Numerical Linear Algebra Techniques for Fusion Applications

Ed D'Azevedo

Computer Science and Mathematics Division

OAK RIDGE NATIONAL LABORATORY

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Managed by UT Battelle, LLC under Contract No. DE-AC05-00OR22725
for the U.S. Department of Energy.

April 19, 2012

OAK RIDGE National Laboratory

Outline

OAK RIDGE National Laboratory

AORSA

All Orders Spectral Algorithm (AORSA)

- Spectral calculations of wave propagation and heating in tokamak plasmas
- Main developers are Fred Jaeger, Lee Berry, Don Batchelor, David Green at ORNL
- SciDAC-1 fusion project, *Numerical Calculations of Wave-Plasma Interactions in Multi-dimensional Systems*, Don Batchelor as PI
- SciDAC-2 fusion project, *Controlling Fusion Plasmas: Electromagnetic Wave Effects Center for Simulation of Wave-Plasma Interactions (CSWPI)*, Paul Bonoli as PI

OAK RIDGE National Laboratory

AORSA

- Models the response of plasma to radio-frequency (RF) waves by solving the inhomogeneous wave equation, or Helmholtz equation

$$-\nabla \times \nabla \times \mathbf{E} + \frac{\omega^2}{c^2}\left(\mathbf{E} + \frac{i}{\omega\epsilon_0}\mathbf{J}_P\right) = -i\omega\mu_0\mathbf{J}_{ant}$$

  where $\mathbf{J}_P$ is plasma wave current computed as an integral operator on $\mathbf{E}$ and $\mathbf{J}_{ant}$ is antenna source current
- Assume weakly non-linear, time average distribution function $f_0(v,t)$ evolves slowly as

$$f(\mathbf{x},\mathbf{v},t) = f_0(\mathbf{x},\mathbf{v},t) + f_1(\mathbf{x},\mathbf{v})e^{-i\omega t}$$

  where $f_1(\mathbf{x},\mathbf{v})\exp(-i\omega t)$ evolves on a fast RF time scale
- Fluctuating plasma current $\mathbf{J}_P$ due to wave

$$\mathbf{J}_P(\mathbf{x}) = \int d\mathbf{x}'\sigma(\mathbf{x},\mathbf{x}'-\mathbf{x})\cdot\mathbf{E}(\mathbf{x}')$$

  where $\mathbf{x}$ is position vector, $\sigma(\mathbf{x},\mathbf{x}-\mathbf{x}')$ is the plasma conductivity kernel

OAK RIDGE National Laboratory

Fourier Harmonics

- The RF electric field is expanded in Fourier harmonics

$$\mathbf{E}(x,y,\varphi) = \sum_{n,m,\ell} \mathbf{E}_{n,m,\ell} e^{i(k_n x + k_m y + \ell\varphi)}$$

  where $x = R - R_0$, $y = Z$, n, m, ℓ are Fourier mode numbers.
- In the limit of a spatially homogeneous plasma

$$\mathbf{J}_P(x,y,\varphi) = \sum_{n,m,\ell} \sigma(k_n, k_m, \ell) \cdot \mathbf{E}_{n,m,\ell} e^{i(k_n x + k_m y + \ell\varphi)}$$

  where $\sigma(k_n, k_m, \ell)$ is the plasma conductivity tensor.
- For tokamak geometry with symmetry about the toroidal axis, the φ harmonics can be uncoupled and calculated separately.

OAK RIDGE National Laboratory

Linear Equations

- Collocation method on a 2D rectangular grid is used to generate the large (dense) system of linear equations.
- System size $N = 3 * M_x * M_y$ where M_x and M_y are the number of Fourier modes used.
- For $M_x = M_y = 256$, system size N is $196,608$, which requires about 618 GBytes of memory and about $20 * 10^{15}$ flops of computation.
- Problem size $N = 3M^2$, storage scales as $O(N^2)$ or $O(M^4)$ and work scales as $O(N^3)$ or $O(M^6)$.
- Complex*16 system solved using ScaLAPACK LU solver.

OAK RIDGE National Laboratory

Performance Optimization

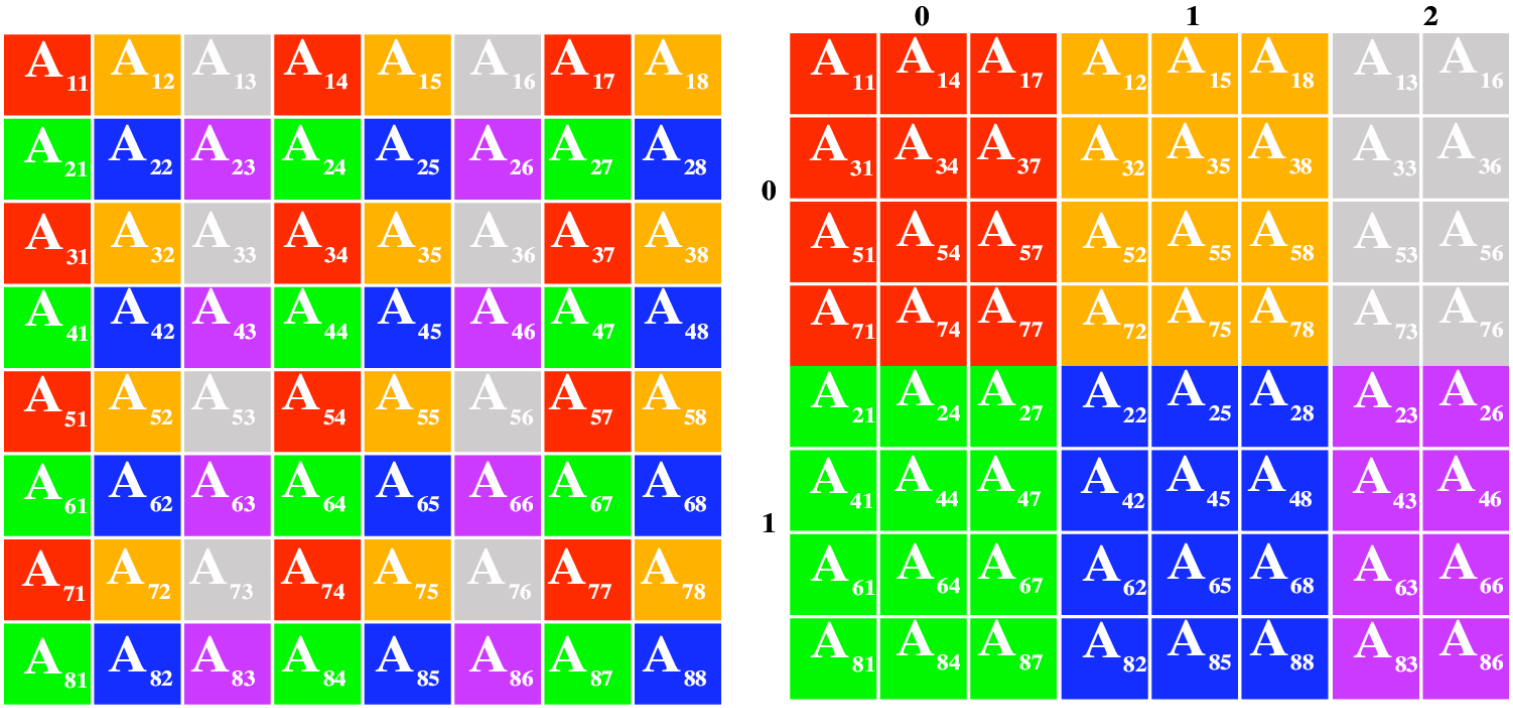
- Reduced system size by considering only points inside the plasma.

