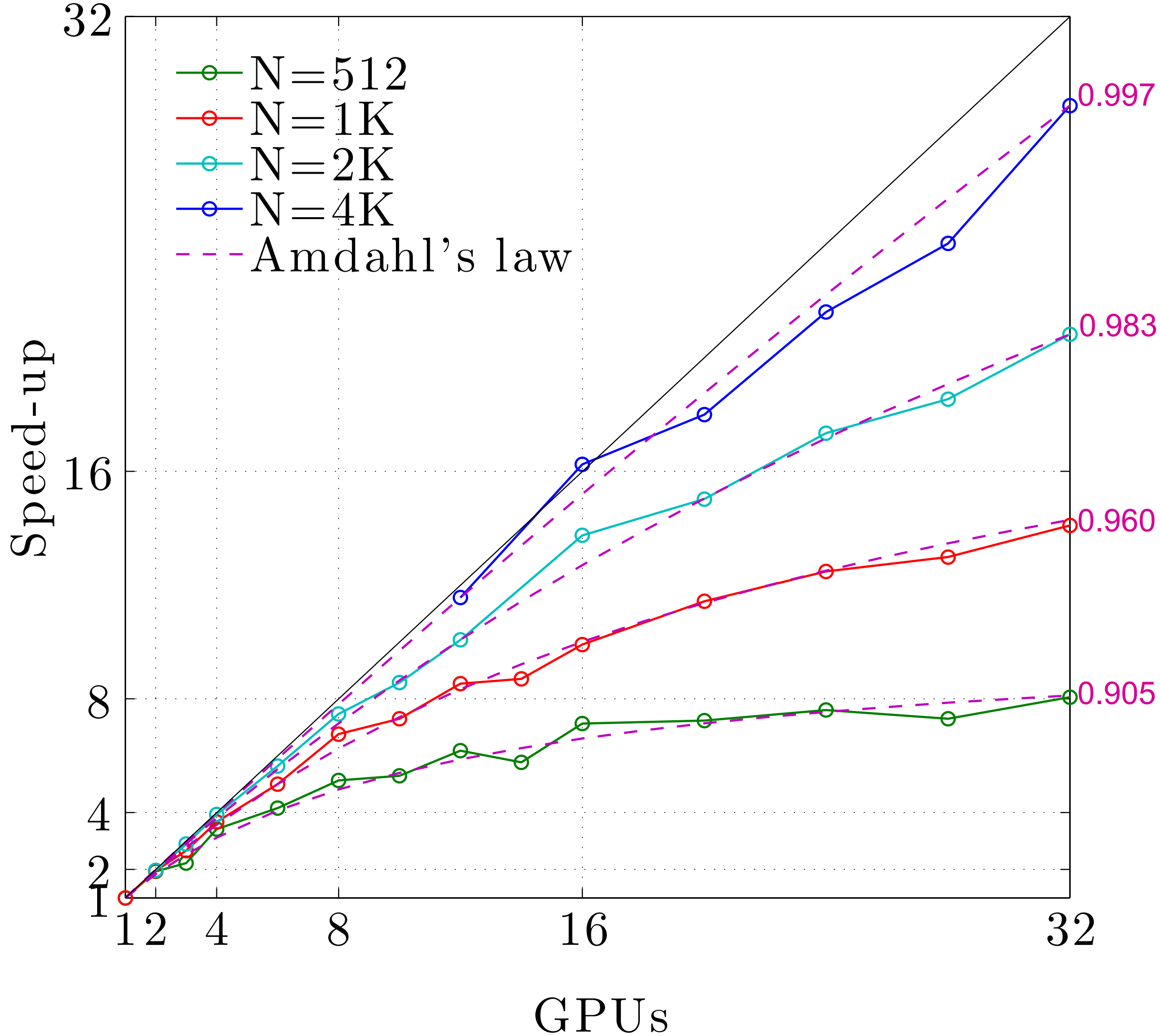
GPU cluster results





GPU cluster results





CPU cluster results (Xeon 2680)

N	256	512	1024	2048
20 cores	5.54	38.7	316	2380
40 cores	3.39	21.8	171	1260
80 cores	1.94	13.3	97,3	700
160 cores	1.36	9.55	60.1	413
320 cores	-	8.01	46.5	314

Walnut case - full reconstruction time (2 iters) without disk I/O in seconds.

Solution size: N^3 / Projections: $400xN^2$

Summary

- Block methods are convenient for large-scale reconstructions where runtimes are an issue
- ASTRA allows you to do block methods with little effort and good performance for limited blocks
- Block methods are well suited for GPUs & CPUs
  - We have an implementation with one projection per block that performs close to SIRT runtimes
- Multi-GPU approach is also efficient by domain decomposition and communication of the FP
- GPU-cluster performance scales but operates at the limit of current communication bandwidths

Summary

Thank you for your attention!

Block method

- Formulated in terms of matrices (as yesterday)

Basic Block-Iteration algorithm

$x^{(0)}$ = initial vector
for $k = 0, 1, 2, \ldots$
$\quad \boldsymbol{x}^{(k,0)} = \boldsymbol{x}^{(k)}$
$\quad$ for $l = 0, 1, 2, \ldots, \mathrm{blks}-1$
$$\boldsymbol{x}^{(k,l+1)} = P_{\mathcal{C}}\left(\boldsymbol{x}^{(k,l)} + \lambda_k \boldsymbol{D}^{-1} \boldsymbol{A}_l^T \boldsymbol{M}_l^{-1} (\boldsymbol{b}_l - \boldsymbol{A}_l \boldsymbol{x}^{(k,l)})\right)$$
$\quad$ end
$\quad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k,\mathrm{blks})}$
end