$$\begin{pmatrix} A_{11} & A_{12} \\ & I \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b1 \\ b2 \end{pmatrix}$$

  where x_1 are variables inside plasma and x_2 are variables outside plasma
- Roughly about $1/3$ of points outside plasma, so solving with $(2/3N)$ requires about $1/3$ amount of time
- Requires formulation in configuration space (not in Fourier space) to identify points outside plasma.
- Require Fourier transform of entire row of matrix and data redistribution for ScaLAPACK

OAK RIDGE National Laboratory

2D Block Cyclic Distribution



Global (left) and distributed (right) views of matrix

Load Imbalance

- Computation in narrow vertical region near mode conversion layer may lead to load imbalance
- Randomize reordering to redistribute work.
- Reuse LU factorization as preconditioner for a range of parameters
- Reduced precision (complex*8) LU factorization with iterative refinement in some cases
- Difficult to find effective preconditioner for iterative method

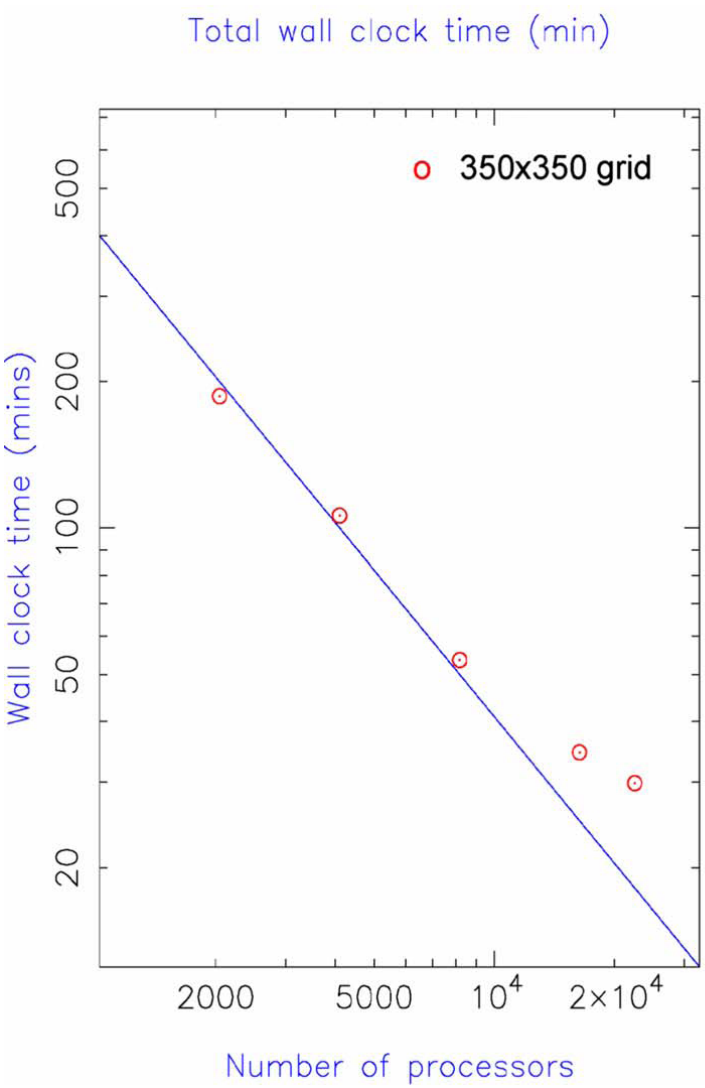
OAK RIDGE National Laboratory

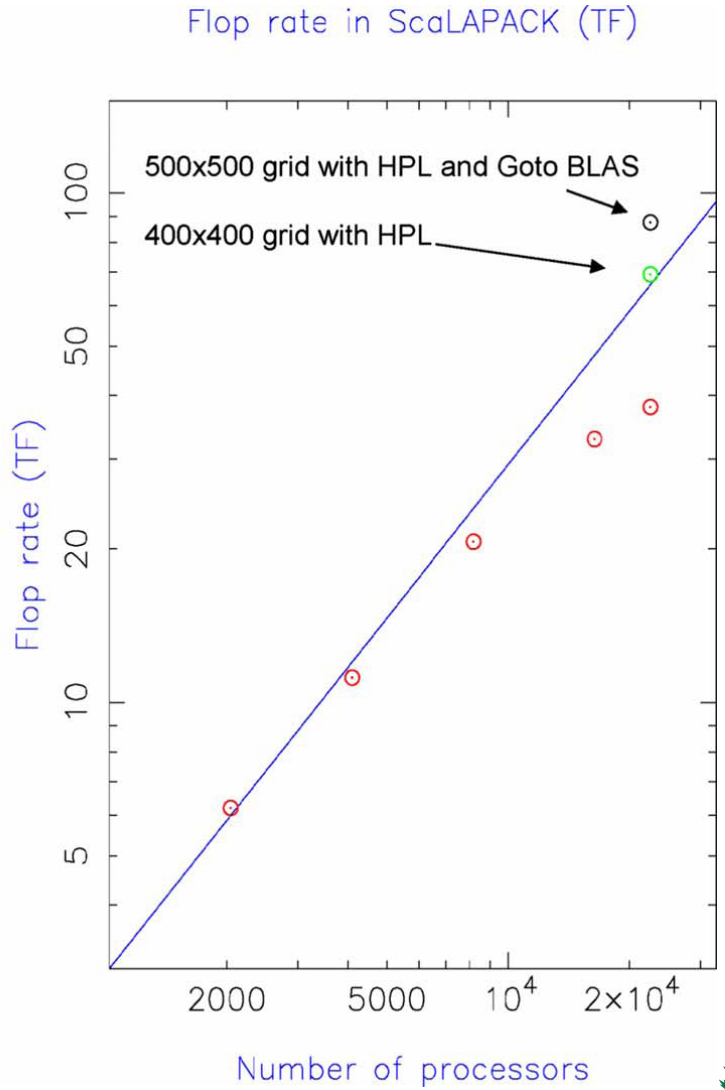
High Performance Linpack (HPL) Benchmark

- Reduced scalability and efficiency of ScaLAPACK over 5000 processors
- HPL benchmark used in TOP500 ranking of world's fastest supercomputers
- Solves real*8 dense linear system in 2D block cyclic distribution, rhs is column $(N+1)$
- Multiple options and enhancements with look-ahead to reduce length of critical path, options for performing broadcasts, recursive left-looking or right-looking factorization methods
- Conversion to use complex*16 and complex*8, modify storage layout, return pivot vector, correct pivoting in lower triangular part

OAK RIDGE National Laboratory

Scalability





OAK RIDGE National Laboratory

Mixed Precision HPL

- Need storage for extra copy of complex*8 matrix
- Not too ill-conditioned, condition number less than 10^7

Table 1: Run times and Gflop rates per core for complex matrix size of 60,000.

Processor grid	Comple*16 ScaLAPACK LU	Complex*8 ScaLAPACK LU	Complex*8 HPL LU	Mixed precision solver
8×8	1580s (5.7GF)	871s (10.3GF)	732s (12.3GF)	757s
16×16	481s (4.7GF)	280s (8.0GF)	213s (10.6GF)	221s
32×32	185s (3.0GF)	117s (4.8GF)	71s (7.9GF)	76s
64×64	104s (1.4GF)	67s (2.1GF)	30s (4.7GF)	34s
128×128	89s (0.4GF)	51s (0.7GF)	19s (1.9GF)	23s

Run times and Gflop rates per core for complex matrix size of 60,000 (Jaguar XT4)

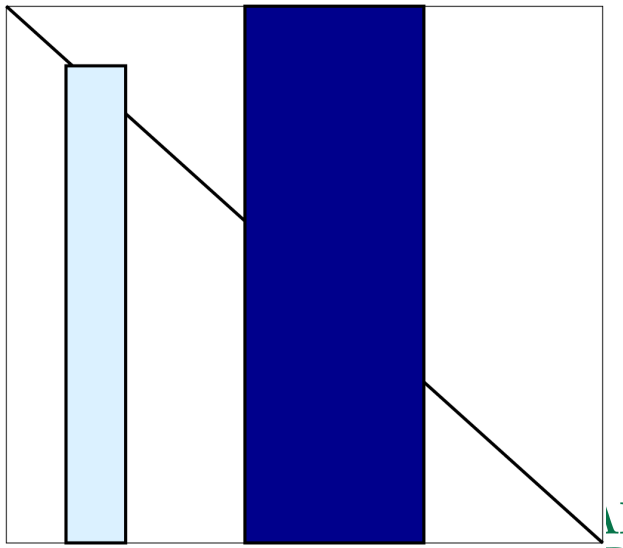
OAK RIDGE National Laboratory

Out-of-core Factorization on GPU

- Bottleneck in data transfer between CPU/GPU
- MAGMA 1.0 limited to matrix in GPU memory
- Out-of-core approach to solve large problems

$$\begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \end{pmatrix}$$

1. Factor diagonal block, $L_{11}U_{11} = A_{11}$
2. Solve for lower triangular part,
   $L_{21}U_{11} = A_{21} \rightarrow L_{21} = A_{21}/U_{11}$
3. Solve for upper triangular part,
   $L_{11}U_{12} = A_{12} \rightarrow U_{12} = L_{11}\ A_{12}$
4. Update and solve for lower part,
   $L_{22}U_{22} = A_{22} - L_{12}U_{21}$



Out-of-core Factorization on GPU

	MKL (12 CPU)	MAGMA 1.0	Out-of-core algorithm
N=25,000	121 Gflops/s	271 Gflops/s	200 Gflops/s
N=35,000	123 Gflops/s	Out of memory	214 Gflops/s

Table: Comparison of MAGMA 1.0 to out-of-core LU factorization (`DGETRF`) using only 1 GBytes out of 5 GBytes of Nvidia M2070.

	MKL (12 CPU)	MAGMA 1.0	Out-of-core algorithm
N=25,000	122 GFlops/s	266 GFlops/s	246 GFlops/s
N=35,000	123 GFlops/s	Out of memory	263 GFlops/s

Table: Comparison of MAGMA 1.0 to out-of-core Cholesky factorization (`DPOTRF`) using only 1 GBytes out of 5 GBytes of Nvidia M2070.

OAK RIDGE National Laboratory

Parallel Out-of-core Matrix Factorization on GPU

- Parallel out-of-core complex*16 LU factorization on 15 nodes of GPU cluster.
- Each node has 12 cores (AMD 2.6 GHz) and two Nvidia M2050 GPU (only 1 used).
- Each M2050 GPU has 4.3 GBytes of device memory.
- Computation used 180 MPI tasks (one MPI task per core).
- Narrow panel factored using ScaLAPACK on CPU.
- Parallel BLAS was used for communication.
- Communication requires moving data between CPU/GPU since MPI has access to only CPU memory.

OAK RIDGE National Laboratory

		ScaLAPACK	Out-of-Core
MB	processor grid	GFLOPs	GFLOPs
64	12×15	1103	1813
128	12×15	1104	1633
64	15×12	1043	1714
128	15×12	885	1554

Table: Comparison of the performance of ScaLAPACK `PZGETRF` with an out-of-core factorization method implemented on both the CPU and GPU for $N = 90,000$ on 180 MPI processes.

Anti-aliasing in AORSA

- *Anti-alias filter in AORSA for modeling ICRF heating of DT plasmas in ITER* by Lee Berry, Don Batchelor, Fred Jaeger, and RF SciDAC Team, presented at 53rd Annual Meeting of the APS Division of Plasma Physics Salt Lake City, Utah, November 14-18, 2011
- Aliasing is a distortion caused in analog-to-digital conversion by a too low rate of data sampling.
- TORIC eliminates aliasing by "over-sampling" in the evaluation of convolutions where the σ is evaluated for twice as many modes as required by the solution

$$\begin{aligned}\mathbf{J}_P(\rho,\vartheta) &= \sum_m \left[\sum_n \sigma^n_{m-n}(\rho)\cdot\mathbf{E}_n\right] e^{im\vartheta} \\ &= \sum_m \mathbf{J}_m e^{im\vartheta}, \quad \mathbf{J}_m = \sum_n \sigma^n_{m-n}(\rho)\cdot\mathbf{E}_n\end{aligned}$$

OAK RIDGE National Laboratory

Antialiasing in AORSA

- AORSA uses collocation

$$\mathbf{J}_P(x_\ell) = \sum_{j=-N}^{N} \sigma(x_\ell, k_j) \cdot \mathbf{E}_{k_j} e^{ik_j x_\ell} \ ,$$

  where ℓ denotes the spatial point (matrix row), and j denotes the mode number (matrix column).

- Fourier transform of the columns of this matrix, the m-th component of $\mathbf{J}$ looks like

$$\mathbf{J}_m = \sum_{n=-N}^{N} \sigma(k_n - k_m, k_n) \cdot \mathbf{E}_{k_n},$$

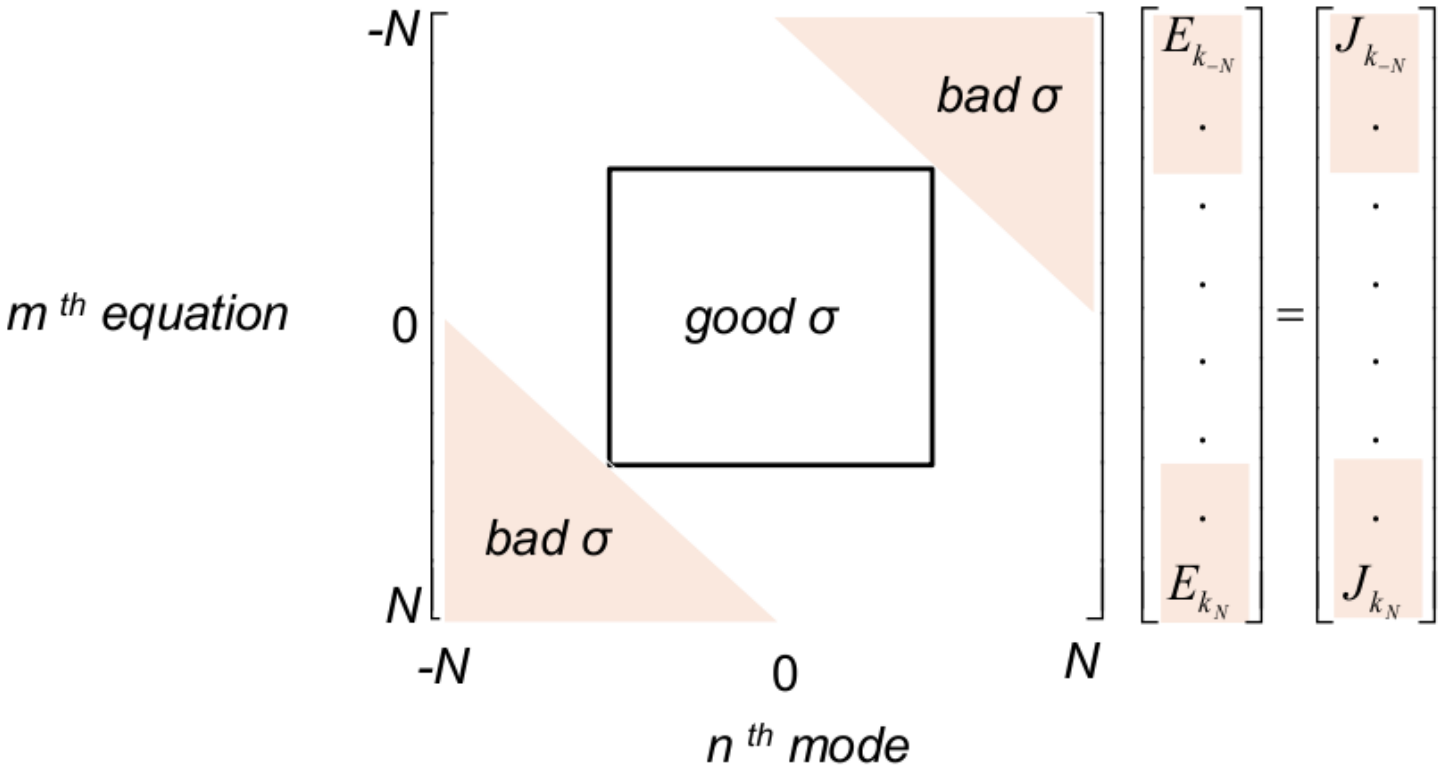
  which is similar to the convolution in TORIC.

OAK RIDGE National Laboratory

AORSA Matrix Viewed in Fourier Space

There are "bad" regions in the matrix where the σ's are aliased

$$\mathbf{J}_m = \sum_{n=-N}^{N} \sigma(k_n - k_m, k_n) \cdot \mathbf{E}_{k_n},$$



De-aliasing by over-samples on fine grid, an filtering modes

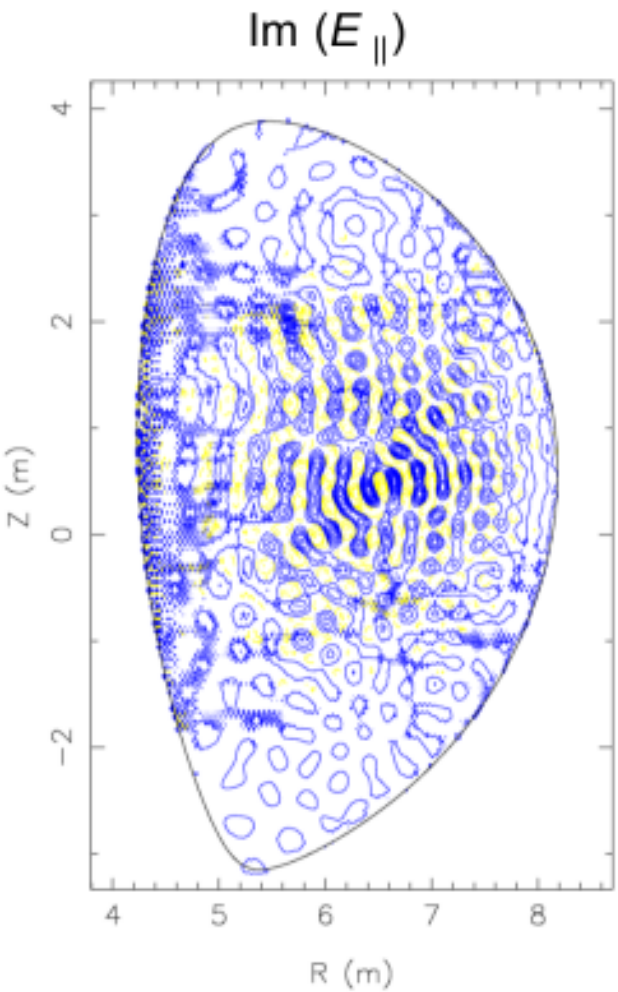
- Fourier transform columns of the large matrix
- Reduce the matrix size by eliminating extra rows and columns that have "bad σ's"
- Fourier transform back to configuration space
- Apply boundary conditions in configuration space for the small problem
- Solve this small "de-aliased" matrix.
- In 2D, it is more difficult and currently solving the large matrix and eliminating the high modes from the solution (very expensive)
- Working on generating and solving the small de-aliased matrix directly rather than solving a larger problem and throwing out half of the solution.

OAK RIDGE National Laboratory

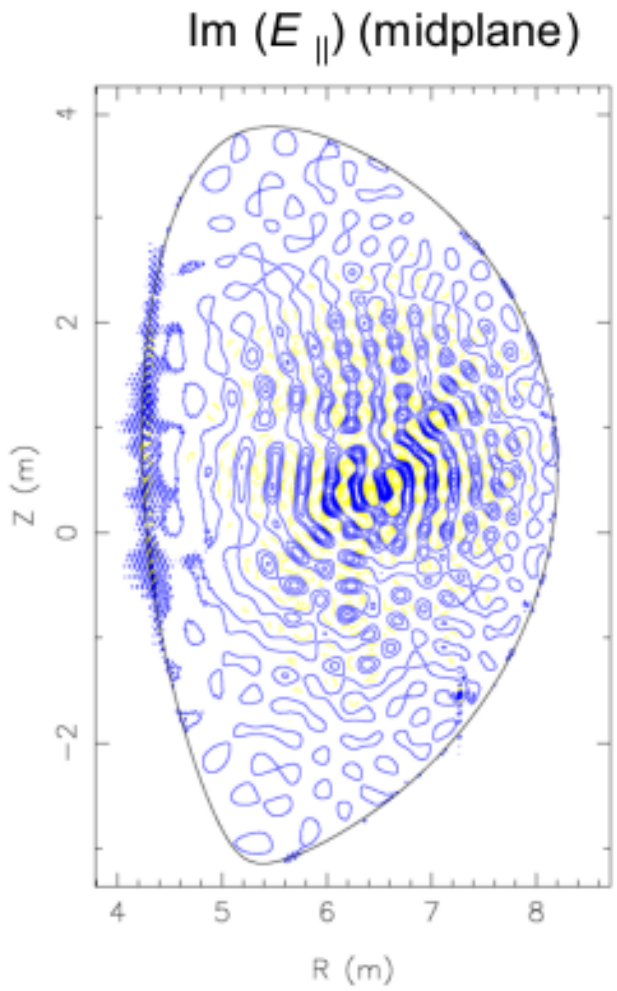
200 modes vs 400 modes with de-aliasing

Short wavelengths close to the edge are better resolved and noise is eliminated.

200x200 (all modes)



400x400 (de-alias 1/2 rule)



OAK RIDGE National Laboratory

Anderson Acceleration

Anderson Acceleration

- Acceleration of fixed point iteration $x_{k+1} = g(x_k)$ for solving mildly non-linear problem, $f(x) = x - g(x) = 0$
- Rediscovered under different names, also known as vector extrapolation, vector ε-algorithm or Anderson mixing in electronic structure computations
- Iteration to self-consistent solution, e.g. AORSA computes global wave solution and velocity distributions, which is passed to CQL3D Fokker-Planck code to compute quasi-linear diffusion coefficients, which are passed back to AORSA and iterate until self-consistency between wave electric field and resonant ion distribution function
- Idea: find weights α_i such that $1 = \sum_i^k \alpha_i$, $\| \sum_i^k \alpha_i f(x_i) \|$ is minimized, then accelerated solution is a weighted combination of previous solutions $x^* = \sum_i^k \alpha_i x_i$
- The method is simple to implement, and does not require Jacobian information

OAK RIDGE National Laboratory

Anderson Acceleration

- If $g(x)$ is linearized as $g(x) = J * x + b$ then the method is equivalent to applying GMRES to $(I - J)x = b$
- Note that GMRES finds x_k in Krylov subspace (Span$(b, Ab, A^2b, \ldots, A^k)$) that minimizes the residual $\|b - Ax_k\|_2$.
- The acceleration is restarted after k iterations since computing weights α_i become more ill-conditioned
- D. G. Anderson, *Iterative procedures for nonlinear integral equations*, J. Assoc. Comput. Machinery, 12 (1965), 547-560
- Avram Sidi, *Methods for Acceleration of Convergence (Extrapolation) of Vector Sequences*. Wiley Encyclopedia of Computer Science and Engineering 2008

OAK RIDGE National Laboratory

Kronecker Products

Kronecker Products

- Kronecker products commonly arise from variable separable formulation, e.g. $f(r,z) = g(r)h(z)$ or approximation on logically rectangular or tensor product grids.
- Let matrix A be $mA \times nA$ and B be $mB \times nB$. For convenience, let them be indexed as $A(ia, ja)$ and $B(ib, jb)$. Let $C = A \otimes B$ (or kron(A, B) in MATLAB notation), then matrix C is size $(mA * mB) \times (nA * nB)$. If matrix A is 3×3, then

$$C = \begin{bmatrix} a_{11}B & a_{12}B & a_{13}B \\ a_{21}B & a_{22}B & a_{23}B \\ a_{31}B & a_{32}B & a_{33}B \end{bmatrix}$$

- Matrix C can be interpreted as a 4-index array $C([ib, ia], [\mathbf{jb}, \mathbf{ja}]) = A(ia, \mathbf{ja}) * B(ib, \mathbf{jb})$, where the composite index $[ib, ia] = ib + (ia - 1) * mB$ is the index in Fortran column-wise order.

Kronecker Product

- Matrix-vector multiply $Y = (A \otimes B) * X$ can be written as very efficient matrix-matrix operations[1],

$$\begin{aligned} Y([ib, ia]) &= C([ib, ia], [\mathbf{jb}, \mathbf{ja}]) * X([\mathbf{jb}, \mathbf{ja}]) \\ &= A(ia, \mathbf{ja}) * B(ib, \mathbf{jb}) * X([\mathbf{jb}, \mathbf{ja}]) \\ &= \boxed{B(ib, \mathbf{jb}) * X(\mathbf{jb}, \mathbf{ja})} * A(ia, \mathbf{ja}) \\ Y &= (B * X) * A^t \end{aligned}$$

- If A and B are $N \times N$, then $C = A \otimes B$ is $N^2 \times N^2$. Kronecker product reduces the work of computing $Y = C * X = (B * X) * A^t$ from $O(N^4)$ to $O(2N^3)$.
- If A and B are sparse matrices, then work is $O(nnz(X)(nnz(A) + nnz(B))$.

[1] We blur the distinction between the matrix $X(jb, ja)$ and vector $X([jb, ja])$ with composite index.

Coding Y = kron(A,B)*X

- Deeply nested loops

```
do ja=1,nA
do jb=1,nB
do ia=1,mA
do ib=1,mB
!   recall C([ib,ia],[jb,ja]) = A(ia,ja)*B(ib,jb)
!  Y([ib,ia]) = C([ib,ia], [jb,ja]) * X([jb,ja])
!
    Y(ib,ia) += B(ib,jb) * X(jb,ja) * A(ia,ja)
enddo; enddo; enddo; enddo
```

- Use array `Z(ib,ja)` to hold common sum over `jb` loop

```
Z(1:mB,1:nA)=matmul(B(1:mB,1:nB),X(1:nB,1:nA))
Y(1:mB,1:mA)=matmul(Z,transpose(A(1:mA,1:nA)))
```

Properties of Kronecker Products

- Other interesting properties of Kronecker products are summarised below,

$$(A + B) \otimes E = A \otimes E + B \otimes E$$
$$(A \otimes B) \otimes E = A \otimes (B \otimes E)$$
$$(A \otimes B)^t = (A^t \otimes B^t)$$

- Reduction of work from $O(N^6)$ to $O(N^3)$.

$$(A \otimes B) * (E \otimes F) = (A * E) \otimes (B * F)$$
$$\mathrm{eig}(A \otimes B) = \mathrm{eig}(A) \otimes \mathrm{eig}(B)$$
$$\det(A \otimes B) = \det(A)^n \det(B)^m, \quad \text{where } A \text{ is } m \times m,\ B \text{ is } n \times n.$$
$$(A \otimes B)^{-1} = (A^{-1} \otimes B^{-1})$$

- If $N = 1000$, $N^6 = 10^{18}$ requires petascale computing, whereas $N^3 = 10^9$ can be performed in seconds on a desktop computer.

1D Interpolation

- Approximate function $f(x) \approx \sum_{j=1}^{n} c_j \phi_j(x)$.
- Basis function $\phi_j(x)$ may be Chebyshev polynomials, B-splines, wavelets or Fourier basis.
- Interpolation condition of tabulated values is used to find c_j's, $f(x_i) = \sum_{j=1}^{n} c_j \phi_j(x_i)$.
- Coefficient c_j's are obtained by solving the linear system $[f(x_i)] = T_x * [c_j]$.
- Note T_x may be sparse if $\{\phi_j(x)\}$ have compact support

$$T_x = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_n) & \cdots & \phi_n(x_n) \end{pmatrix} .$$

OAK RIDGE National Laboratory

1D Interpolation

- Evaluation at new set of points $[f(\tilde{x}_i)]$ is computed as matrix multiply, $[f(\tilde{x}_i)] = T_{\tilde{x}} * [c_j]$,

$$T_{\tilde{x}} = \begin{pmatrix} \phi_1(\tilde{x}_1) & \cdots & \phi_n(\tilde{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\tilde{x}_m) & \cdots & \phi_n(\tilde{x}_m) \end{pmatrix} .$$

2D Interpolation

- Tabulated values $f(x_i, y_j)$ on a rectangular grid is approximated as $f(x,y) \approx \sum_{k,\ell} c_{k\ell}\phi_k(x)\phi_\ell(y)$ (variable separable).
- Interpolation conditions lead to

$$\begin{aligned} \mathrm{vec}(F) &= (T_y \otimes T_x)\,\mathrm{vec}(C) \\ \mathrm{vec}(C) &= \left(T_y^{-1} \otimes T_x^{-1}\right)\mathrm{vec}(F) \\ C &= T_x^{-1} F T_y^{-t} \ , \end{aligned}$$

  where $F = [f(x_i, y_j)]$ $C = [c_{k\ell}]$.
- Evaluation at new points $(\tilde{x}_i, \tilde{y}_j)$ is computed as

$$\left[\tilde{F}_{ij}\right] = (T_{\tilde{y}} \otimes T_{\tilde{x}})\,[c_{k\ell}] = T_{\tilde{x}} C {T_{\tilde{y}}}^t \ .$$

OAK RIDGE National Laboratory

Higher Dimensions

- Interpolate tabulated values of 4-index function on rectangular (tensor product) grid,

$$f(w,x,y,z) \approx \sum_{ijk\ell} c_{ijk\ell}\phi_i(w)\phi_j(x)\phi_k(y)\phi_\ell(z) \ .$$

- Coefficients computed efficiently as

$$\begin{aligned} \mathrm{vec}(F) &= (T_z \otimes T_y \otimes T_x \otimes T_w)\,\mathrm{vec}(C) \\ \mathrm{vec}(C) &= \left(T_z^{-1} \otimes T_y^{-1} \otimes T_x^{-1} \otimes T_w^{-1}\right)\mathrm{vec}(F) \ . \end{aligned}$$

- Evaluation at new points is computed as

$$\mathrm{vec}(\tilde{F}) = \left(T_{\tilde{z}} \otimes T_{\tilde{y}} \otimes T_{\tilde{x}} \otimes T_{\tilde{w}}\right)\mathrm{vec}(C) \ .$$

OAK RIDGE National Laboratory

Nearest Kronecker Product

- Given $mn \times mn$ matrix C, find $m \times m$ matrices A, $n \times n$ matrix B to minimise $\|C - A \otimes B\|_F$.
- Let matrix $\tilde{C}$ be a rearrangement of entries of matrix C

$$\tilde{C}([ia, \mathbf{ja}], [ib, \mathbf{jb}]) = C([ib, ia], [\mathbf{jb}, \mathbf{ja}])$$

- Equivalent to reshaping C as 4 index array, then permuting the axis
  $\tilde{C}$= `permute( reshape(C, [mB,mA, nB,nA]), [2, 4, 1,3]);`
- The matrices A, B are obtained from the singular vectors (optimal rank-1 approximation) of matrix $\tilde{C}([ia, ja], [ib, jb]) \approx \sigma_1 \mathbf{a}\mathbf{b}^t$

$$\begin{aligned} C([ib, ia], [\mathbf{jb}, \mathbf{ja}]) &\approx A([ia, \mathbf{ja}]) * B([ib, \mathbf{jb}]) \\ \tilde{C}([ia, \mathbf{ja}], [ib, \mathbf{jb}]) &\approx \sigma_1 \mathbf{a}\mathbf{b}^t \\ &\approx A([ia, \mathbf{ja}]) * B([ib, \mathbf{jb}]) \end{aligned}$$

- C. van Loan, *The Ubiquitous Kronecker Product*, Journal of Computational and Applied Mathematics, 1(**123**), pages 85-100, 2000.

OAK RIDGE National Laboratory

Generalized Result

- Idea: Rewrite sum of Kronecker products in decreasing importance of Kronecker products.
- Given a sum of Kronecker products, $C = \sum_k^r A_k \otimes B_k$, we can rewrite the same expression as $C = \sum_k^r \sigma_k \tilde{A}_k \otimes \tilde{B}_k$ where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$ and $\|\tilde{A}_k\|_F = \|\tilde{B}_k\|_F = 1$.
- Result follows from Singular Value Decomposition (SVD) of rearranged matrix E, which has rank r.

$$\begin{aligned}
\tilde{C} &= [A_1(:)|\ldots|A_r(:)]\,[B_1(:)|\ldots|B_r(:)]^t = \mathcal{A}\mathcal{B}^t \\
&= Q_A R_A (Q_B R_B)^t = Q_A (R_A R_B^t) Q_B^t \\
&= Q_A (U D V^t) Q_B^t = (Q_A U) D (Q_B V)^t = \tilde{\mathcal{A}} D \tilde{\mathcal{B}}^t \\
\tilde{C} &= \left[\tilde{A}_1(:)|\ldots|\tilde{A}_r(:)\right] \mathrm{diag}(\sigma_1, \ldots, \sigma_r) \left[\tilde{B}_1(:)|\ldots|\tilde{B}_r(:)\right]
\end{aligned}$$

where QR factorization (or Gram-Schmidt orthogonalization) of $\mathcal{A}$ and $\mathcal{B}$ gives Q_A, Q_B with orthogonal columns with R_A and R_B are $r \times r$ upper triangular matrices

Nearest Kronecker Product

- If A_k or B_k have prescribed sparsity pattern (say A_k is tridiagonal and C is block tridiagonal), we can restrict SVD to non-zero entries
- For example, let A be an $m \times m$ tridiagonal matrix, and $A(:) = Pa$ where $(3m-2) \times 1$ vector a holds the non-zero entries of A, and matrix P $(m^2 \times (3m-2))$ consists of columns of identity matrix that correspond to tridiagonal positions in A. Note that $P^t P$ is the identity matrix of size $(3m-2) \times (3m-2)$.
- Idea: Compute SVD of $P^t \tilde{C}$ $((3m-2) \times nnz(B))$, then expand $\tilde{A}(:) = Pa$
- Adding or removing a Kronecker product is related to up-dating or down-dating the QR factorization of $\mathcal{A}$ and $\mathcal{B}$.

OAK RIDGE National Laboratory

Preconditioning by Kronecker Products

- If $C \approx A \otimes B$, then $(A \otimes B)^{-1} = (A^{-1} \otimes B^{-1})$ may be an effective preconditioner.
- If $C \approx A_1 \otimes B_1 + A_2 \otimes B_2$, then consider a fast solver for sum of 2 Kronecker products.
- If $C \approx \sum_k^r \tilde{A}_k \otimes \tilde{B}_k$, then rewrite sum in decreasing order of importance and use first few terms as preconditioner.

OAK RIDGE National Laboratory

Fast Solver for Sum of 2 Kronecker Products

- Fast solver for $C = A_1 \otimes B_1 + A_2 \otimes B_2$ by computing the QZ decomposition of A_1, A_2.
- Find common matrices U_A, V_A to simultaneously transform $U_A * A_1 * V_A = D_{A_1}$, $U_A * A_2 * V_A = D_{A_2}$ to diagonal matrices.
- Solve $Cx = r$ by transforming system to diagonal matrices $U = U_A \otimes U_B$, $V = V_A \otimes V_B$, to solve $UC(Vy) = (Ur)$, $x = Vy$,

$$
\begin{aligned}
UCV &= (U_A \otimes U_B) C (V_A \otimes V_B) \\
&= (U_A \otimes U_B)(A_1 \otimes B_1 + A_2 \otimes B_2)(V_A \otimes V_B) \\
&= (U_A A_1 V_A) \otimes (U_B B_1 V_B) + (U_A A_2 V_A) \otimes (U_B B_2 V_B) \\
&= (D_{A_1} \otimes D_{B_1}) + (D_{A_2} \otimes D_{B_2}), \quad \text{is diagonal.}
\end{aligned}
$$

OAK RIDGE National Laboratory

QZ Decomposition

- $QZ(A, B)$ generalized eigen decomposition to produce eigenvectors V, triangular $\mathcal{A}$, $\mathcal{B}$, and orthogonal Q, Z, such that $\alpha = \text{diag}(\mathcal{A})$, $\beta = \text{diag}(\mathcal{B})$, $\lambda = \alpha/\beta$,

$$\mathcal{A} = QAZ, \quad \mathcal{B} = QBZ, \quad AV\,\text{diag}(\beta) = BV\,\text{diag}(\alpha)$$

- Let $U = \tilde{U}^{-1}$ where

$$\tilde{U}(:, i) = \begin{cases} AV(:, i) & \text{if } |\alpha_i| \geq |\beta_i| \\ BV(:, i) & \text{otherwise} \end{cases}$$

  then UAV is diagonal, $U(BV\,\text{diag}(\alpha)) = U(AV\,\text{diag}(\beta))$ is also diagonal.
- Problem if V (or $\tilde{U}$) is not full rank or ill-conditioned.

General Sum of Kronecker Products

- No (known) general fast solver for sum of 3 or more Kronecker products.
- Finite difference method for incompressible Navier-Stokes equations on staggered rectangular grid

$$L = C_\ell \otimes C_n \otimes A_m + C_\ell \otimes B_n \otimes C_m + E_\ell \otimes C_n \otimes C_m$$

  where A_m, B_n, E_ℓ, and C matrices are tridiagonal. Derive transformations so that $(I \otimes U_A \otimes U_B)L(I \otimes V_A \otimes V_B)$ is tridiagonal.
- Kamm and Nagy considered the approximation $C = \sum_k^r A_k \otimes B_k \approx UDV'$, for where $U = U_A \otimes U_B$, $V = V_A \otimes V_B$, are obtained from SVD $A_1 = U_A D_A V_A'$, $B_1 = U_B D_B V_B'$ and $D = \mathrm{diag}(U'(C)V)$.
- Ford and Tyrtyshnikov considered a discrete wavelet transform on a sum of Kronecker products ($D = W'CW$, W is wavelet transform) to obtain a sufficiently sparse representation so that a sparse direct solver can be applied.

Resources about Kronecker Products

- *The Ubiquitous Kronecker Product* by C. Van Loan, Journal of Computational and Applied Mathematics, 123(2000), pp. 85-100.
- *Approximation with Kronecker Products* by C. Van Loan and N. Pitsianis in Linear Algebra for Large Scale and Real Time Applications, M. S. Moonen and G. H. Golub, eds., Kluwer Publications, 1993, pp. 293-314. (See also `http://www.cs.duke.edu/~nikos/KP/home.html`).
- *The Kronecker product and stochastic automata networks* by Amy N. Langville and William J. Stewart, J. Comput. Appl. Math., 167(2004) 429-447.
- *Computational Frameworks for the Fast Fourier Transform* by C. Van Loan, SIAM, 1992.

OAK RIDGE National Laboratory

- *Combining Kronecker product approximation with discrete wavelet transform to solve dense function-related linear systems* by J. M. Ford and E. E. Tyrtyshnikov, SIAM J. Sci. Comput., 25(3):861-981, 2003.
- *A fast Poisson solver for the finite difference solution of the incompressible Navier-Stokes equations* by G. H. Golub, L. C. Huang, H. Simon, W.-P. Tang, SIAM J. Sci. Comput., 19(5):1606-1624,1998.
- *Kronecker product and SVD approximations in image restoration* by J. Kamm and J. G. Nagy, Linear Algebra and its Applications, 284:177-192, 1998.

OAK RIDGE National Laboratory

Extremescale Computing

OAK RIDGE National Laboratory

Challenges in Exascale (10^{18})Computing

- The major issue is energy.
- Moving data is the largest cost of energy.
- **It is not about FLOPS. It is about data movement.**
- Exascale machine may have only 64 PBytes of memory
- Need massive amounts ($O(10^9)$ threads) of parallelism. Global synchronization will be very expensive.
- Need to use vectorization (AVX, SSE4 instructions, GPU/MIC accelerator).
- Need attention on fault-tolerance or resilience.

OAK RIDGE National Laboratory

Algorithm Needs

- Low order FV/FEM, unstructured mesh, incomplete LU preconditioner, classic Krylov with many synchronizations for dot products may not fully exploit future machine
- May need reformulation in
  - regular refinement on grid cells, variables on boundaries of cells
  - high-order discretization
  - parallel in time method
  - relaxed synchronization (asynchronous global sums and dot products)
  - reduce data movement by computing in blocks (e.g. out-of-core algorithm)

OAK RIDGE National Laboratory

Resources on Exascale Computing

- Workshop reports on exascale applications `http://science.energy.gov/ascr/news-and-resources/program-documents/`
- Scientific Grand Challenges: Fusion Energy Sciences and the Role of Computing at the Extreme Scale, `http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Fusion_report.pdf`
- International exascale software project, `http://www.exascale.org/iesp/Main_Page`
- Peter M. Kogge (editor), *ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems*, Univ. of Notre Dame, CSE Dept. Tech. Report TR-2008-13, Sept. 28, 2008, `http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf`
- Extreme-Scale Solvers Workshop, `http://www.orau.gov/extremesolvers2012/`

OAK RIDGE National Laboratory

Summary

- AORSA wave code uses Fourier basis and requires the parallel solution of large dense complex linear system.
- Current development on 2D anti-aliasing.
- Anderson acceleration (vector extrapolation) for fixed-point iteration using weighted combination of previous iterates.
- Variable separable formulation on rectangular grids commonly leads to Kronecker products.
- Preconditioner based on sum of Kronecker products.
- Extremescale computing will place emphasis on reducing data movement instead of flops.

OAK RIDGE National Laboratory